Banc GSM émulé — dossier de run

/opt/GSM/osmo-operator

Banc GSM émulé — banc-max

2026-10-25

Table des matières

1 Synthèse **10**

2 Resultats des tests **11**

2.1 /opt/GSM/osmo-operator/README.md . . . 11
2.1.1 osmo-operator . . . 11
2.2 /opt/GSM/osmo-operator/.github/release-notes.md . . . 14
2.3 /opt/GSM/osmo-operator/configs/srsran/user_db.csv . . . 15
2.4 /opt/GSM/osmo-operator/data/adresses.txt . . . 16
2.5 /opt/GSM/osmo-operator/data/air-mesh.txt . . . 16
2.6 /opt/GSM/osmo-operator/data/c139.txt . . . 17
2.7 /opt/GSM/osmo-operator/data/noms.txt . . . 17
2.8 /opt/GSM/osmo-operator/data/op.txt . . . 18
2.9 /opt/GSM/osmo-operator/data/pays.txt . . . 18
2.10 /opt/GSM/osmo-operator/doc/README-architecture.md . . . 19
2.10.1 osmo-operator — architecture . . . 19
2.11 /opt/GSM/osmo-operator/doc/README-arm.md . . . 21
2.11.1 osmo-operator — arm64 / Raspberry Pi 4 . . . 21
2.12 /opt/GSM/osmo-operator/doc/README-docker.md . . . 22
2.12.1 osmo-operator — Docker . . . 22
2.13 /opt/GSM/osmo-operator/doc/README-docs.md . . . 24
2.13.1 osmo-operator — documentation map . . . 24
2.14 /opt/GSM/osmo-operator/doc/README-dsp.md . . . 27
2.14.1 osmo-operator — the DSP chain (`--dsp`) . . . 27
2.15 /opt/GSM/osmo-operator/doc/README-grgsm.md . . . 28
2.15.1 osmo-operator — the gr-gsm layer 1 . . . 29
2.16 /opt/GSM/osmo-operator/doc/README-iso.md . . . 30
2.16.1 osmo-operator — the live ISO . . . 31
2.17 /opt/GSM/osmo-operator/doc/README-lte.md . . . 32
2.17.1 osmo-operator — 4G, CSFB and the postmarketOS phone . . . 33
2.18 /opt/GSM/osmo-operator/doc/README-native.md . . . 34
2.18.1 osmo-operator — native install (no Docker) . . . 34
2.19 /opt/GSM/osmo-operator/doc/README-ss7.md . . . 35
2.19.1 osmo-operator — multi-operator, SS7 and WAN . . . 35
2.20 /opt/GSM/osmo-operator/doc/README-tests.md . . . 38
2.20.1 osmo-operator — tests and reference results . . . 38
2.21 /opt/GSM/osmo-operator/doc/README-usage.md . . . 41
2.21.1 osmo-operator — using the bench . . . 41
2.22 /opt/GSM/osmo-operator/doc/c54x_exe-depuis-ubuntu-vierge.md . . . 43
2.22.1 c54x_exe depuis une Ubuntu 24.04 vierge . . . 43
2.23 /opt/GSM/osmo-operator/doc/c54x_exe-from-scratch-ubuntu.en.md . . . 47
2.23.1 c54x_exe from a clean Ubuntu 24.04 . . . 47
2.24 /opt/GSM/osmo-operator/doc/install-from-scratch.en-notes.md . . . 50
2.24.1 English notes per Dockerfile stage (read by tools/dockerfile-to-install-md.sh –lang en) . . . 50

2.25 /opt/GSM/osmo-operator/doc/install-from-scratch.en.md . . . 51
2.25.1 Installing from scratch (generated from the Dockerfile) . . . 51
2.25.2 Appendix: c54x_exe-from-scratch-ubuntu.en.md . . . 66
2.26 /opt/GSM/osmo-operator/doc/install-from-scratch.md . . . 69
2.26.1 Installation depuis zero (generee depuis le Dockerfile) . . . 69
2.26.2 Annexe : c54x_exe-depuis-ubuntu-vierge.md . . . 94
2.27 /opt/GSM/osmo-operator/doc/magnetite.md . . . 97
2.27.1 FreeCalypso Magnetite dans qosmo (le vrai firmware TI, cible Motorola C139) . . . 97
2.28 /opt/GSM/osmo-operator/environment/README.md . . . 99
2.28.1 `environment/` — la configuration, par domaine . . . 99
2.29 /opt/GSM/osmo-operator/navigation/QUICKSTART.md . . . 101
2.29.1 Console SS7 - demarrage rapide . . . 101
2.30 /opt/GSM/osmo-operator/navigation/STATUS.md . . . 104
2.30.1 Etat des lieux et reste a faire . . . 104
2.31 /opt/GSM/osmo-operator/pont/README.md . . . 108
2.31.1 Chiffrement A5, Kc et SACCH — ce qui a été mesuré, et pourquoi c'est écrit ainsi . . . 108
2.32 /opt/GSM/osmo-operator/services/README.md . . . 118
2.32.1 `services/` — unités systemd . . . 118
2.33 /opt/GSM/osmo-operator/wiki/Build.md . . . 119
2.33.1 Build — Docker image, ISO, `.deb` packages, native install . . . 120
2.34 /opt/GSM/osmo-operator/wiki/Environnement.md . . . 123
2.34.1 Environment — who wins, and what reaches QEMU . . . 123
2.35 /opt/GSM/osmo-operator/wiki/Home.md . . . 126
2.35.1 osmo-operator — Quick start & reference . . . 127
2.36 /opt/GSM/osmo-operator/wiki/ISO.md . . . 131
2.36.1 The osmo-operator ISO — full description, build, installation, capabilities . . . 131
2.37 /opt/GSM/osmo-operator/wiki/Installer-c54x_exe.md . . . 146
2.37.1 Installing c54x_exe (the Calypso DSP, TI mask ROM) and the DSP bridge . . . 146
2.38 /opt/GSM/osmo-operator/wiki/Installer-grgsm_exe.md . . . 150
2.38.1 Installing grgsm_exe — and the gr-gsm setup . . . 150
2.39 /opt/GSM/osmo-operator/wiki/Installer-qosmo.md . . . 152
2.39.1 Installing qosmo (QEMU Calypso) — and the osmocom-bb firmware . . . 152
2.40 /opt/GSM/osmo-operator/wiki/Lancer-a-la-main.md . . . 156
2.40.1 Launching the emulated phone by hand, process by process . . . 156
2.41 /opt/GSM/osmo-operator/wiki/Payload-ISM.md . . . 160
2.41.1 SMS payload over ISM radio, the rest over IP (`--payload-ism`) . . . 160
2.42 /opt/GSM/osmo-operator/wiki/README-notebook.md . . . 162
2.42.1 The install notebook . . . 162
2.43 /opt/GSM/osmo-operator/wiki/Reference.md . . . 165
2.43.1 osmo-operator-desktop.iso . . . 165
2.43.2 osmo-operator — a full GSM network on one machine . . . 168
2.44 /opt/GSM/osmo-operator/wiki/Resultats.md . . . 182
2.44.1 Reference results — `checks/check_all.sh` . . . 182
2.45 /opt/GSM/osmo-operator/wiki/SIM.md . . . 183
2.45.1 SIM: the bench's card in an Omnikey HID reader, and softSIM . . . 184
2.46 /opt/GSM/osmo-operator/wiki/Start-native.md . . . 186
2.46.1 `start-native.sh` — step by step . . . 186
2.47 /opt/GSM/osmo-operator/wiki/Telephone-emule.md . . . 190
2.47.1 Installing the emulated phone on your own Osmocom stack . . . 190
2.48 /opt/GSM/osmo-operator/wiki/WAN.md . . . 194
2.48.1 WAN — linking two nodes . . . 194

3 Fichiers **198**
3.1 /opt/GSM/osmo-operator/.gitignore . . . 198
3.2 /opt/GSM/osmo-operator/Dockerfile . . . 198
3.3 /opt/GSM/osmo-operator/Dockerfile.lite . . . 218
3.4 /opt/GSM/osmo-operator/Dockerfile.run . . . 220
3.5 /opt/GSM/osmo-operator/Dockerfile.stp . . . 223
3.6 /opt/GSM/osmo-operator/addition.sh . . . 226
3.7 /opt/GSM/osmo-operator/build-iso.sh . . . 246
3.8 /opt/GSM/osmo-operator/build.sh . . . 248

3.9 /opt/GSM/osmo-operator/compose.sh . . . 254
3.10 /opt/GSM/osmo-operator/compose.yaml . . . 257
3.11 /opt/GSM/osmo-operator/full_qmd.sh . . . 260
3.12 /opt/GSM/osmo-operator/full_text.sh . . . 269
3.13 /opt/GSM/osmo-operator/generate_configs.sh . . . 272
3.14 /opt/GSM/osmo-operator/globals.conf . . . 287
3.15 /opt/GSM/osmo-operator/globals.conf.example . . . 294
3.16 /opt/GSM/osmo-operator/hellogist . . . 300
3.17 /opt/GSM/osmo-operator/install.sh . . . 300
3.18 /opt/GSM/osmo-operator/launch.sh . . . 303
3.19 /opt/GSM/osmo-operator/reset . . . 315
3.20 /opt/GSM/osmo-operator/set_stp_ip.sh . . . 315
3.21 /opt/GSM/osmo-operator/ss7-console.py . . . 324
3.22 /opt/GSM/osmo-operator/start-native.sh . . . 324
3.23 /opt/GSM/osmo-operator/start.sh . . . 325
3.24 /opt/GSM/osmo-operator/test . . . 325
3.25 /opt/GSM/osmo-operator/tout-en-un.py . . . 325
3.26 /opt/GSM/osmo-operator/update.sh . . . 331
3.27 /opt/GSM/osmo-operator/.deb-cache/.keep . . . 348
3.28 /opt/GSM/osmo-operator/.github/actions/docker-disk/action.yml . . . 348
3.29 /opt/GSM/osmo-operator/.github/workflows/build-arm.yml . . . 350
3.30 /opt/GSM/osmo-operator/.github/workflows/build-iso.yml . . . 355
3.31 /opt/GSM/osmo-operator/.github/workflows/docker-arm.yml . . . 362
3.32 /opt/GSM/osmo-operator/.github/workflows/docker.yml . . . 365
3.33 /opt/GSM/osmo-operator/armbian/build-armbian.sh . . . 371
3.34 /opt/GSM/osmo-operator/armbian/userpatches/customize-image.sh . . . 376
3.35 /opt/GSM/osmo-operator/checks/_mode.sh . . . 379
3.36 /opt/GSM/osmo-operator/checks/annuaire.sh . . . 393
3.37 /opt/GSM/osmo-operator/checks/check_all.sh . . . 396
3.38 /opt/GSM/osmo-operator/checks/diag-interstp.sh . . . 404
3.39 /opt/GSM/osmo-operator/checks/diag-stp-operator.sh . . . 408
3.40 /opt/GSM/osmo-operator/checks/global_check.sh . . . 410
3.41 /opt/GSM/osmo-operator/checks/operator_summary.sh . . . 419
3.42 /opt/GSM/osmo-operator/checks/ss7_check.sh . . . 427
3.43 /opt/GSM/osmo-operator/checks/vty-debug-dump.sh . . . 435
3.44 /opt/GSM/osmo-operator/checks/wan_ss7_check.sh . . . 442
3.45 /opt/GSM/osmo-operator/configs/asound.conf . . . 447
3.46 /opt/GSM/osmo-operator/configs/extensions.conf . . . 448
3.47 /opt/GSM/osmo-operator/configs/info.html . . . 457
3.48 /opt/GSM/osmo-operator/configs/linphonerc . . . 461
3.49 /opt/GSM/osmo-operator/configs/manager-osmo.conf . . . 463
3.50 /opt/GSM/osmo-operator/configs/mobile.cfg . . . 463
3.51 /opt/GSM/osmo-operator/configs/modules.conf . . . 466
3.52 /opt/GSM/osmo-operator/configs/osmo-bsc.cfg . . . 468
3.53 /opt/GSM/osmo-operator/configs/osmo-bts-trx-bts1.cfg . . . 474
3.54 /opt/GSM/osmo-operator/configs/osmo-bts-trx.cfg . . . 475
3.55 /opt/GSM/osmo-operator/configs/osmo-bts-virtual.cfg . . . 476
3.56 /opt/GSM/osmo-operator/configs/osmo-bts.cfg . . . 478
3.57 /opt/GSM/osmo-operator/configs/osmo-ggsn.cfg . . . 480
3.58 /opt/GSM/osmo-operator/configs/osmo-hlr.cfg . . . 481
3.59 /opt/GSM/osmo-operator/configs/osmo-mgw.cfg . . . 483
3.60 /opt/GSM/osmo-operator/configs/osmo-msc.cfg . . . 485
3.61 /opt/GSM/osmo-operator/configs/osmo-pcu.cfg . . . 488
3.62 /opt/GSM/osmo-operator/configs/osmo-sgsn.cfg . . . 489
3.63 /opt/GSM/osmo-operator/configs/osmo-sip-connector.cfg . . . 489
3.64 /opt/GSM/osmo-operator/configs/osmo-stp.cfg . . . 490
3.65 /opt/GSM/osmo-operator/configs/osmo-trx.cfg . . . 491
3.66 /opt/GSM/osmo-operator/configs/osmo-upf.cfg . . . 492
3.67 /opt/GSM/osmo-operator/configs/pjsip.conf . . . 492
3.68 /opt/GSM/osmo-operator/configs/rtp.conf . . . 498
3.69 /opt/GSM/osmo-operator/configs/sms-routing.conf . . . 498

3.70 /opt/GSM/osmo-operator/configs/conky/dino.html . . . 499
3.71 /opt/GSM/osmo-operator/configs/conky/labgsm.html . . . 528
3.72 /opt/GSM/osmo-operator/configs/conky/osmo-conky.conf . . . 531
3.73 /opt/GSM/osmo-operator/configs/conky/osmo-conky.lua . . . 532
3.74 /opt/GSM/osmo-operator/configs/crypthome/crypthome-fond . . . 534
3.75 /opt/GSM/osmo-operator/configs/crypthome/crypthome-postinstall . . . 535
3.76 /opt/GSM/osmo-operator/configs/crypthome/crypthome-session-setup . . . 538
3.77 /opt/GSM/osmo-operator/configs/crypthome/crypthome-session.desktop . . . 539
3.78 /opt/GSM/osmo-operator/configs/crypthome/crypthome.default . . . 539
3.79 /opt/GSM/osmo-operator/configs/crypthome/init-crypthome.sh . . . 541
3.80 /opt/GSM/osmo-operator/configs/crypthome/lock-home . . . 545
3.81 /opt/GSM/osmo-operator/configs/crypthome/refresh-owner-apps . . . 546
3.82 /opt/GSM/osmo-operator/configs/crypthome/run-as-owner . . . 547
3.83 /opt/GSM/osmo-operator/configs/crypthome/unlock-home . . . 549
3.84 /opt/GSM/osmo-operator/configs/crypthome/unlock-home.service . . . 556
3.85 /opt/GSM/osmo-operator/configs/gprs/modem-gprs.cfg . . . 556
3.86 /opt/GSM/osmo-operator/configs/gprs/osmo-bsc-bts2-egprs.frag . . . 557
3.87 /opt/GSM/osmo-operator/configs/gprs/osmo-bts-virtual-gprs.cfg . . . 558
3.88 /opt/GSM/osmo-operator/configs/gprs/osmo-pcu-gprs.cfg . . . 559
3.89 /opt/GSM/osmo-operator/configs/open5gs/hss.yaml . . . 559
3.90 /opt/GSM/osmo-operator/configs/open5gs/mme.yaml . . . 560
3.91 /opt/GSM/osmo-operator/configs/open5gs/nrf.yaml . . . 561
3.92 /opt/GSM/osmo-operator/configs/open5gs/pcrf.yaml . . . 563
3.93 /opt/GSM/osmo-operator/configs/open5gs/sgwc.yaml . . . 565
3.94 /opt/GSM/osmo-operator/configs/open5gs/sgwu.yaml . . . 566
3.95 /opt/GSM/osmo-operator/configs/open5gs/smf.yaml . . . 567
3.96 /opt/GSM/osmo-operator/configs/open5gs/subscribers.json . . . 574
3.97 /opt/GSM/osmo-operator/configs/open5gs/upf.yaml . . . 575
3.98 /opt/GSM/osmo-operator/configs/open5gs/dump/open5gs/accounts.metadata.json . . . 577
3.99 /opt/GSM/osmo-operator/configs/open5gs/dump/open5gs/prelude.json . . . 577
3.100/opt/GSM/osmo-operator/configs/open5gs/dump/open5gs/sessions.metadata.json . . . 577
3.101/opt/GSM/osmo-operator/configs/open5gs/dump/open5gs/subscribers.metadata.json . . . 577
3.102/opt/GSM/osmo-operator/configs/open5gs/freeDiameter/hss.conf . . . 577
3.103/opt/GSM/osmo-operator/configs/open5gs/freeDiameter/mme.conf . . . 578
3.104/opt/GSM/osmo-operator/configs/open5gs/freeDiameter/pcrf.conf . . . 578
3.105/opt/GSM/osmo-operator/configs/open5gs/freeDiameter/smf.conf . . . 579
3.106/opt/GSM/osmo-operator/configs/plymouth/osmo-bts/osmo-bts.plymouth . . . 579
3.107/opt/GSM/osmo-operator/configs/plymouth/osmo-bts/osmo-bts.script . . . 579
3.108/opt/GSM/osmo-operator/configs/plymouth/vortex-ubuntu/SOURCE . . . 582
3.109/opt/GSM/osmo-operator/configs/plymouth/vortex-ubuntu/logo.svg . . . 582
3.110/opt/GSM/osmo-operator/configs/plymouth/vortex-ubuntu/vortex-ubuntu.grub . . . 584
3.111/opt/GSM/osmo-operator/configs/plymouth/vortex-ubuntu/vortex-ubuntu.plymouth . . . 584
3.112/opt/GSM/osmo-operator/configs/plymouth/vortex-ubuntu/vortex-ubuntu.script . . . 584
3.113/opt/GSM/osmo-operator/configs/pmos/pmbootstrap_v3.cfg . . . 587
3.114/opt/GSM/osmo-operator/configs/screenfetch/osmo-lab.sh . . . 588
3.115/opt/GSM/osmo-operator/configs/srsran/enb.conf . . . 589
3.116/opt/GSM/osmo-operator/configs/srsran/rb.conf . . . 596
3.117/opt/GSM/osmo-operator/configs/srsran/rr.conf . . . 599
3.118/opt/GSM/osmo-operator/configs/srsran/sib.conf . . . 601
3.119/opt/GSM/osmo-operator/configs/srsran/ue.conf . . . 606
3.120/opt/GSM/osmo-operator/data/c123-zones.json . . . 614
3.121/opt/GSM/osmo-operator/data/c139-zones.json . . . 615
3.122/opt/GSM/osmo-operator/data/claude.svg . . . 616
3.123/opt/GSM/osmo-operator/data/deka-toy.svg . . . 616
3.124/opt/GSM/osmo-operator/data/deka.svg . . . 617
3.125/opt/GSM/osmo-operator/data/osmo-dino.svg . . . 618
3.126/opt/GSM/osmo-operator/data/osmo-info.svg . . . 618
3.127/opt/GSM/osmo-operator/data/osmo-launch.svg . . . 618
3.128/opt/GSM/osmo-operator/data/osmo-lte.svg . . . 619
3.129/opt/GSM/osmo-operator/data/osmo-multi.svg . . . 619
3.130/opt/GSM/osmo-operator/data/osmo-paint.svg . . . 620

3.131/opt/GSM/osmo-operator/data/osmo-tutorial.svg . . . 620
3.132/opt/GSM/osmo-operator/data/tutorial.html . . . 621
3.133/opt/GSM/osmo-operator/data/desktop/claude.desktop . . . 623
3.134/opt/GSM/osmo-operator/data/desktop/osmo-addition.desktop . . . 623
3.135/opt/GSM/osmo-operator/data/desktop/osmo-dino.desktop . . . 624
3.136/opt/GSM/osmo-operator/data/desktop/osmo-drivers.desktop . . . 624
3.137/opt/GSM/osmo-operator/data/desktop/osmo-info.desktop . . . 624
3.138/opt/GSM/osmo-operator/data/desktop/osmo-install.desktop . . . 625
3.139/opt/GSM/osmo-operator/data/desktop/osmo-launch.desktop . . . 625
3.140/opt/GSM/osmo-operator/data/desktop/osmo-lte.desktop . . . 626
3.141/opt/GSM/osmo-operator/data/desktop/osmo-multi.desktop . . . 627
3.142/opt/GSM/osmo-operator/data/desktop/osmo-paint.desktop . . . 628
3.143/opt/GSM/osmo-operator/data/desktop/osmo-tutorial.desktop . . . 628
3.144/opt/GSM/osmo-operator/data/desktop/osmo-update.desktop . . . 628
3.145/opt/GSM/osmo-operator/environment/load.env . . . 629
3.146/opt/GSM/osmo-operator/environment/modes.env . . . 629
3.147/opt/GSM/osmo-operator/environment/paths.env . . . 630
3.148/opt/GSM/osmo-operator/fft-web/fft_web.py . . . 632
3.149/opt/GSM/osmo-operator/helpers/create_interop.sh . . . 636
3.150/opt/GSM/osmo-operator/helpers/prepare_host.sh . . . 639
3.151/opt/GSM/osmo-operator/install_modules/00-prereqs.sh . . . 641
3.152/opt/GSM/osmo-operator/install_modules/10-deps.sh . . . 641
3.153/opt/GSM/osmo-operator/install_modules/20-user.sh . . . 643
3.154/opt/GSM/osmo-operator/install_modules/30-sources.sh . . . 643
3.155/opt/GSM/osmo-operator/install_modules/40-patches.sh . . . 644
3.156/opt/GSM/osmo-operator/install_modules/50-build.sh . . . 645
3.157/opt/GSM/osmo-operator/install_modules/60-binaires.sh . . . 646
3.158/opt/GSM/osmo-operator/install_modules/70-configs.sh . . . 646
3.159/opt/GSM/osmo-operator/install_modules/80-bureau.sh . . . 647
3.160/opt/GSM/osmo-operator/install_modules/90-verify.sh . . . 650
3.161/opt/GSM/osmo-operator/install_modules/_lib/inst.sh . . . 650
3.162/opt/GSM/osmo-operator/installer/calamares/settings.conf . . . 651
3.163/opt/GSM/osmo-operator/installer/calamares/branding/osmo/branding.desc . . . 653
3.164/opt/GSM/osmo-operator/installer/calamares/branding/osmo/show.qml . . . 654
3.165/opt/GSM/osmo-operator/installer/calamares/jobs/osmosupplements/main.py . . . 657
3.166/opt/GSM/osmo-operator/installer/calamares/jobs/osmosupplements/module.desc . . . 658
3.167/opt/GSM/osmo-operator/installer/calamares/modules/bootloader.conf . . . 658
3.168/opt/GSM/osmo-operator/installer/calamares/modules/contextualprocess-mirror.conf . . . 658
3.169/opt/GSM/osmo-operator/installer/calamares/modules/contextualprocess-nvidia.conf . . . 659
3.170/opt/GSM/osmo-operator/installer/calamares/modules/displaymanager.conf . . . 659
3.171/opt/GSM/osmo-operator/installer/calamares/modules/finished.conf . . . 660
3.172/opt/GSM/osmo-operator/installer/calamares/modules/fstab.conf . . . 660
3.173/opt/GSM/osmo-operator/installer/calamares/modules/grubcfg.conf . . . 660
3.174/opt/GSM/osmo-operator/installer/calamares/modules/keyboard.conf . . . 661
3.175/opt/GSM/osmo-operator/installer/calamares/modules/locale.conf . . . 661
3.176/opt/GSM/osmo-operator/installer/calamares/modules/mount.conf . . . 661
3.177/opt/GSM/osmo-operator/installer/calamares/modules/packagechooser-image.conf . . . 662
3.178/opt/GSM/osmo-operator/installer/calamares/modules/packagechooser-mirror.conf . . . 662
3.179/opt/GSM/osmo-operator/installer/calamares/modules/packagechooser-nvidia.conf . . . 663
3.180/opt/GSM/osmo-operator/installer/calamares/modules/packagechooser-supplements.conf . . . 663
3.181/opt/GSM/osmo-operator/installer/calamares/modules/partition.conf . . . 664
3.182/opt/GSM/osmo-operator/installer/calamares/modules/shellprocess-grub.conf . . . 666
3.183/opt/GSM/osmo-operator/installer/calamares/modules/shellprocess-osmo.conf . . . 666
3.184/opt/GSM/osmo-operator/installer/calamares/modules/unpackfs.conf . . . 668
3.185/opt/GSM/osmo-operator/installer/calamares/modules/users.conf . . . 669
3.186/opt/GSM/osmo-operator/installer/calamares/modules/welcome.conf . . . 670
3.187/opt/GSM/osmo-operator/instances/instance.env.example . . . 670
3.188/opt/GSM/osmo-operator/iso_modules/00-options.sh . . . 673
3.189/opt/GSM/osmo-operator/iso_modules/10-clavier.sh . . . 679
3.190/opt/GSM/osmo-operator/iso_modules/20-hote.sh . . . 681
3.191/opt/GSM/osmo-operator/iso_modules/20-stockage-docker.sh . . . 684

3.192/opt/GSM/osmo-operator/iso_modules/21-docker-build.sh . . . 685
3.193/opt/GSM/osmo-operator/iso_modules/22-wan-table.sh . . . 690
3.194/opt/GSM/osmo-operator/iso_modules/30-image-configs.sh . . . 691
3.195/opt/GSM/osmo-operator/iso_modules/31-image-source.sh . . . 694
3.196/opt/GSM/osmo-operator/iso_modules/40-rootfs.sh . . . 695
3.197/opt/GSM/osmo-operator/iso_modules/50-injection-image.sh . . . 697
3.198/opt/GSM/osmo-operator/iso_modules/51-depot.sh . . . 698
3.199/opt/GSM/osmo-operator/iso_modules/51-softsim.sh . . . 700
3.200/opt/GSM/osmo-operator/iso_modules/52-qemu.sh . . . 701
3.201/opt/GSM/osmo-operator/iso_modules/60-dashboard.sh . . . 710
3.202/opt/GSM/osmo-operator/iso_modules/70-scripts.sh . . . 713
3.203/opt/GSM/osmo-operator/iso_modules/71-debs-banc.sh . . . 717
3.204/opt/GSM/osmo-operator/iso_modules/80-chroot.sh . . . 717
3.205/opt/GSM/osmo-operator/iso_modules/81-cloture-systeme.sh . . . 735
3.206/opt/GSM/osmo-operator/iso_modules/82-arm-natif.sh . . . 744
3.207/opt/GSM/osmo-operator/iso_modules/82-services.sh . . . 747
3.208/opt/GSM/osmo-operator/iso_modules/83-espace-ram.sh . . . 756
3.209/opt/GSM/osmo-operator/iso_modules/84-comptes.sh . . . 762
3.210/opt/GSM/osmo-operator/iso_modules/85-extensions-gnome.sh . . . 764
3.211/opt/GSM/osmo-operator/iso_modules/85-glava.sh . . . 765
3.212/opt/GSM/osmo-operator/iso_modules/85-installeur-bureau.sh . . . 765
3.213/opt/GSM/osmo-operator/iso_modules/86-finitions.sh . . . 781
3.214/opt/GSM/osmo-operator/iso_modules/87-lite.sh . . . 787
3.215/opt/GSM/osmo-operator/iso_modules/88-lte-pmos.sh . . . 788
3.216/opt/GSM/osmo-operator/iso_modules/89-home-chiffre.sh . . . 797
3.217/opt/GSM/osmo-operator/iso_modules/90-iso.sh . . . 800
3.218/opt/GSM/osmo-operator/iso_modules/91-secure-boot.sh . . . 806
3.219/opt/GSM/osmo-operator/iso_modules/92-rpi-image.sh . . . 810
3.220/opt/GSM/osmo-operator/iso_modules/99-fin.sh . . . 811
3.221/opt/GSM/osmo-operator/launch/osmo-launch.sh . . . 812
3.222/opt/GSM/osmo-operator/launch/start-docker.sh . . . 817
3.223/opt/GSM/osmo-operator/launch/start-interstp.sh . . . 862
3.224/opt/GSM/osmo-operator/launch/start-multi.sh . . . 869
3.225/opt/GSM/osmo-operator/launch/start-nitb.sh . . . 882
3.226/opt/GSM/osmo-operator/launch/start-oqc.sh . . . 883
3.227/opt/GSM/osmo-operator/lib/audio.sh . . . 884
3.228/opt/GSM/osmo-operator/lib/gabarits.sh . . . 896
3.229/opt/GSM/osmo-operator/navigation/**init**.py . . . 904
3.230/opt/GSM/osmo-operator/navigation/ber.py . . . 905
3.231/opt/GSM/osmo-operator/navigation/diagram.py . . . 907
3.232/opt/GSM/osmo-operator/navigation/m3ua.py . . . 911
3.233/opt/GSM/osmo-operator/navigation/mapops.py . . . 918
3.234/opt/GSM/osmo-operator/navigation/quickcmd.py . . . 922
3.235/opt/GSM/osmo-operator/navigation/responder.py . . . 924
3.236/opt/GSM/osmo-operator/navigation/sccp.py . . . 928
3.237/opt/GSM/osmo-operator/navigation/ss7-console.py . . . 931
3.238/opt/GSM/osmo-operator/navigation/ss7-diag.py . . . 935
3.239/opt/GSM/osmo-operator/navigation/ss7.py . . . 942
3.240/opt/GSM/osmo-operator/navigation/tcap.py . . . 945
3.241/opt/GSM/osmo-operator/navigation/test-console.sh . . . 948
3.242/opt/GSM/osmo-operator/navigation/topo.py . . . 951
3.243/opt/GSM/osmo-operator/navigation/tui.py . . . 958
3.244/opt/GSM/osmo-operator/navigation/vty.py . . . 972
3.245/opt/GSM/osmo-operator/network/apn-nat.sh . . . 976
3.246/opt/GSM/osmo-operator/network/firewall-wan.sh . . . 977
3.247/opt/GSM/osmo-operator/network/loopback.sh . . . 978
3.248/opt/GSM/osmo-operator/network/node-conf.sh . . . 980
3.249/opt/GSM/osmo-operator/network/osmo-ip-plan.sh . . . 982
3.250/opt/GSM/osmo-operator/network/pcu-socket.sh . . . 986
3.251/opt/GSM/osmo-operator/network/set-node-id.sh . . . 987
3.252/opt/GSM/osmo-operator/network/set_ip_vm.sh . . . 992

3.253/opt/GSM/osmo-operator/network/setup-docker-lan-route.sh . . . 999
3.254/opt/GSM/osmo-operator/network/setup-payload-ism.sh . . . 1002
3.255/opt/GSM/osmo-operator/network/setup-vbox-interco.sh . . . 1005
3.256/opt/GSM/osmo-operator/network/setup-wan-interop.sh . . . 1008
3.257/opt/GSM/osmo-operator/network/setup-wan-mesh.sh . . . 1017
3.258/opt/GSM/osmo-operator/network/setup-wan-sms.sh . . . 1030
3.259/opt/GSM/osmo-operator/network/wan-nodes.sh . . . 1031
3.260/opt/GSM/osmo-operator/opt-gsm/grgsm_fft_live.py . . . 1038
3.261/opt/GSM/osmo-operator/opt-gsm/record_drain.py . . . 1039
3.262/opt/GSM/osmo-operator/opt-gsm/si_bridge.py . . . 1041
3.263/opt/GSM/osmo-operator/opt-gsm/si_bridge_loop.sh . . . 1047
3.264/opt/GSM/osmo-operator/packaging/.gitignore . . . 1047
3.265/opt/GSM/osmo-operator/packaging/apt-fast-install.sh . . . 1047
3.266/opt/GSM/osmo-operator/packaging/apt-mirror.sh . . . 1050
3.267/opt/GSM/osmo-operator/packaging/build-debs.sh . . . 1052
3.268/opt/GSM/osmo-operator/packaging/build-pmos-kernel-deb.sh . . . 1056
3.269/opt/GSM/osmo-operator/packaging/osmo-deb.sh . . . 1058
3.270/opt/GSM/osmo-operator/packaging/snapshot-lte-debs.sh . . . 1061
3.271/opt/GSM/osmo-operator/patches/c54x_exe-apm-par-fenetre.patch . . . 1063
3.272/opt/GSM/osmo-operator/patches/c54x_exe-magnetite-run-can-nb-toa.patch . . . 1064
3.273/opt/GSM/osmo-operator/patches/c54x_exe-rssi-air-et-arret-instance.patch . . . 1072
3.274/opt/GSM/osmo-operator/patches/c54x_exe-sonde-depot-tch-irq.patch . . . 1074
3.275/opt/GSM/osmo-operator/patches/fc-magnetite-alr-arfcn-forced.patch . . . 1074
3.276/opt/GSM/osmo-operator/patches/fc-magnetite-banc.patch . . . 1078
3.277/opt/GSM/osmo-operator/patches/gapk-pq-alsa-output-probe.patch . . . 1083
3.278/opt/GSM/osmo-operator/patches/grgsm-optional-qtgui-import.patch . . . 1087
3.279/opt/GSM/osmo-operator/patches/grgsm-receiver-publish-bsic-fn.patch . . . 1087
3.280/opt/GSM/osmo-operator/patches/magnetite-parole-ul-au-go.patch . . . 1088
3.281/opt/GSM/osmo-operator/patches/magnetite-sms-msginfo-doublefree.patch . . . 1088
3.282/opt/GSM/osmo-operator/patches/osmo-bts-force-rand-toy.patch . . . 1090
3.283/opt/GSM/osmo-operator/patches/osmo-hlr-force-rand-toy.patch . . . 1092
3.284/opt/GSM/osmo-operator/patches/osmo-msc-force-rand-toy.patch . . . 1092
3.285/opt/GSM/osmo-operator/patches/osmo-trx-ipc-ts-frame-align.patch . . . 1093
3.286/opt/GSM/osmo-operator/patches/osmo-trx-rach-per-ra-table.patch . . . 1093
3.287/opt/GSM/osmo-operator/patches/osmocom-bb-clck-gen-frame-tolerance.patch . . . 1097
3.288/opt/GSM/osmo-operator/patches/osmocom-bb-fbsb-timeout-scan-to.patch . . . 1097
3.289/opt/GSM/osmo-operator/patches/osmocom-bb-firmware-gprs.patch . . . 1099
3.290/opt/GSM/osmo-operator/patches/osmocom-bb-firmware-map-ld.patch . . . 1109
3.291/opt/GSM/osmo-operator/patches/osmocom-bb-force-rand-toy.patch . . . 1109
3.292/opt/GSM/osmo-operator/patches/osmocom-bb-osmocon-kc-filter.patch . . . 1110
3.293/opt/GSM/osmo-operator/patches/osmocom-bb-plmn-a6-reinsert.patch . . . 1111
3.294/opt/GSM/osmo-operator/patches/osmocom-bb-sim-plmn-from-bcd.patch . . . 1111
3.295/opt/GSM/osmo-operator/patches/osmocom-bb-transceiver-dsp-error-ratelimit.patch . . . 1112
3.296/opt/GSM/osmo-operator/patches/osmocom-bb-transceiver-qemu-bench.patch . . . 1116
3.297/opt/GSM/osmo-operator/patches/pmaports-linux-postmarketos-stable-ppp.patch . . . 1129
3.298/opt/GSM/osmo-operator/patches/pmbootstrap-osmo-bench-qemu.patch . . . 1130
3.299/opt/GSM/osmo-operator/patches/qosmo-calypso-air-realiste.patch . . . 1135
3.300/opt/GSM/osmo-operator/patches/qosmo-calypso-bsp-magnetite-sb-fin-trame.patch . . . 1140
3.301/opt/GSM/osmo-operator/patches/qosmo-calypso-bsp-pas-de-routeur-nb-sur-tch.patch . . . 1148
3.302/opt/GSM/osmo-operator/patches/qosmo-calypso-bsp-pm-par-fenetre.patch . . . 1151
3.303/opt/GSM/osmo-operator/patches/qosmo-calypso-faux-air-arfcn.patch . . . 1155
3.304/opt/GSM/osmo-operator/patches/qosmo-calypso-sim-ef-table.patch . . . 1157
3.305/opt/GSM/osmo-operator/patches/qosmo-calypso-trx-tdma-dsp-ack.patch . . . 1164
3.306/opt/GSM/osmo-operator/patches/qosmo-gardien-bts0.patch . . . 1169
3.307/opt/GSM/osmo-operator/pont/**init**.py . . . 1170
3.308/opt/GSM/osmo-operator/pont/air.py . . . 1171
3.309/opt/GSM/osmo-operator/pont/airmesh.py . . . 1177
3.310/opt/GSM/osmo-operator/pont/cipher.py . . . 1186
3.311/opt/GSM/osmo-operator/pont/config.py . . . 1188
3.312/opt/GSM/osmo-operator/pont/downlink.py . . . 1189
3.313/opt/GSM/osmo-operator/pont/gsm.py . . . 1195

3.314/opt/GSM/osmo-operator/pont/pont.py . . . 1199
3.315/opt/GSM/osmo-operator/pont/pont_dsp.py . . . 1200
3.316/opt/GSM/osmo-operator/pont/record.py . . . 1200
3.317/opt/GSM/osmo-operator/pont/state.py . . . 1204
3.318/opt/GSM/osmo-operator/pont/stats.py . . . 1206
3.319/opt/GSM/osmo-operator/pont/trx.py . . . 1208
3.320/opt/GSM/osmo-operator/pont/uplink.py . . . 1216
3.321/opt/GSM/osmo-operator/pont/voix_hote.py . . . 1222
3.322/opt/GSM/osmo-operator/pont/vsp_audio.py . . . 1224
3.323/opt/GSM/osmo-operator/pont/dsp/**init**.py . . . 1227
3.324/opt/GSM/osmo-operator/pont/dsp/clock.py . . . 1229
3.325/opt/GSM/osmo-operator/pont/dsp/dedicated.py . . . 1231
3.326/opt/GSM/osmo-operator/pont/dsp/downlink.py . . . 1232
3.327/opt/GSM/osmo-operator/pont/dsp/tch.py . . . 1233
3.328/opt/GSM/osmo-operator/pont/dsp/trx.py . . . 1234
3.329/opt/GSM/osmo-operator/pont/dsp/uplink.py . . . 1234
3.330/opt/GSM/osmo-operator/scripts/audio-chain.sh . . . 1241
3.331/opt/GSM/osmo-operator/scripts/audio-diag.sh . . . 1243
3.332/opt/GSM/osmo-operator/scripts/banc-repro.sh . . . 1246
3.333/opt/GSM/osmo-operator/scripts/call-diag.sh . . . 1249
3.334/opt/GSM/osmo-operator/scripts/calypso-mode.sh . . . 1252
3.335/opt/GSM/osmo-operator/scripts/check.sh . . . 1254
3.336/opt/GSM/osmo-operator/scripts/dsp-rom-max.sh . . . 1255
3.337/opt/GSM/osmo-operator/scripts/egprs-dsp.sh . . . 1259
3.338/opt/GSM/osmo-operator/scripts/entrypoint.sh . . . 1265
3.339/opt/GSM/osmo-operator/scripts/gapk-start.sh . . . 1266
3.340/opt/GSM/osmo-operator/scripts/ism_payload.py . . . 1278
3.341/opt/GSM/osmo-operator/scripts/l1ctl_bridge.py . . . 1284
3.342/opt/GSM/osmo-operator/scripts/magnetite-c139.sh . . . 1289
3.343/opt/GSM/osmo-operator/scripts/osmo-banc-aliases.sh . . . 1290
3.344/opt/GSM/osmo-operator/scripts/osmo-disclaimer.sh . . . 1291
3.345/opt/GSM/osmo-operator/scripts/osmo-gnuarm-profile.sh . . . 1292
3.346/opt/GSM/osmo-operator/scripts/osmo-start.sh . . . 1292
3.347/opt/GSM/osmo-operator/scripts/osmo-wg-profile.sh . . . 1294
3.348/opt/GSM/osmo-operator/scripts/pulse-gsm-setup.sh . . . 1295
3.349/opt/GSM/osmo-operator/scripts/run.sh . . . 1297
3.350/opt/GSM/osmo-operator/scripts/send-mt-sms.sh . . . 1305
3.351/opt/GSM/osmo-operator/scripts/sms-interop-relay.py . . . 1306
3.352/opt/GSM/osmo-operator/scripts/sms-routing-setup.sh . . . 1320
3.353/opt/GSM/osmo-operator/scripts/sms_notify.py . . . 1329
3.354/opt/GSM/osmo-operator/scripts/smsc-start.sh . . . 1330
3.355/opt/GSM/osmo-operator/scripts/status.sh . . . 1333
3.356/opt/GSM/osmo-operator/scripts/voix-forks.sh . . . 1333
3.357/opt/GSM/osmo-operator/scripts/wslg-audio-bridge.sh . . . 1334
3.358/opt/GSM/osmo-operator/services/80-osmo-operator.preset . . . 1339
3.359/opt/GSM/osmo-operator/services/open5gs-webui.service . . . 1340
3.360/opt/GSM/osmo-operator/services/osmo-banc.service . . . 1341
3.361/opt/GSM/osmo-operator/services/osmo-bts-trx.service . . . 1343
3.362/opt/GSM/osmo-operator/services/osmo-dsp-rom-max.service . . . 1343
3.363/opt/GSM/osmo-operator/services/osmo-egprs-dsp.service . . . 1344
3.364/opt/GSM/osmo-operator/services/osmo-egprs-web-install.service . . . 1344
3.365/opt/GSM/osmo-operator/services/osmo-egprs-web.service . . . 1345
3.366/opt/GSM/osmo-operator/services/osmo-lte.service . . . 1347
3.367/opt/GSM/osmo-operator/services/osmo-multi.service . . . 1347
3.368/opt/GSM/osmo-operator/services/osmo-reglages.service . . . 1349
3.369/opt/GSM/osmo-operator/services/osmo-supplements.service . . . 1349
3.370/opt/GSM/osmo-operator/startup_scripts/01-defauts-options.sh . . . 1349
3.371/opt/GSM/osmo-operator/startup_scripts/02-noeud-couleurs.sh . . . 1358
3.372/opt/GSM/osmo-operator/startup_scripts/03-affichage.sh . . . 1359
3.373/opt/GSM/osmo-operator/startup_scripts/04-environnement.sh . . . 1360
3.374/opt/GSM/osmo-operator/startup_scripts/05-run-sh.sh . . . 1362

3.375/opt/GSM/osmo-operator/startup_scripts/06-banc-dsp-menu.sh . . . 1363
3.376/opt/GSM/osmo-operator/startup_scripts/07-plan-mobile.sh . . . 1374
3.377/opt/GSM/osmo-operator/startup_scripts/08-mobile-ms1.sh . . . 1374
3.378/opt/GSM/osmo-operator/startup_scripts/09-mobile-cfg-tmux.sh . . . 1374
3.379/opt/GSM/osmo-operator/startup_scripts/10-identite-noeud.sh . . . 1380
3.380/opt/GSM/osmo-operator/startup_scripts/11-regen-configs.sh . . . 1384
3.381/opt/GSM/osmo-operator/startup_scripts/12-pcap-softsim.sh . . . 1388
3.382/opt/GSM/osmo-operator/startup_scripts/13-adresses.sh . . . 1397
3.383/opt/GSM/osmo-operator/startup_scripts/14-demarrage.sh . . . 1403
3.384/opt/GSM/osmo-operator/startup_scripts/15-fin.sh . . . 1405
3.385/opt/GSM/osmo-operator/tools/at-cmd.py . . . 1406
3.386/opt/GSM/osmo-operator/tools/conky-osmo-status.sh . . . 1408
3.387/opt/GSM/osmo-operator/tools/daram_sweep.sh . . . 1416
3.388/opt/GSM/osmo-operator/tools/deaccent.pl . . . 1418
3.389/opt/GSM/osmo-operator/tools/dockerfile-to-install-md.sh . . . 1418
3.390/opt/GSM/osmo-operator/tools/fft.sh . . . 1423
3.391/opt/GSM/osmo-operator/tools/fft2.sh . . . 1426
3.392/opt/GSM/osmo-operator/tools/fft_global.sh . . . 1427
3.393/opt/GSM/osmo-operator/tools/glava-build.sh . . . 1429
3.394/opt/GSM/osmo-operator/tools/globals-complet.py . . . 1431
3.395/opt/GSM/osmo-operator/tools/gnome-extensions-install.sh . . . 1434
3.396/opt/GSM/osmo-operator/tools/grgsm_relay_decode.py . . . 1435
3.397/opt/GSM/osmo-operator/tools/grub-theme-render.py . . . 1436
3.398/opt/GSM/osmo-operator/tools/make-docker-image.sh . . . 1438
3.399/opt/GSM/osmo-operator/tools/matrix.sh . . . 1439
3.400/opt/GSM/osmo-operator/tools/osmo-banc-cmd.sh . . . 1441
3.401/opt/GSM/osmo-operator/tools/osmo-c123-gtk.py . . . 1441
3.402/opt/GSM/osmo-operator/tools/osmo-dino-ciel.py . . . 1451
3.403/opt/GSM/osmo-operator/tools/osmo-dino.py . . . 1459
3.404/opt/GSM/osmo-operator/tools/osmo-drivers.sh . . . 1462
3.405/opt/GSM/osmo-operator/tools/osmo-epc.sh . . . 1464
3.406/opt/GSM/osmo-operator/tools/osmo-extras-install.sh . . . 1468
3.407/opt/GSM/osmo-operator/tools/osmo-fft-snap.py . . . 1475
3.408/opt/GSM/osmo-operator/tools/osmo-firmwares-install.sh . . . 1492
3.409/opt/GSM/osmo-operator/tools/osmo-fond-antenne.py . . . 1498
3.410/opt/GSM/osmo-operator/tools/osmo-fond-bleu.py . . . 1499
3.411/opt/GSM/osmo-operator/tools/osmo-fond-repos.py . . . 1502
3.412/opt/GSM/osmo-operator/tools/osmo-fond.py . . . 1504
3.413/opt/GSM/osmo-operator/tools/osmo-gcc-alternatives.sh . . . 1508
3.414/opt/GSM/osmo-operator/tools/osmo-glava.sh . . . 1508
3.415/opt/GSM/osmo-operator/tools/osmo-gtk-run.py . . . 1510
3.416/opt/GSM/osmo-operator/tools/osmo-info . . . 1518
3.417/opt/GSM/osmo-operator/tools/osmo-launcher.py . . . 1519
3.418/opt/GSM/osmo-operator/tools/osmo-lte-install.sh . . . 1528
3.419/opt/GSM/osmo-operator/tools/osmo-lte.sh . . . 1537
3.420/opt/GSM/osmo-operator/tools/osmo-moon.py . . . 1543
3.421/opt/GSM/osmo-operator/tools/osmo-ofono-bridge.py . . . 1546
3.422/opt/GSM/osmo-operator/tools/osmo-op.sh . . . 1550
3.423/opt/GSM/osmo-operator/tools/osmo-panel.py . . . 1554
3.424/opt/GSM/osmo-operator/tools/osmo-phonesim-banc.py . . . 1569
3.425/opt/GSM/osmo-operator/tools/osmo-pmos-build.sh . . . 1613
3.426/opt/GSM/osmo-operator/tools/osmo-pmos-install.sh . . . 1617
3.427/opt/GSM/osmo-operator/tools/osmo-pmos-qemu.sh . . . 1622
3.428/opt/GSM/osmo-operator/tools/osmo-pmos-setup.sh . . . 1629
3.429/opt/GSM/osmo-operator/tools/osmo-pmos.sh . . . 1636
3.430/opt/GSM/osmo-operator/tools/osmo-reglages.py . . . 1643
3.431/opt/GSM/osmo-operator/tools/osmo-ril-atmodem.py . . . 1651
3.432/opt/GSM/osmo-operator/tools/osmo-sim-install.sh . . . 1658
3.433/opt/GSM/osmo-operator/tools/osmo-sim.sh . . . 1661
3.434/opt/GSM/osmo-operator/tools/osmo-supplements-firstboot.sh . . . 1667
3.435/opt/GSM/osmo-operator/tools/osmo-telecom-boot.sh . . . 1668

3.436/opt/GSM/osmo-operator/tools/osmo-topzone.py . . . 1669
3.437/opt/GSM/osmo-operator/tools/osmo-ts-probe.py . . . 1675
3.438/opt/GSM/osmo-operator/tools/osmo-wallpaper.sh . . . 1682
3.439/opt/GSM/osmo-operator/tools/osmo-wg.sh . . . 1687
3.440/opt/GSM/osmo-operator/tools/osmo-wireshark-root.sh . . . 1690
3.441/opt/GSM/osmo-operator/tools/overlay-draw.py . . . 1692
3.442/opt/GSM/osmo-operator/tools/paysage-render.py . . . 1701
3.443/opt/GSM/osmo-operator/tools/plymouth-render.py . . . 1708
3.444/opt/GSM/osmo-operator/tools/qemu_bcch_grgsm.py . . . 1711
3.445/opt/GSM/osmo-operator/tools/screenfetch-art.py . . . 1714
3.446/opt/GSM/osmo-operator/tools/si.py . . . 1717
3.447/opt/GSM/osmo-operator/tools/status.sh . . . 1718
3.448/opt/GSM/osmo-operator/tools/vbox-wan-lab.sh . . . 1718
3.449/opt/GSM/osmo-operator/tools/vty-connect.exp . . . 1728
3.450/opt/GSM/osmo-operator/tools/vty-menu.sh . . . 1728
3.451/opt/GSM/osmo-operator/tools/wallpaper-render.py . . . 1731
3.452/opt/GSM/osmo-operator/tools/softsim/0001-simos-comp128v1-a38.patch . . . 1739
3.453/opt/GSM/osmo-operator/tools/softsim/0002-sap-serve-next-client.patch . . . 1740
3.454/opt/GSM/osmo-operator/tools/softsim/0003-sap-log-timestamps.patch . . . 1741
3.455/opt/GSM/osmo-operator/tools/softsim/0004-pcsc-disconnect-idempotent.patch . . . 1741
3.456/opt/GSM/osmo-operator/tools/softsim/comp128.rb . . . 1742
3.457/opt/GSM/osmo-operator/tools/softsim/make_sim_op1.py . . . 1744
3.458/opt/GSM/osmo-operator/tools/softsim/ruby_compat.rb . . . 1746
3.459/opt/GSM/osmo-operator/web/app.js . . . 1746
3.460/opt/GSM/osmo-operator/web/index.html . . . 1751
3.461/opt/GSM/osmo-operator/web/style.css . . . 1753
3.462/opt/GSM/osmo-operator/wiki/installer-osmo-jusquau-dsp.ipynb . . . 1754
3.463/opt/GSM/osmo-operator/wiki/lancer-les-modes.ipynb . . . 1773
3.464/opt/GSM/osmo-operator/wiki/lancer-pas-a-pas.ipynb . . . 1778

1 Synthèse

run	élément	valeur
(pas de verdict.txt)		

dossier de run	contenu
Verdict et couverture	échelle des barreaux et tableau de couverture couche 1
Résultats des tests	captures VTY, diagnostics, rapports pytest, grafcets (.txt .md .mmd)
Fichiers	sources, scripts, configs et docs du banc (tout fichier texte)
Logs	journaux (.log), en dernier

Méthode

Dossiers : /opt/GSM/osmo-operator. Extensions : tous les fichiers texte (hors .git, caches, binaires, sauvegardes, LICENSE et sorties précédentes). Entrée 5488 Ko, sortie 5323 Ko.
Compactage sans perte : fichiers identiques cités une fois ; lignes consécutives identiques « ×N » ; lignes ne différant que par des nombres : gabarit avec ⟨k⟩ puis la liste ordonnée des valeurs (k-uplets), chaque ligne se reconstruit en remettant les valeurs dans l'ordre. Seuls les codes couleur, la négociation telnet des captures VTY et les caractères de commande sont retirés ; les lignes de plus de 1000 caractères sont coupées en morceaux terminés par ↩. Les .md sont inclus tels quels (titres rétrogradés) et les .mmd rendus en diagrammes Mermaid, sans compactage.

2 Resultats des tests

2.1 /opt/GSM/osmo-operator/README.md

9232 octets, 130 lignes

2.1.1 osmo-operator

A complete **GSM + LTE operator on one machine**: an Osmocom core network (BTS, BSC, MSC, HLR, MGW, SMSC, Asterisk) and **emulated Calypso phones**. The ARM runs under QEMU (qosmo) and the TI C54x DSP runs outside it, executing the real mask ROM (c54x_exe). The phone is either the OsmocomBB `layer1` + `mobile`, or a Motorola C139 running the **original TI firmware** (FreeCalypso Magnetite). It camps, authenticates, ciphers, sends SMS and holds voice calls. No SDR and no physical phone: the radio is emulated end to end, down to the baseband.



Figure 1: The bench in action: CSFB call, I/Q spectra, srsUE, Wireshark, Linphone

2.1.1.1 Start here

I want to…	Read
Boot a ready-made live image (fastest, nothing to build)	iso · latest release
Pick a mode and run it	Modes below · the **?** of the dock (`osmo-info`) · lancer-les-modes.ipynb
Install on Ubuntu 24.04 without Docker	native · installer-osmo-jusquau-dsp.ipynb
Run in Docker, or build the images	docker
Run on a Raspberry Pi 4 / arm64	arm
Study the real C54x DSP on the mask ROM	dsp
Run the original TI firmware (C139, Magnetite)	magnetite
Use the gr-gsm layer 1	grgsm
Add 4G, CSFB and the postmarketOS phone	lte
Several operators, SS7 interconnect, a WAN between machines	ss7 · WAN
Drive the bench day to day: tmux, VTY, ports, Wireshark	usage
Check that the bench is healthy, reproduce a fault	tests
See how the pieces fit	architecture · all documents

2.1.1.2 Modes

Everything goes through `./start.sh` (`start-native.sh` is the same script, kept as a link). `--regen` rebuilds `/etc/osmocom` from the templates, `--no-attach` stays out of tmux, `./start.sh --help` lists every option.

Command	What runs	State
`./start.sh --bts-classique --ms`	**Default** (red phone icon). `osmo-bts-trx` ↔ `pont/pont_dsp.py`; the phone runs osmocom-bb `layer1` + `mobile` on the DSP ROM	LU, SMS, calls
`./start.sh --bts-classique --ms-magnetite`	Same network, the phone is a C139 running the TI firmware; `--gapk` for voice on the host	LU, SMS; calls in A5/0 (`ENCRYPTION="a5 0" ./start.sh --regen …`)

Command	What runs	State
`./start.sh --bts --ms --ts 2`	The Calypso of QEMU **is the BTS radio** (`trx` firmware + `transceiver`); a second Calypso camps on it over the air; one Calypso per timeslot	camps, LU; no call yet
`./start.sh --bts --ms --ts 2/1`	TS0 + TS1 on one Calypso TRX (default of `--bts`)	calls
`./start.sh --grgsm`	gr-gsm layer 1 instead of the DSP	end to end
`banc-rssi`	The osmocom-bb `rssi` app on a 4th Calypso, fed by the pont; screen `c123 --instance 4`	DL levels, sync, BCCH
`./start.sh --stop`	Stop the whole stack	

Phone screens and keypads: `c123 --instance N` (1 = phone of `--bts-classique` or TRX of `--bts`, 2 = phone over the air, 3 = second TRX, 4 = rssi). Nothing telecom starts at boot (`services/80-osmo-operator.preset`): starting a network is your gesture.

Quickest path from the ISO (password `osmo`):

```
cat osmo-operator-desktop.iso.part-* > osmo-operator-desktop.iso
sha256sum -c --ignore-missing SHA256SUMS
qemu-system-x86_64 -cdrom osmo-operator-desktop.iso -m 8G -enable-kvm -cpu host -smp 4
```

From a clone (native, or inside the container):

```
./start.sh --regen --bts-classique --ms   # first run: generate the configs, start, attach to tmux
./start.sh --stop
```

2.1.1.3 Notebooks

Notebook	What it does
installer-osmo-jusquau-dsp.ipynb (nbviewer)	Fresh Ubuntu 24.04 to a call on the DSP, one cell per step with checks; optional Magnetite build
lancer-les-modes.ipynb (nbviewer)	Each mode above: start, check, stop; the phone screens inline, keys, `banc-rssi`
lancer-pas-a-pas.ipynb (nbviewer)	The bench process by process: HLR, MGW, MSC, BSC, BTS, QEMU, osmocon, mobile, pont

How to run them (disposable container, VM, unattended `nbconvert`): README-notebook.

2.1.1.4 Supplements

What is not in the ISO installs with `sudo ./addition.sh` (a checklist window, or flags):

Flag	Installs
`--multi`	Docker, the operator image, the SS7 multi-operator topology
`--opencl` · `--claude` · `--extras` · `--sim`	GPU OpenCL and A5 tools · Claude Code · games and media · PC/SC SIM tools
`--gnuarm`	the historic OsmocomBB toolchain, arm-elf-gcc 4.5.2 (`/root/gnuarm`)
`--magnetite`	Wine + TI TMS470, FreeCalypso Magnetite for the C139, the bench patches, the flash image
`--trx`	osmocom-bb-trx: the Docker image and the `trx` firmware (TX) built with gnuarm

Flag	Installs
`--calypso-fw`	the bench's Calypso firmware (`layer1`, `rssi`) rebuilt with gnuarm (prebuilt in firmware)
`--fernvale`	MediaTek MT6260: fernly, fernvale-nuttx, osmocom-bb-mtk-layer1
`--simcom`	the SIMCom documentation and SDKs
`--firmwares`	all the firmware supplements above

2.1.1.5 Wiki

Page	Contents
Home	quick start, launching, reference card
Start-native	the launcher step by step, option manual
Reference	architecture, layer-1 chains, build graph, multi-PLMN SS7, diagnostics
Build · ISO	`build.sh`, `build-iso.sh`, `.deb`, `install.sh`; what the ISO contains
Environnement	which variables win, what reaches QEMU
Resultats	reference `check_all.sh` runs and how to read the deviations
Telephone-emule	plug **only** the emulated phone into your own Osmocom stack
Lancer-a-la-main	the phone started by hand, with proof at each step
Installer-qosmo · Installer-c54x_exe · Installer-grgsm_exe	building the QEMU fork and the two external layer 1s
SIM	a sysmoISIM in an Omnikey reader, or a softSIM, over SAP
WAN · Payload-ISM	linking nodes; SMS text over an ISM radio link

2.1.1.6 Other documents

Document	Subject
`pont/README.md`	A5 ciphering, Kc and SACCH: a measurement report
`doc/magnetite.md`	the TI firmware in qosmo: build, flash image, debugging
`environment/README.md`	environment variables and path resolution
`services/README.md`	the systemd units shipped, and the boot policy
`navigation/QUICKSTART.md`	the SS7 console
Components and their READMEs	every Osmocom module and third-party component, with the local patches

2.1.1.7 Repositories and sites

qosmo · c54x_exe · grgsm_exe	the QEMU fork and the two external layer 1s
firmware · osmocom-bb-trx	prebuilt Calypso firmware; osmocom-bb jolly/testing with TX
osmo-egprs-web	the dashboard (http://127.0.0.1:8080)
bbaranoff.github.io · pl4y.store · github.com/bbaranoff	the author's resources and projects
Upstream	Osmocom · FreeCalypso · Open5GS · srsRAN

---

osmo-operator is a teaching and research GSM bench: a full core network and emulated Calypso phones on one machine, down to the DSP mask ROM, with multi-operator SS7 interconnect when you want it.

2.2 /opt/GSM/osmo-operator/.github/release-notes.md

6068 octets, 101 lignes

Figure 2: Build ISO

2.2.0.1 osmo-operator-desktop.iso · v0.1-9

Live ISO password: **`osmo`** · postmarketOS phone password: **`147147`**

A complete GSM + LTE operator on one live image, with a GNOME desktop. No SDR and no physical phone: the radio is emulated end to end, down to the baseband. The Calypso's ARM runs under QEMU (qosmo), its TI C54x DSP runs the real mask ROM (c54x_exe).

2.2.0.1.1 Download

Download every `.part-NN` file, not just the first one (GitHub caps release files at 2 GiB). Then:

```
cat osmo-operator-desktop.iso.part-* > osmo-operator-desktop.iso
sha256sum -c --ignore-missing SHA256SUMS
```



Figure 3: osmo-operator: Open Source, No RF

2.2.0.1.2 New in 0.1-9

Phones and radio - **Classic BTS is the default**: `./start.sh --bts-classique --ms`. `osmo-bts-trx` talks to `pont/pont_dsp.py`; the emulated phone runs osmocom-bb `layer1` + `mobile` on the DSP mask ROM. The red phone icon starts exactly this. - **The original TI firmware**: `--ms-magnetite` boots a Motorola C139 running FreeCalypso Magnetite (TI L2/L3 and UI, colour screen and keypad in `c123`). It camps, registers, sends and receives SMS; calls work in A5/0, voice on the host with `--gapk`. Build: `sudo ./addition.sh --magnetite` (doc/magnetite.md). - **The Calypso as the BTS radio**: `--bts` runs the `trx` firmware in QEMU; `--ms` adds a second Calypso that camps on it over the air; `--ts 1 | 2 | 2/1` chooses the timeslots. - **`banc-rssi`**: the osmocom-bb `rssi` app on a fourth Calypso, fed by the pont in classic BTS mode, with a realistic air: the cell only on its ARFCN, -60 dBm downlink, noise elsewhere, AGC, BCCH decoded. - **Voice**: `--vocoder-rom` (the FR codec of the DSP ROM, default) or `--gapk` (host vocoder, keeps real time). - C123 and C139 screens and keypads in QEMU: `c123 --instance N`.

Robustness - BTS#0 runs under a watchdog: an `osmo-bts-trx` exit on "PC clock skew too high" restarts it after 3 s. - The pont caps the pause it asks of the BTS (`PONT_CLCK_PAUSE_PLAFOND`), the root cause of those exits. - **Nothing telecom starts at boot** (systemd preset `80-osmo-operator.preset`): the bench, the Osmocom core, the BTS, Asterisk, oFono, Open5GS and srsRAN wait for you. Web consoles, sound and network addressing still start.

Desktop and docs - The **?** at the top of the dock opens the help page (modes, modules, notebooks, repositories, references), served by the dashboard so that the snap Firefox can open it. - A new Plymouth boot screen: flat 2D, an antenna and its circles. - Settings page for the whole stack: http://127.0.0.1:8090 (`globals.conf`, Osmocom configs, Calypso instances). - Notebooks: install, run each mode, process by process. - The launcher is now `start.sh` (`start-native.sh` stays as a link).

Supplements (`sudo ./addition.sh`, not in the ISO) - `--gnuarm` (arm-elf-gcc 4.5.2), `--magnetite` (Wine + TI TMS470), `--trx` (osmocom-bb-trx image + TX firmware), `--calypso-fw` (the bench firmware rebuilt with gnuarm), `--fernvale` (MediaTek MT6260: fernly, NuttX, MTK layer 1), `--simcom` (SIMCom docs and SDKs), `--firmwares` (all of them).

Known limits - Magnetite: calls need A5/0 (with A5/1 the FACCH is decoded too early). - `--bts --ts 2`: camps and registers, no call yet (the second TRX is not frame-aligned); `--ts 2/1` calls. - `banc-rssi`: the uplink per-timeslot view does not show the phone yet. - The ROM vocoder costs real time on slow hosts: use `--gapk`. Do not compile while a bench runs.

2.2.0.1.3 What's inside

- **2G (Osmocom)**: osmo-stp, osmo-bsc, osmo-msc, osmo-hlr, osmo-sgsn, osmo-ggsn, osmo-pcu, osmo-mgw; two cells in LAC 1 (cell 6001 BSIC 7, cell 6012 BSIC 8); GPRS/EDGE; voice through osmo-sip-connector and Asterisk; SMS over GSUP and SMPP; deterministic RAND for reproducible runs; live GSMTAP for Wireshark.
- **The DSP**: a TMS320C54x emulated outside QEMU on the dumped mask ROM, sharing the API RAM with the ARM and stepping one TDMA frame at a time.
- **4G (Open5GS + srsRAN)** over ZMQ, PLMN 001/01, null NAS ciphering for analysis; **CSFB** through SGs between the MME and osmo-msc.
- **postmarketOS phone**: data on 4G, voice and SMS on 2G under one identity, oFono on a virtual AT modem.

2.2.0.1.4 Running the image

x86_64, **8 GB of RAM and 4 cores** minimum; in a VM, enable nested virtualization.

```
qemu-system-x86_64 -cdrom osmo-operator-desktop.iso -m 8G -enable-kvm \
  -cpu host -smp 4 -nic user,hostfwd=tcp::8080-:8080
```

VirtualBox or VMware: Linux / Ubuntu 64-bit, 8 GB, 4 CPUs, nested VT-x/AMD-V ticked, the ISO in the optical drive. USB stick: `sudo dd if=osmo-operator-desktop.iso of=/dev/sdX bs=4M status=progress conv=fsync` (Rufus: DD mode).

2.2.0.1.5 Getting started

Dock	Starts
?	this help: every mode, module, notebook and repository
red phone	the 2G bench, `./start.sh --bts-classique --ms`
signal bars	4G: Open5GS, srsENB, srsUE and SGs
phone	the postmarketOS VM (start it last)

Other modes from a terminal: `cd /opt/GSM/osmo-operator && sudo ./start.sh --help`. Health check: `checks/diag-stp-operator.sh`. Documentation: README · wiki · bbaranoff.github.io · pl4y.store.

2.3 /opt/GSM/osmo-operator/configs/srsran/user_db.csv

2000 octets, 21 lignes → 21 lignes

```
#
# .csv to store UE's information in HSS
# Kept in the following format: "Name,Auth,IMSI,Key,OP_Type,OP/OPc,AMF,SQN,QCI,IP_alloc"
#
# Name:     Human readable name to help distinguish UE's. Ignored by the HSS
# Auth:     Authentication algorithm used by the UE. Valid algorithms are XOR
#           (xor) and MILENAGE (mil)
# IMSI:     UE's IMSI value
# Key:      UE's key, where other keys are derived from. Stored in hexadecimal
# OP_Type:  Operator's code type, either OP or OPc
# OP/OPc:   Operator Code/Cyphered Operator Code, stored in hexadecimal
# AMF:      Authentication management field, stored in hexadecimal
# SQN:      UE's Sequence number for freshness of the authentication
# QCI:      QoS Class Identifier for the UE's default bearer.
# IP_alloc: IP allocation stratagy for the SPGW.
#           With 'dynamic' the SPGW will automatically allocate IPs
#           With a valid IPv4 (e.g. '172.16.0.2') the UE will have a statically assigned IP.
#
# Note: Lines starting by '#' are ignored and will be overwritten
ue2,mil,001010001000001,00112233445566778899aabbccdd0101,opc,63bfa50ee6523365ff14c1f45f88737d,8000,576,9,dynamic
⌋
↪ ue1,xor,001010123456789,00112233445566778899aabbccddeeff,opc,63bfa50ee6523365ff14c1f45f88737d,9001,000000001234,7,dynamic
```

2.4 /opt/GSM/osmo-operator/data/adresses.txt

818 octets, 20 lignes → 20 lignes

```
# adresses.txt - "rue, ville" par ligne. start-docker.sh prefixe un numero tire au
# hasard. Rien de tout cela n'existe non plus.
rue des Ondes Courtes, Fontaine-sur-Onde
impasse du Burst, Val-de-Trame
allee Erlang, Bourg-la-Cellule
boulevard de la Trame, Saint-Handover
quai des Timeslots, Neuville-les-Relais
place du Handover, Clairmont-du-Spectre
chemin des Bursts Perdus, Rocheporte
avenue du Codec, Aubain-sur-Bande
sentier du Roaming, Montcodec
cour des Antennes, Fontaine-sur-Onde
passage du Silence Radio, Val-de-Trame
rue du Ki Perdu, Bourg-la-Cellule
venelle des Slots Vides, Saint-Handover
rond-point du Paging, Neuville-les-Relais
descente de la Bande Basse, Clairmont-du-Spectre
montee du Rejet d'Authentification, Rocheporte
square des Abonnes Fantomes, Aubain-sur-Bande
esplanade du Timeout, Montcodec
```

2.5 /opt/GSM/osmo-operator/data/air-mesh.txt

1724 octets, 34 lignes → 34 lignes

```
# air-mesh.txt - QUI ENTEND QUI, sur le maillage de bursts.
#
# Une arete par ligne :
#     <noeud a> <noeud b> <attenuation dB> [uni]
#
# Sans le mot "uni" a la fin, l'arete vaut DANS LES DEUX SENS. Avec, elle ne
# vaut que de a vers b - utile pour un noeud qui emet fort et recoit mal, ou
# pour forcer un handover a sens unique.
#
# UNE PAIRE ABSENTE DE CE FICHIER NE S'ENTEND PAS. C'est ce qui fait un
# maillage et non un bus : sans graphe, chaque mobile entendrait toutes les
# BTS du banc a pleine puissance, il n'y aurait plus de cellule, plus de
# reselection, plus rien a demontrer.
#
# ORDRE DE GRANDEUR. Le RSSI annonce par la BTS tourne autour de -60 dBm ; le
# plancher d'airmesh est a -110 dBm par defaut. Donc :
#     10-25 dB   deux cellules voisines, les deux lisibles - reselection
#     30-45 dB   cellule lointaine, lisible mais moins bonne que la voisine
#     50 dB et + a la limite : de quoi voir un mobile hesiter puis decrocher
#     au-dela de 50, ou pas de ligne du tout : hors de portee
#
# Ce fichier n'est PAS relu a chaud : airmesh le charge au demarrage. Le
# modifier demande de relancer airmesh, pas toute la pile.

# ── Le banc courant : trois noeuds ──────────────────────────────────────
# 1 et 2 sont deux conteneurs sur la meme machine, donc deux cellules
# adjacentes. 3 est la VM : plus loin, lisible depuis 2 seulement.
1 2 20
2 3 35

# 1 et 3 ne s'entendent pas : la ligne n'existe pas, et c'est delibere.
# Un mobile du noeud 1 qui veut joindre le noeud 3 doit passer par la cellule
# du noeud 2 - c'est le seul moyen d'avoir un vrai chemin a deux sauts sur un
# banc de trois machines.
```

2.6 /opt/GSM/osmo-operator/data/c139.txt

297 octets, 3 lignes → 3 lignes

```
data/c139.jpg : photo « Motorola C139-1170.jpg » de Raimond Spekking (travail personnel), Wikimedia Commons,
licence CC BY-SA 4.0 (https://commons.wikimedia.org/wiki/File:Motorola_C139-1170.jpg), recadree sur le telephone
et reduite a 720x1561 le 21/10/2026 pour osmo-c123-gtk.py --modele c139.
```

2.7 /opt/GSM/osmo-operator/data/noms.txt

1105 octets, 54 lignes → 54 lignes

```
# noms.txt - etat civil des abonnes du banc. Un "Prenom Nom" par ligne.
# Tout est faux : ces gens n'existent pas, ils servent a mettre un visage sur
# un IMSI dans un journal. start-docker.sh en pioche un au hasard par abonne.
# Lignes vides et lignes commencant par # ignorees.
Aline Boulanger
Bastien Carpentier
Chloe Delaunay
Damien Estevez
Elodie Fontaine
Farid Grandjean
Gwenael Hautier
Hind Ibanez
Ismael Jouvenel
Jocelyne Kerbrat
Karim Lachance
Lucie Marceau
Mathis Nourrisson
Nadia Ozanne
Octave Pelletier
Prune Quesnel
Quentin Rabourdin
Rosalie Sauvage
Samir Thibault
Tiphaine Urbain
Ulysse Vasseur
Valentine Wattiez
Wassim Ximenes
Yasmine Yvenou
Zoe Zelmati
Anatole Bourdaloue
Berenice Crampon
Casimir Ducrocq
Delphine Frochot
Eusebe Gringoire
Fanny Hurluberlu
Gaspard Ippolite
Henriette Jacquemin
Isidore Klapisch
Josephine Lartigue
Kevin Mouchard
Ludivine Nectoux
Marcel Oudinot
Nathalie Pignouf
Odilon Quenneville
Pierrette Ravachol
Raoul Soubeyran
Solange Tourbillon
Thibaut Uzanne
Ursule Vaillancourt
Victorien Wachtel
Wilhelmine Xanthos
```

```
Xavier Yvrard
Yolande Zuberbuhler
Zacharie Amblard
```

2.8 /opt/GSM/osmo-operator/data/op.txt

575 octets, 25 lignes → 25 lignes

```
# op.txt - noms d'operateur, un par ligne. start-docker.sh en tire un au hasard par
# operateur au lancement.
# PAS D'ESPACE dans un nom : la commande VTY "short name" d'osmo-msc prend UN
# mot, et un nom coupe en deux y passe pour un argument surnumeraire - osmo-msc
# refuse alors la ligne et demarre sans nom de reseau.
FreeRadis
OrangeMecanique
SFRagile
BouyguesTrous
CelluleGrise
OndesDeChoc
RoamingEmpire
AntenneRelais
SansFilDeFer
BurstEnStock
TrameCassee
ReseauQuiTombe
BarreEtDemie
CoupeCircuit
HandoverPerdu
KiPerdu
PasDeSignal
DebordeTelecom
EchoLointain
GigaFlop
```

2.9 /opt/GSM/osmo-operator/data/pays.txt

709 octets, 24 lignes → 24 lignes

```
# pays.txt - les pays du banc, un par ligne. Le MCC designe un PAYS, et sur ce
# banc un pays est un noeud : le noeud 1 habite la premiere ligne, le 2 la
# deuxieme... start-docker.sh melange la liste a chaque lancement, puis s'y tient pour
# toute la duree du run (un pays qui change en cours de route ne veut rien dire).
# Aucun n'existe. C'est fait expres : si l'on voit "Syldavie" dans un journal,
# on sait qu'on regarde le banc et pas un vrai reseau.
Syldavie
Bordurie
Poldevie
Molvanie
Ruritanie
Freedonie
Elbonie
Latverie
Zubrowka
Grand-Fenwick
Costa Gravas
Bas-Bourbier
Haute-Ferraille
Republique du Bruit Blanc
Duche de Sans-Signal
Ile de la Trame Perdue
Confins de Bassebande
Principaute de Roamie
```

2.10 /opt/GSM/osmo-operator/doc/README-architecture.md

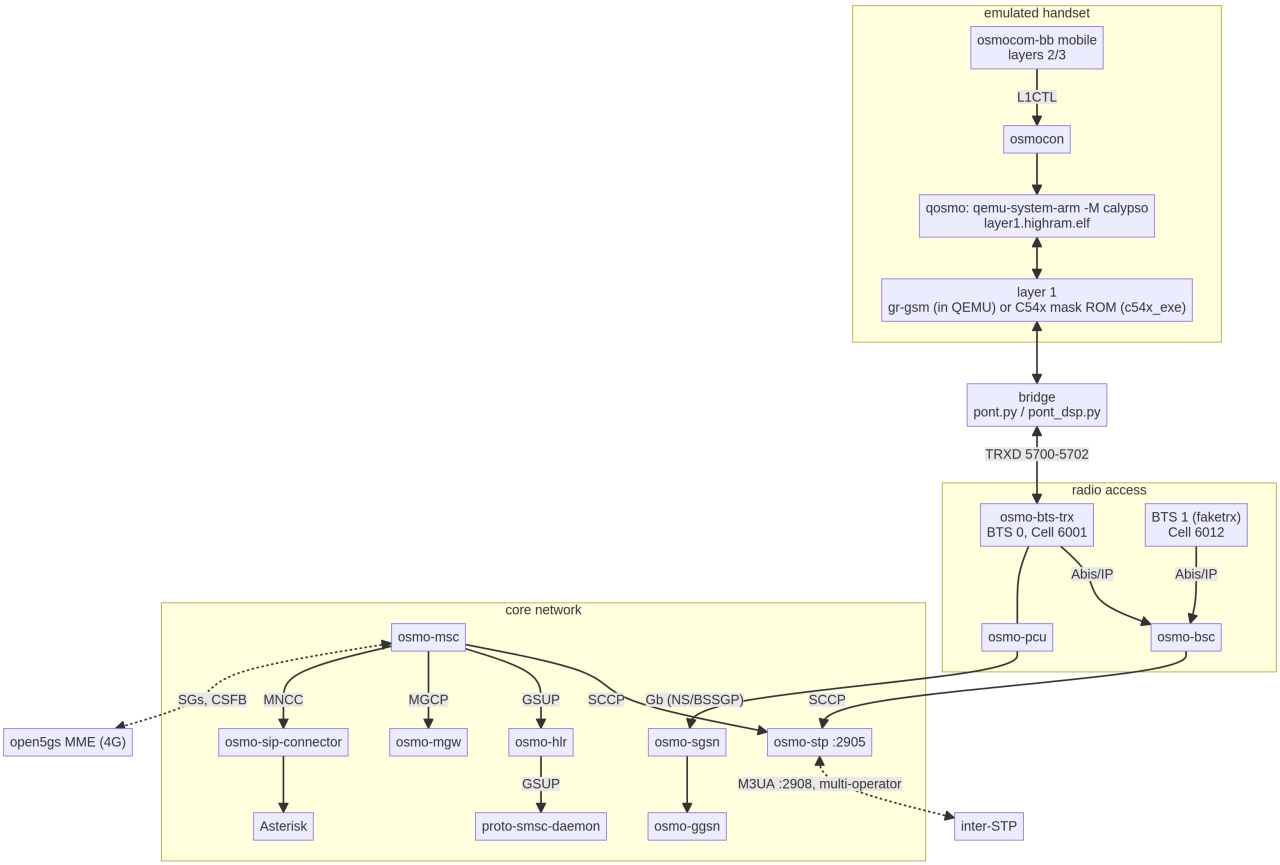
4561 octets, 134 lignes

2.10.1 osmo-operator — architecture

How the pieces fit: one PLMN on one machine, the emulated handset, and the flows of a location update, an SMS and a call.

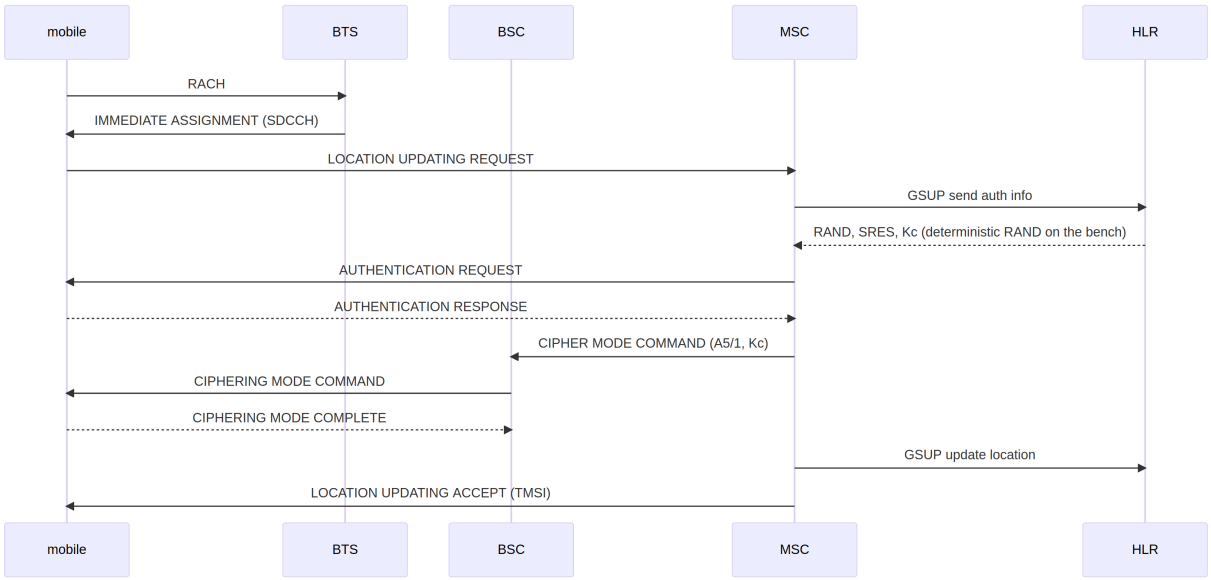
← README · Details: wiki/Reference, README-grgsm, README-dsp, README-ss7, README-lte

2.10.1.1 The whole bench



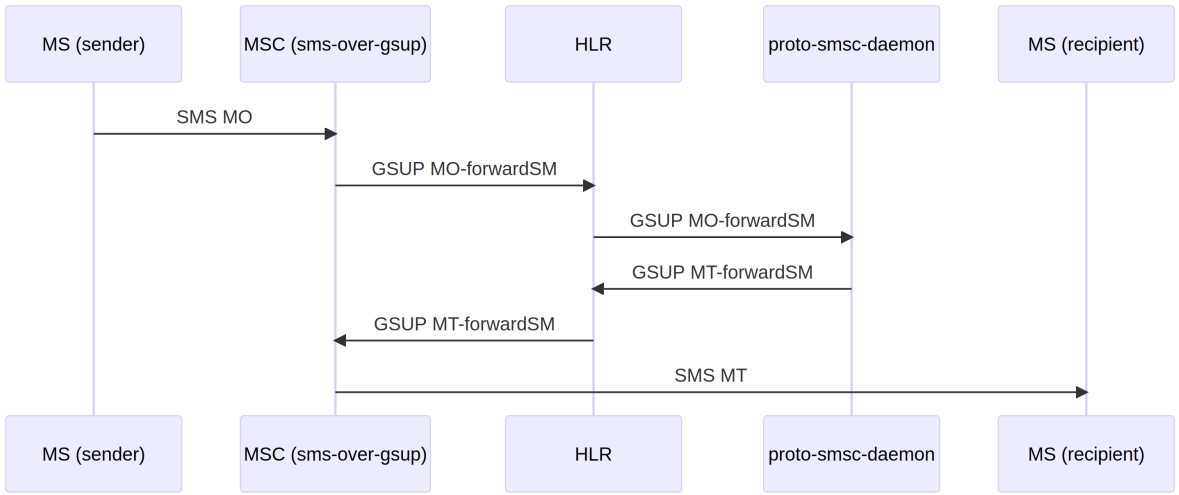
Everything in one operator talks over `127.0.0.1`. The local STP listens on `127.0.0.1:2905` straight away, without waiting for the Docker interface.

2.10.1.2 Location update and authentication



The A3A8 patches (`*-force-rand-toy`) make RAND deterministic, so a run is reproducible from one boot to the next.

2.10.1.3 SMS



SMPP is on `:2775` with a test ESME. Between operators, `sms-interop-relay.py` forwards over TCP 7890 (README-ss7).

2.10.1.4 Voice



`600` is the echo test: on the reference run, a 1 kHz tone comes back after 0.54 s (README-tests).

2.10.1.5 GPRS / EDGE

BSSGP/NS from the PCU to osmo-sgsn, then PDP contexts through osmo-ggsn (GTPv1-C/U). On the DSP chain the GPRS modem runs on `virtphy` (README-dsp).

2.10.1.6 Where the code lives

path	role
`start-native.sh`	the launcher (native, container, ISO)
`start-docker.sh`, `compose.sh`	Docker, multi-operator
`start-interstp.sh`, `set_stp_ip.sh`	the SS7 hub and node addressing
`network/`	WAN table, mesh, firewall, VirtualBox, ISM
`lib/gabarits.sh`, `configs/`, `generate_configs.sh`	config templates
`pont/`	the radio bridges (`pont.py`, `pont_dsp.py`, `airmesh.py`)
`scripts/`	run-time services and diagnostics (SMS relay, audio, call diag)
`checks/`	read-only health checks
`iso_modules/`, `build-iso.sh`	the ISO and SD image
`packaging/`	`.deb` cache and standalone packages
`/opt/GSM/qosmo`	QEMU fork, `run.sh`, `run_modules/`
`/opt/GSM/c54x_exe`, `/opt/GSM/grgsm_exe`	external layer 1s

2.11 /opt/GSM/osmo-operator/doc/README-arm.md

4410 octets, 88 lignes

2.11.1 osmo-operator — arm64 / Raspberry Pi 4

The same stack, built for arm64 and shipped as an **SD card image** for the Raspberry Pi 4 (Armbian 24.04 base). It is not a live ISO: the root filesystem is a persistent ext4.

← README · Details: wiki/ISO, wiki/Build

2.11.1.1 What you get, and what you don't

	arm64 image
Format	`osmo-operator-<…>-rpi4.img`, MBR: FAT32 `RPICFG` (firmware, kernel, DTBs) + ext4 `armbi_root`
Base	Armbian 24.04, bcm2711 kernel and rpi4b BSP (apt.armbian.com)
2G bench	yes: Osmocom core, Calypso QEMU, gr-gsm and DSP layer 1s, compiled natively for aarch64
`--lite`	yes
Desktop (`--desktop`), installer, `--all`	no: refused by `build-iso.sh` for arm64
4G / CSFB / postmarketOS	no: srsRAN and Open5GS are not built for aarch64 (`88-lte-pmos.sh` exits early)

2.11.1.2 Flash and boot

```
sudo dd if=osmo-operator-<…>-rpi4.img of=/dev/sdX bs=4M status=progress conv=fsync
```

At first boot the root partition grows to fill the card (`osmo-grow-root`). Then use the bench as in README-native: `./start-native.sh`.

2.11.1.3 Build it

On an x86 host, everything is emulated (qemu-user-static, `docker buildx --platform linux/arm64`, `debootstrap --foreign`). It works, but **expect about ten hours** for the first build. The `.deb` cache makes later builds short.

```
sudo ./build.sh --arch=arm64          # the arm64 Docker image only (tagged :arm64)
sudo ./build-iso.sh --arm --lite      # the SD image
```

On an arm64 host, `build.sh` and `iso_modules/40-rootfs.sh` see that target and host match, and take the native path: no qemu-user, no buildx `--platform`.

What is specific to the arm64 build:

- `iso_modules/82-arm-natif.sh` compiles `toast`, the `qosmo` launcher, `c54x_exe` / `grgsm_exe` and the bench `.deb` **inside the arm64 chroot**. Doing it on the x86 host would silently put x86 binaries in the rootfs. It also installs the Pi firmware partition contents, with a hook that replays on every kernel update.
- `iso_modules/92-rpi-image.sh` assembles the image without mounting anything (`mke2fs -d`, `mcopy`, `sfdisk`, `dd`).

Host packages: `./build.sh --arm` installs everything the arm64 chain needs on the host (`qemu-user-static`, `binfmt-support`, `docker-buildx`, `debootstrap`, `mtools`, `dosfstools`, `e2fsprogs`, `kpartx`, …).

2.11.1.4 CI

- `.github/workflows/docker-arm.yml` builds the arm64 base image **natively** on an `ubuntu-24.04-arm` runner and publishes it to GHCR as `osmocom-nitb:base-arm64-<fingerprint>`.
- `.github/workflows/build-arm.yml` pulls that image and produces the SD image. It runs on tags `rpi-*` / `arm-*` or by hand. The emulated x86 path cannot finish within GitHub's 6-hour job limit, hence the native runner.

2.11.1.5 Build it with the official Armbian framework (armbian/build)

`armbian/build-armbian.sh` builds the same image through github.com/armbian/build instead of the hand-made assembly of `build-iso.sh --arm`: Armbian produces the kernel, the board support, the firmware partition and the `.img`; the stack is dropped in through the framework's own hook, `userpatches/customize-image.sh`, with the contents of the project's arm64 Docker image bind-mounted as `/tmp/overlay`.

```
sudo ./build.sh --arch=arm64                       # host packages (qemu-user, binfmt, buildx...) + the arm64
↪  stack image, once
sudo armbian/build-armbian.sh                      # clone armbian/build, overlay, compile.sh
sudo armbian/build-armbian.sh --prepare            # everything but compile.sh (inspect userpatches/)
sudo armbian/build-armbian.sh --board=orangepi5 --branch=edge
```

Defaults: `BOARD=rpi4b BRANCH=current RELEASE=noble`, clone in `/opt/GSM/armbian-build` (`--armbian-dir=`), image in its `output/images/`. The stack source is `osmocom-nitb:arm64` if present, else the published `ghcr.io/<repo>/osmocom-nitb:base-arm64-<fingerprint>`, else `--image=`. `customize-image.sh` installs the runtime packages, `dpkg -i` the build `.debs`, copies `/usr/local`, `/opt/GSM`, `/etc/osmocom` and the units, compiles natively what an x86 host cannot (toast, the qosmo launcher, `c54x_exe`/`grgsm_exe`, the bench `.debs`), sets the accounts (root/osmo, osmocom/osmo) and skips Armbian's first-login wizard. `--no-stack` gives a bare Armbian image.

2.12 /opt/GSM/osmo-operator/doc/README-docker.md

4357 octets, 120 lignes

2.12.1 osmo-operator — Docker

Run the bench in containers, or build the images yourself. Docker is also how several operators run side by side on one host (README-ss7).

← README · Details: wiki/Build, wiki/Environnement, wiki/Reference §3

2.12.1.1 Run the published image

```
sudo docker pull ghcr.io/bbaranoff/osmo-operator/osmocom-nitb:latest
git clone https://github.com/bbaranoff/osmo-operator
cd osmo-operator
sudo ./launch/start-docker.sh
```

Then enter the container and bring the stack up:

```
sudo docker exec -ti osmo-operator-1 bash

cd /opt/GSM/osmo-operator
./start-native.sh --regen     # regenerate configs from the templates (first run)
./start-native.sh --stop      # make sure nothing is left over
./start-native.sh             # start the bench
```

`start-native.sh` attaches to tmux at the end: you are looking at a live GSM network with a mobile camped on it. Re-attach later with:

```
docker exec -ti osmo-operator-1 tmux -S /tmp/osmocom_tmux attach -t osmocom
```

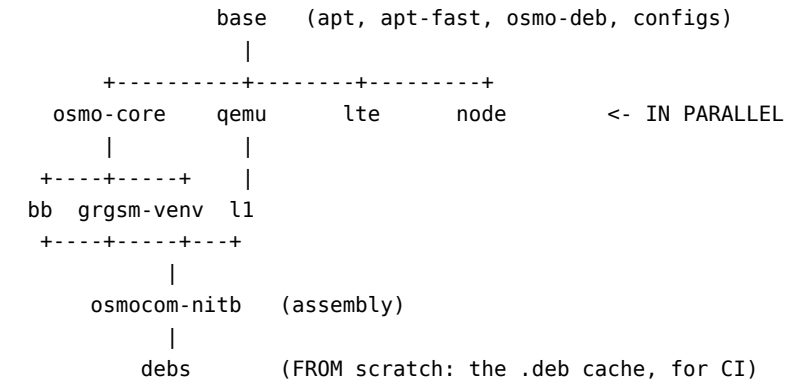
Everything `start-native.sh` does is the same as in native mode: README-native.

2.12.1.2 Build the images

```
sudo ./build.sh               # build osmocom-nitb (apt-fast, compose v2, .deb cache)
sudo ./build.sh --no-cache    # recompile everything and rewrite the cache
sudo ./build.sh --lite        # without the workshops
sudo ./build.sh --stp         # osmocom-stp: the inter-STP alone (Dockerfile.stp)
sudo ./build.sh --arch=arm64  # cross-build, see README-arm
```

2.12.1.2.1 The build is a graph, not a chain

`Dockerfile` is split into BuildKit stages, so the long compilations that do not depend on each other run **at the same time**:



`bb`, `l1` and `grgsm-venv` are deliberately **not** parallel to `osmo-core`: they link against libosmocore. `--target osmocom-nitb` is **mandatory** (`debs` is the last stage, so without it Docker builds an empty image). `build.sh` and `compose.yaml` already pass it.

2.12.1.2.2 The `.deb` cache

Everything the image compiles comes out as a `.deb` through `packaging/osmo-deb.sh` and stays **on the host**, in `/var/cache/osmo-debs` (`OSMO_DEB_CACHE` to move it). The second build installs with `dpkg -i` and compiles nothing. Each package carries both the installed files and the `/opt/GSM` source tree, so a cached rebuild still gives you a workshop you can read, patch and `make` in.

```
./compose.sh debs                  # what is in the cache
./packaging/build-debs.sh          # standalone .deb -> packaging/dist/
sudo dpkg -i packaging/dist/*.deb  # refuses to install over a git clone at the same path
```

2.12.1.3 Docker Compose

```
sudo ./compose.sh build [--no-cache] [--lite] [--stp]
sudo ./compose.sh up [--ms 2] [--phy faketrx]    # hub + osmo-operator-1, HLR populated
sudo ./compose.sh down | ps | logs | shell
```

2.12.1.4 PHY modes (mobile side, in a container)

Selected through `PHY_MODE` inside the container before calling `run.sh`:

`PHY_MODE`	stack	use
`faketrx` (default)	`fake_trx` → `trxcon` → `mobile`	multi-MS, fast, no DSP
`virtphy`	`osmo-bts-virtual` ↔ `virtphy` ↔ `mobile`	multi-MS over UDP multicast
`qemu`	`osmo-bts-trx` ↔ bridge ↔ Calypso QEMU ↔ `mobile`	emulated baseband, 1 MS

`PHY_MODE=qemu` forces `N_MS=1`: the bridge uses fixed UDP ports, so one emulated Calypso per container. For several virtual handsets, run several containers.

2.12.1.5 Several operators

`sudo ./launch/start-docker.sh` asks for a bridge mode and N operators (1 to 9), each in its own container, joined by an inter-STP container. See README-ss7.

2.12.1.6 CI images

`.github/workflows/docker.yml` publishes the amd64 base to GHCR (`osmocom-nitb:base-<fingerprint>`), and `docker-arm.yml` the arm64 one (`base-arm64-<fingerprint>`). An unchanged base (same fingerprint) is not rebuilt.

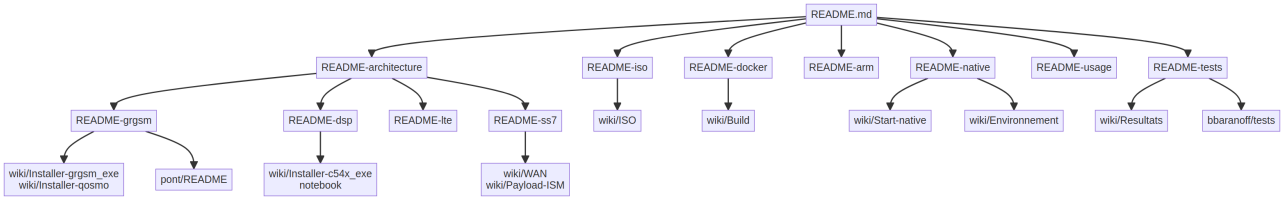
2.13 /opt/GSM/osmo-operator/doc/README-docs.md

11013 octets, 162 lignes

2.13.1 osmo-operator — documentation map

Every document in the project, and where to start depending on what you want.

← README



2.13.1.1 Topic READMEs (this repository)

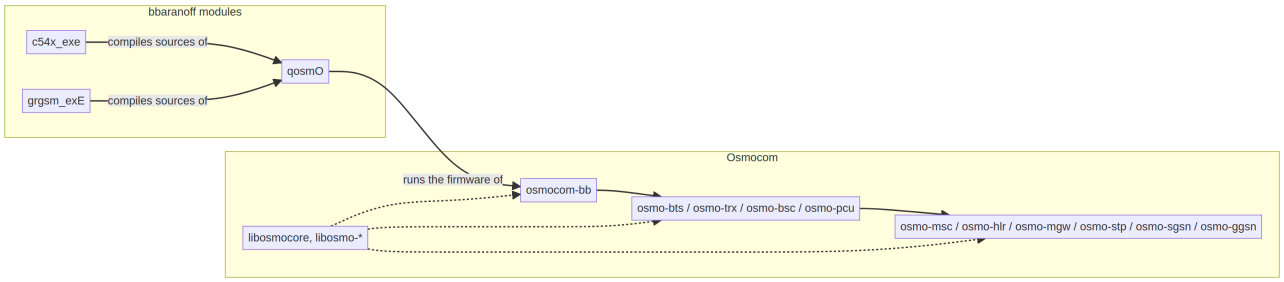
document	subject
README	entry point, quick start, index
README-iso	the live ISO: download, run, launchers, contents
README-docker	Docker images, build graph, `.deb` cache, Compose, PHY modes
README-native	`install.sh`, `start-native.sh`, profiles, `qosmo` launcher
README-arm	arm64 / Raspberry Pi 4 SD image
README-architecture	how the pieces fit, LU / SMS / call flows, code map
README-grgsm	the gr-gsm layer 1 chain, radio diagnostics
README-dsp	the C54x mask-ROM DSP chain, current status
README-lte	4G, CSFB, postmarketOS phone
README-ss7	multi-operator, inter-STP, point codes, WAN
README-usage	tmux, VTY, numbers, ports, Wireshark
README-tests	`banc-max.sh` ladder, reference run, pytest, `checks/`

2.13.1.2 Wiki

page	subject
Home	quick start and reference card
Reference	the former full README
Start-native	`start-native.sh` step by step, option manual
Build	`build.sh`, `build-iso.sh`, `.deb`, `install.sh`
install-from-scratch · English	the Dockerfile replayed step by step on a bare Ubuntu 24.04 (generated by `tools/dockerfile-to-install-md.sh`; English notes in `install-from-scratch.en-notes.md`)
c54x_exe-depuis-ubuntu-vierge · English	the DSP emulator alone, from a clean machine to the running TI mask ROM; the arm-elf toolchain
ISO	ISO contents, build, install
Environnement	variable precedence, what reaches QEMU
Resultats	reference `check_all.sh` runs
Telephone-emule	the emulated phone on your own Osmocom stack
Lancer-a-la-main · lancer-pas-a-pas.ipynb	the bench process by process (page, and runnable notebook)
Installer-qosmo	build the QEMU fork
Installer-grgsm_exe	build the gr-gsm layer 1
Installer-c54x_exe	build the DSP emulator
README-notebook · installer-osmo-jusquau-dsp.ipynb	the install notebook and how to run it
SIM	real SIM in an Omnikey reader, or softSIM, over SAP
WAN	linking two nodes, LAN or Internet
Payload-ISM	SMS text over an ISM radio link

2.13.1.3 Components and their READMEs

Every piece of software the bench is built from, with a link to its own README. “Patches” are the files of `patches/` applied on top of upstream.



2.13.1.3.1 bbaranoff modules

component	role	source tree
qosmO	QEMU fork `-M calypso` (ARM7TDMI + platform), `run.sh`, `run_modules/`, the `qosmo` launcher; layer 1 chosen at `configure` (`--enable-l1-grgsm` / `--enable-l1-dsp`)	`/opt/GSM/qosmo`
c54x_exe	TMS320C54x on the TI mask ROM, outside QEMU; `pont_dsp.py`; the `--dsp` chain (README-dsp)	`/opt/GSM/c54x_exe`
grgsm_exE	gr-gsm layer 1 outside QEMU (README-grgsm)	`/opt/GSM/grgsm_exe`
qosmo-overlay	the files qosmO adds to / changes in upstream QEMU	`/opt/GSM/qosmo-overlay`
tests	end-to-end test suite, `banc-max.sh`, run reports (README-tests)	`/opt/GSM/tests`
osmo-egprs-web	the web dashboard on `:8080`	`/opt/GSM/osmo-egprs-web`
firmware	prebuilt osmocom-bb firmware (`layer1.highram.elf`)	`/opt/GSM/firmware`

2.13.1.3.2 Osmocom

Upstream on gitea.osmocom.org; sources in `/opt/GSM/<name>`.

component	role	patches
libosmocore	core library: VTY, logging, GSM coding, GSUP	
libosmo-abis	Abis/IP (BTS ↔ BSC)	
libosmo-netif	network interface helpers (SCTP, IPA)	
libosmo-sigtran	SCCP / M3UA, and **osmo-stp** (local STP and inter-STP)	
libosmo-gprs	GPRS protocol stack (RLC/MAC, LLC, SNDCP)	
libosmo-dsp	DSP helpers	
libsmpp34	SMPP 3.4	
libgtpnl	kernel GTP-U through netlink	
osmo-bts	BTS (`osmo-bts-trx`, `osmo-bts-virtual`)	`osmo-bts-force-rand-toy`
osmo-trx	transceiver	`osmo-trx-ipc-ts-frame-align`, `osmo-trx-rach-per-ra-table`
osmo-bsc	BSC	
osmo-msc	MSC / VLR, SMS-over-GSUP, SGs (CSFB)	`osmo-msc-force-rand-toy`
osmo-hlr	HLR, GSUP	`osmo-hlr-force-rand-toy`
osmo-mgw	media gateway (MGCP, RTP)	
osmo-sip-connector	MNCC ↔ SIP (to Asterisk)	
osmo-pcu	PCU (GPRS/EDGE)	
osmo-sgsn	SGSN	
osmo-ggsn	GGSN (GTPv1-C/U); no top-level README.md upstream	
gapk	GSM audio codecs (FR/HR/EFR), audio bridge to the mobile	`gapk-pq-alsa-output-probe`

component	role	patches
osmocom-bb	phone side: `mobile`, `trxcon`, `osmocon`, `layer1` firmware, `fake_trx` (no README in the repo: project wiki)	`osmocom-bb-*` (7 patches)
gsup-smsc-proto	`proto-smsc-daemon` / `proto-smsc-sendmt`, the SMSC	
softsim	softSIM served over SAP (wiki/SIM)	

2.13.1.3.3 Third-party

component	role	patches
gr-gsm	GSM receiver for GNU Radio (layer 1 of the `--grgsm` chain)	`grgsm-optional-qtgui-import`, `grgsm-receiver-publish-bsic-fn`
Asterisk	PBX: voice, echo test 600, SIP trunks between nodes	
open5gs	4G core (README-lte)	
srsRAN 4G	srsENB / srsUE over ZMQ	
sysmo-usim-tool	programming the sysmoISIM	
pmbootstrap	postmarketOS phone image	`pmbootstrap-osmo-bench-qemu`, `pmaports-linux-postmarketos-stable-ppp`

2.13.1.4 Other documents

document	subject
`pont/README.md`	A5, Kc and SACCH: measurement report on the radio bridge
`environment/README.md`	environment variables and path resolution
`services/README.md`	shipped systemd units
`navigation/QUICKSTART.md`, `navigation/STATUS.md`	the SS7 console
bbaranoff/tests	end-to-end test suite and run reports
qosmO	the QEMU fork: one base, two layer 1s
c54x_exe	the C54x DSP on the TI mask ROM
grgsm_exE	the gr-gsm layer 1 outside QEMU

2.14 /opt/GSM/osmo-operator/doc/README-dsp.md

4099 octets, 90 lignes

2.14.1 osmo-operator — the DSP chain (`--dsp`)

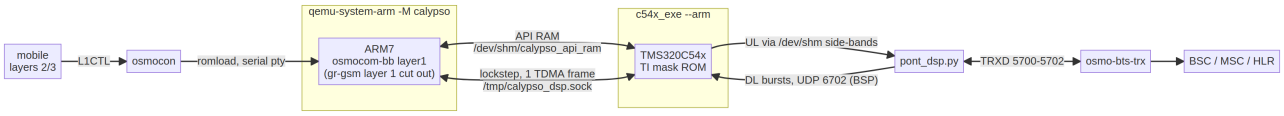
With `--dsp`, the handset's own DSP takes over. A TMS320C54x core is emulated outside QEMU (c54x_exe) and runs the mask ROM dumped from a real Calypso (PROM0-3, DROM, PDROM). This has been the default since 2026-10-01.

← README · Details: wiki/Installer-c54x_exe, wiki/Start-native §8-dsp, install notebook, c54x_exe README

```
./start-native.sh                       # real C54x mask ROM (default)
./start-native.sh --dsp                 # the same, explicitly
MODE=dsp PONT=1 /opt/GSM/c54x_exe/run.sh  # the mobile chain alone, no network
BANC_DSP=none ./start-native.sh --dsp     # the stack only, no DSP bench
```

2.14.1.1 How it fits together

`--dsp` does **not** select a different QEMU. The stack comes up exactly as in gr-gsm mode (core, BTS, side-car, tmux), and only the four Calypso modules (`qemu`, `pty`, `osmocon`, `l2`) are handed over to `c54x_exe/run.sh`:



The five steps of `c54x_exe/run.sh`:

```
1. c54x_exe --arm     the DSP: /tmp/calypso_dsp.sock + shared API RAM, DL bursts on UDP 6702
2. qemu-system-arm    the ARM + osmocom-bb layer1, gr-gsm layer 1 cut out
3. osmocon            romload on the serial pty, L1CTL relay
4. mobile             osmocom-bb layers 2/3
5. pont_dsp.py        TRXD 5700-5702 facing osmo-bts-trx
```

`--dsp` starts **no bridge of its own**: `c54x_exe/run.sh` runs one on 5700-5702 (step 5). A second one on the same ports would die on *Address already in use*. `./start-native.sh --stop` tears down the DSP bench too, without repeating `--dsp`.

2.14.1.2 What the ROM does today

From the reference run of 2026-10-16 (`tout-en-un-20261016-164539.qmd`, details in README-tests): ladder **27/27**, and on the signal path **14 steps by the ROM, 0 by the host**.

- Handled by the ROM: FCCH detection, SCH/BSIC, BCCH SI1/2/4, SACCH SI5/6, RACH, SDCCH both ways, A5 on the modelled coprocessor, TCH/F downlink (Viterbi), FACCH and SACCH uplink, uplink TCH speech, and the ROM's FR codec through the voice serial port (VSP).
- End to end: camping, location update, SMS MO/MT, USSD, MO/MT calls with A5/1; a 1 kHz tone comes back after 0.54 s at +59.7 dB.
- Open points: `B_BFI` is still raised on bad frames and FACCH steals (masked by `layer1.highram_patched`), frequency hopping is not handled, and the DSP takes 4.3–4.6 ms of host time per 4.62 ms frame (worst DSP frame 3.37 ms on the reference run).

2.14.1.3 GPRS / EGPRS on the DSP bench

`scripts/egprs-dsp.sh` (and `osmo-egprs-dsp.service`) runs the EGPRS workflow next to the DSP mobile. The calypso layer1 firmware has no PDCH support (`L1CTL_GPRS_*`), so the **DSP gate is closed**: the GPRS modem runs on a virtual layer 1 (`virtphy`) in the same PLMN/LAC/RA, while the DSP mobile camps on BTS 0.

2.14.1.4 Building

`c54x_exe` compiles `qosmo`'s sources (it does not copy them) and needs the ROM files in `rom/calypso_dsp.*.bin`. Step by step: wiki/Installer-c54x_exe. The notebook replays the whole Dockerfile natively, from apt to a call on the DSP.

Test it: `cd /opt/GSM/tests && ./banc-max.sh --dsp --restart`.

2.15 /opt/GSM/osmo-operator/doc/README-grgsm.md

4798 octets, 109 lignes

2.15.1 osmo-operator — the gr-gsm layer 1

`--grgsm` runs the handset with gr-gsm as layer 1. This chain works end to end: camping, location update, authentication, A5/1, SMS and voice calls. Use it to get work done. To study the real DSP instead, see README-dsp.

← README · Details: wiki/Installer-grgsm_exe, wiki/Installer-qosmo, wiki/Lancer-a-la-main, `pont/README.md`

```
./start-native.sh --grgsm
```

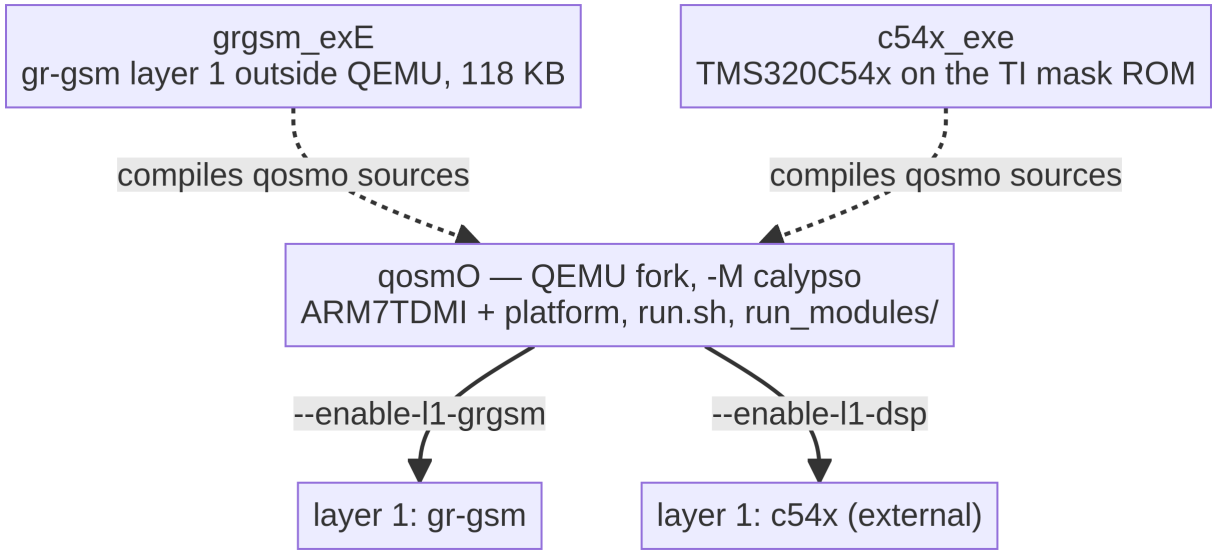
2.15.1.1 The two layer-1 chains

Both chains run the same core network, the same BTS and the same `mobile`. They differ in **who demodulates the air interface**.

	`--grgsm`	**`--dsp`** (default since 2026-10-01)
Layer 1	gr-gsm inside QEMU + `grgsm_exe` bridge	real TI mask-ROM C54x, **outside** QEMU
QEMU's job	ARM7 + gr-gsm layer 1	ARM7 only (`CALYPSO_DSP_EXTERN=1`)
Bridge	`pont/pont.py`	`pont/pont_dsp.py`
Mobile VTY	4247	4347 (4247 on the 2026-10-16 reference run)

2.15.1.2 One QEMU tree, two layer 1s

Since 2026-09-25 there is a **single QEMU tree**. The layer 1 is a `configure` option, not a separate repository.



tree	what it is
`/opt/GSM/qosmo`	the QEMU fork
`/opt/GSM/grgsm_exe`	the gr-gsm layer 1 outside QEMU, 118 KB against the 93 MB `qemu-system-arm` that normally hosts it
`/opt/GSM/c54x_exe`	the DSP, see README-dsp

`qemu-system-arm -M help` names the layer 1 actually linked in:

configure	`-M help` shows
(nothing)	`couche 1 : aucune`

configure	`-M help` shows
`--enable-l1-grgsm`	`couche 1 : grgsm`
`--enable-l1-dsp`	`couche 1 : c54x`

2.15.1.3 The chain



To start it process by process, with proof at each step: wiki/Lancer-a-la-main. To plug only the emulated phone into your own Osmocom stack: wiki/Telephone-emule.

2.15.1.4 Diagnosing the radio side

Read `pont/README.md` before chasing a radio fault through a CRC counter. The causes documented there (overwritten Kc, the SACCH L1 header, C0 filling, an inert `CALYPSO_CANNED`) *all* show up as "the bridge has CRC errors".

```
./start-native.sh --check-paths     # verify the declared dependencies
./start-native.sh --verbose         # show the modules' output
```

log	contents
`/var/log/osmocom/qemu.log`	ARM/DSP emulation, BSP, IRQ
`/var/log/osmocom/bridge.log`	QEMU ticks, IND CLOCK, DL/UL bursts
`/var/log/osmocom/run.sh.log`	orchestration

symptom	likely cause	check
`bridge: timeout`	QEMU never started the TPU/DSP	`grep TINT0 /var/log/osmocom/qemu.log`
`FBSB result=255` (no cell)	layer 1 does not detect the FB	`grep "IMR change" /var/log/osmocom/qemu.log | wc -l`
`osmo-bts-trx: PC clock skew too high`	the bridge stopped sending `IND CLOCK`	restart the bridge
`/tmp/osmocom_l2` never created	the firmware does not boot	`head -50 /var/log/osmocom/qemu.log`
Address already in use on 5700	two bridges: a `--dsp` run left one behind	`./start-native.sh --stop`

QEMU monitor: `socat - unix-connect:/tmp/qemu-calypso-mon.sock`. `--assembly-logs` gives a trace of the emulated ARM (`-d in_asm`: each block once, when translated). `ASSEMBLY_LOGS_FLAGS=in_asm,exec,nochain` also logs every execution: the ARM becomes ~150 times slower. The bridge clock follows the DSP down to `PONT_HORLOGE_MAX` times real time (16 with `ASSEMBLY_LOGS=1`, 4 otherwise; the trace goes to `/var/tmp/c54x-pont`), but the wall-clock timers of mobile and of the core network still run, so use that only to chase the firmware.

Test it: `cd /opt/GSM/tests && ./banc-max.sh --grgsm --restart` (README-tests).

2.16 /opt/GSM/osmo-operator/doc/README-iso.md

5543 octets, 120 lignes

2.16.1 osmo-operator — the live ISO

The fastest path: a GNOME live image with the whole bench (2G, DSP, 4G, CSFB, postmarketOS phone), Wireshark, Linphone and the I/Q spectra. Nothing to compile or configure.



Figure 4: The ISO desktop: Telecom Operator, no RF, full stack

← README · Details: wiki/ISO, wiki/Home, wiki/Build

Live ISO password	`osmo`
postmarketOS phone password	`147147`
Minimum host	x86_64, 8 GB RAM, 4 cores, nested virtualization in a VM
Image identity	`IMAGE_VERSION="OSMO_EGPRS_V2"` · `OSMO_ROLE=operator` · `OSMO_LITE=0`

2.16.1.1 Download

Latest release, or:

- Google Drive (single file)
- GitHub v0.1-9: part-00 · part-01 · part-02 · part-03 · SHA256SUMS

  [!IMPORTANT] Download **every** `.part-NN` file, not just the first one. GitHub caps release files at 2 GiB, so the image ships in pieces.

```
cat osmo-operator-desktop.iso.part-* > osmo-operator-desktop.iso
sha256sum -c --ignore-missing SHA256SUMS
```

2.16.1.2 Run the image

The live root is a tmpfs (~6 GB in `toram` mode). Logs and captures are bounded and purged at every boot. The bench runs two QEMUs (Calypso and pmOS): in a VM, **enable nested virtualization**, or the phone VM falls back to software emulation and becomes very slow.

QEMU

```
qemu-system-x86_64 -cdrom osmo-operator-desktop.iso -m 8G -enable-kvm \
  -cpu host -smp 4 -nic user,hostfwd=tcp::8080-:8080
```

VirtualBox

1. New VM: Linux / Ubuntu 64-bit, 8 GB, 4 CPUs, 3D off.
2. System → Processor: tick *Nested VT-x/AMD-V* (if greyed out: `VBoxManage modifyvm "<name>" --nested-hw-virt on`).
3. ISO on the optical drive. No disk is needed.
4. NAT port forward: host 8080 → guest 8080.

VMware: new VM from the ISO, Linux / Ubuntu 64-bit, 8 GB, 4 cores, tick *Virtualize Intel VT-x/EPT or AMD-V/RVI*, skip Easy Install.

USB stick (hybrid ISO, boots UEFI or BIOS):

```
sudo dd if=osmo-operator-desktop.iso of=/dev/sdX bs=4M status=progress conv=fsync
```

On Windows, use Rufus in *DD Image* mode. On real hardware KVM is available directly, which is the most comfortable setup.

2.16.1.3 Getting started

Three launchers are pinned in the dock. 2G and 4G are independent and can start in any order. **Launch the smartphone last**, once both networks are up.

	Launcher	What it starts
	2G (red handset)	The GSM core and the emulated radio: osmo-stp / bsc / msc / hlr / sgsn / ggsn / pcu, the Calypso under QEMU, the two BTS and the two osmocom-bb mobiles, Asterisk, the SMSC.
	4G (signal bars)	open5gs, srsENB and srsUE over ZMQ, and the SGs interface to the MSC (CSFB). See README-lte.
☎	**Smartphone**	The postmarketOS VM: oFono on the virtual modem, data on 4G, voice and SMS on 2G.

From a terminal, the same 2G bench, on the Calypso DSP instead of gr-gsm:

```
cd /opt/GSM/osmo-operator && sudo ./start-native.sh --dsp
```

`--menu` asks interactively (layer 1 `grgsm_exe` or `c54x_exe`). Check that everything is in place with `checks/diag-stp-operator.sh`, or the full suite in README-tests.

2.16.1.4 What's inside

- **2G / GSM (Osmocom)**: osmo-stp (M3UA, SCTP :2905), osmo-bsc, osmo-msc, osmo-hlr, osmo-sgsn, osmo-ggsn, osmo-pcu, osmo-mgw (MGCP :2427). Two DCS1800 cells in LAC 1: BTS 0 (Cell ID 6001, BSIC 7) and BTS 1 (Cell ID 6012, BSIC 8). GPRS/EDGE, voice through Asterisk (SIP :5060), SMS-over-GSUP with SMPP :2775, A3A8 with a deterministic RAND so that runs are reproducible, live GSMTAP capture.
- **The emulated Calypso**: `qemu-system-arm -M calypso -cpu arm946` runs `layer1.highram.elf`, loaded by `osmocon` exactly as on a C123. See README-grgsm.
- **The DSP**: the Calypso's TMS320C54x on TI's real mask ROM, with `--dsp`. See README-dsp.
- **4G, CSFB and postmarketOS**: see README-lte.

2.16.1.5 Installing to disk and building the ISO

The live image installs with Calamares. The accounts are the ones from the *Users* screen plus an unlocked root, LUKS is offered in the partitioning choices, and a *Graphics drivers* page offers `nvidia-driver-610` when `lspci` sees a card.

`sudo ./build-iso.sh` does **one** Docker build (`osmocom-nitb`) and derives each rootfs from it: interstp, normal, lite (normal without the workshops), desktop. Options, modules (`iso_modules/NN-*.sh`) and CI are covered in wiki/ISO and wiki/Build. The Raspberry Pi image is in README-arm.

2.17 /opt/GSM/osmo-operator/doc/README-lte.md

3388 octets, 88 lignes

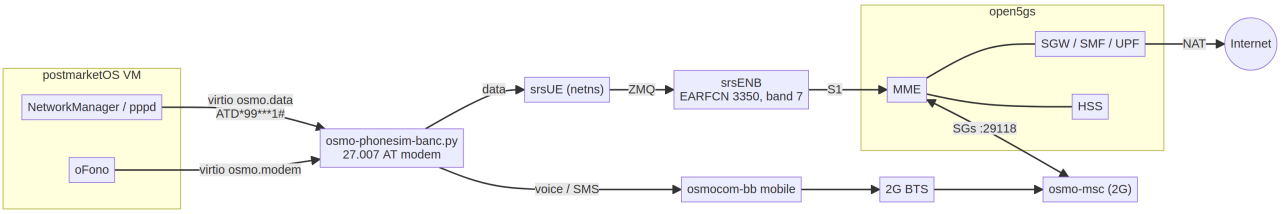
2.17.1 osmo-operator — 4G, CSFB and the postmarketOS phone

On top of the 2G bench: an LTE network (open5gs + srsRAN over ZMQ, no SDR), tied to the GSM core by SGs for **CSFB**, and a postmarketOS phone VM that does data on 4G and voice/SMS on 2G under the same identity.

← README · Details: wiki/ISO, wiki/Home, wiki/Telephone-emule

Available on amd64 only. srsRAN and Open5GS are not built for arm64 (README-arm).

2.17.1.1 Overview



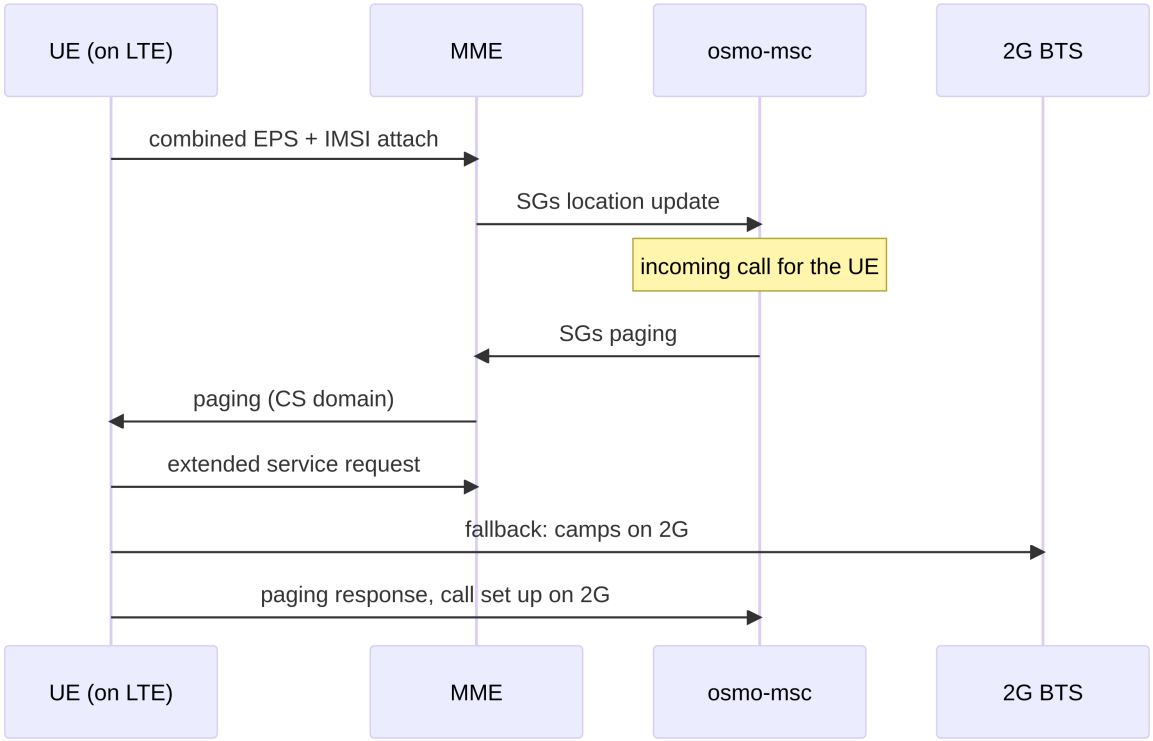
2.17.1.2 4G / LTE

- open5gs core (MME, SGW-C/U, SMF, UPF, PCRF, HSS) on dedicated loopbacks, Prometheus :9090, WebUI :9999, MongoDB behind the HSS.
- PLMN 001/01, TAC 7. NAS null ciphering (EIA2/EIA1 + EEA0) on purpose: this is an analysis bench, and NAS reads in clear in Wireshark.
- srsENB (10 MHz, EARFCN 3350, band 7) and srsUE over ZMQ. Milenage test subscriber IMSI 001010001000001. The UE gets an IP in a netns and reaches the Internet through NAT.

Shipped in `/opt/LTE`, launched with `osmo-lte` and `osmo-epc` (`tools/osmo-lte.sh`, `tools/osmo-epc.sh`), or the **4G** launcher in the dock.

2.17.1.3 CSFB

- SGs between osmo-msc and the MME (:29118), mapping `TAI (001-01, TAC 7) → LAI (001-01, LAC 1)`.
- Combined EPS+IMSI attach: an incoming call is paged by the MME and the UE falls back to the BTS.
- Two subscribers (100101, 100102), one per cell, wired for an inter-cell call with fallback.



2.17.1.4 postmarketOS: the phone

- x86_64 pmOS VM with a patched kernel (`CONFIG_PPP` enabled, missing upstream). The prebuilt kernel is in `/opt/user_interface/kernel/pmos/`. Password `147147`.
- 4G data: NetworkManager → pppd (`ATD*99***1#`) → virtio-console `osmo.data` → `osmo-phonesim-banc.py` → UE netns → srsUE → srsENB → UPF → Internet.
- 2G voice and SMS: virtio-console `osmo.modem`, where `osmo-phonesim-banc.py` plays a 27.007 AT modem driven by oFono. No dongle.
- Audio: two duplex PulseAudio chains (GAPK bridge to the osmocom-bb mobile, and an echo-cancelled chain).

Tools: `tools/osmo-pmos*.sh` (build, install, setup, QEMU), and a patched `pmbootstrap` in `/opt/user_interface/pmos`. **Launch the phone last**, once 2G and 4G are up.

Test it: `cd /opt/GSM/tests && ./banc-max.sh --4g` (README-tests).

2.18 /opt/GSM/osmo-operator/doc/README-native.md

4434 octets, 104 lignes

2.18.1 osmo-operator — native install (no Docker)

The repository installs directly on Ubuntu 24.04 (noble, the same base as the Dockerfile). The launcher, `start-native.sh`, is the same one used inside the container and on the ISO.

← README · Details: wiki/Start-native, wiki/Build, wiki/Environnement, wiki/Lancer-a-la-main

2.18.1.1 Install

The complete native path is the notebook `wiki/installer-osmo-jusquau-dsp.ipynb`: on a fresh Ubuntu 24.04 it builds everything (apt dependencies, Osmocom core, SMSC, osmocom-bb, firmware, qosmo, c54x_exe, runtime setup) and ends with a verified call on the DSP. How to run it (disposable container, fresh 24.04, unattended): wiki/README-notebook.

`install.sh` covers only part of it: apt dependencies (extracted from the Dockerfile), the `osmocom` account, osmocom-bb, gapk, libosmo-dsp, toast, the 4G, configs and desktop icons. It does **not** build the Osmocom core, qosmo, c54x_exe or the firmware, so on a stock 24.04 its last step fails with "demons absents". The notebook reuses its `deps`, `user` and `configs` steps.

```
sudo ./install.sh                 # its steps: deps, sources, build, binaries, configs, desktop
./install.sh --list               # show the steps, do nothing
./install.sh --check              # what is already in place
sudo ./install.sh --only deps     # one step
sudo ./install.sh --only bureau   # desktop icons and shortcuts only
```

To install the pieces one by one instead: wiki/Installer-qosmo, wiki/Installer-grgsm_exe, wiki/Installer-c54x_exe.

2.18.1.2 Start and stop

```
./start-native.sh --regen     # first run: regenerate configs from the templates
./start-native.sh             # start the bench (delegates to /opt/GSM/qosmo/run.sh)
./start-native.sh --status    # what is running
./start-native.sh --stop      # stops the stack AND the DSP bench, with or without --dsp
```

Every start stops the stack first: there is no hot restart. `--force` only restarts modules that are already running instead of skipping them. `--no-attach` starts without attaching to tmux, which is what `osmo-banc.service` uses.

2.18.1.3 Choosing what runs

The layer 1 (who demodulates the air interface):

```
./start-native.sh --grgsm     # gr-gsm layer 1, works end to end   → README-grgsm
./start-native.sh --dsp       # real C54x mask ROM (default)        → README-dsp
./start-native.sh --menu      # ask interactively
```

The profile (how many base stations come up, and how):

profile	stack
`faketrx-qemu` (default)	core + BTS#0 QEMU + BTS#1 faketrx (also `hybrid`)
`faketrx`	core + `fake_trx` + `trxcon` + `mobile`
`qemu`	the Calypso QEMU pipeline alone
`noproc`	core only (also `core`)

```
./start-native.sh faketrx
./start-native.sh --profile qemu
./start-native.sh --list        # show the plan without running it
./start-native.sh --dry-run     # walk through it with no side effects
```

Identity and network options (`--node`, `--op`, `--hub-ip`, `--wan`) are in README-ss7. A real SIM card in an Omnikey reader (`--sap`) is in wiki/SIM.

2.18.1.4 The `qosmo` launcher

`40-qemu.sh` does not call `qemu-system-arm` directly: it calls the C launcher `qosmo`, installed next to it by `ninja install`. The launcher knows which layer 1 its QEMU was built with.

```
qosmo -k /opt/GSM/firmware/board/compal_e88/layer1.highram.elf
qosmo --help
```

Defaults: `-M calypso`, `-cpu arm946`, `-gdb tcp::1234`, `-serial pty -serial pty`, a unix monitor, L1CTL on `/tmp/osmocom_l2`, TRXDv0 on `0.0.0.0:6702`. `start-native.sh` resolves it (`--launcher <bin>` or `QOSMO_LAUNCHER`), builds it if the source is there and the binary is not, and falls back to the plain `qemu-system-arm` command line without it.

2.18.1.5 Variables

Which variable wins (command line, environment, `/etc/osmo-role`, defaults) and what actually reaches QEMU: wiki/Environnement and `environment/README.md`. Systemd units: `services/README.md`.

Using the running bench (tmux, VTY, Wireshark): README-usage.

2.19 /opt/GSM/osmo-operator/doc/README-ss7.md

5913 octets, 141 lignes

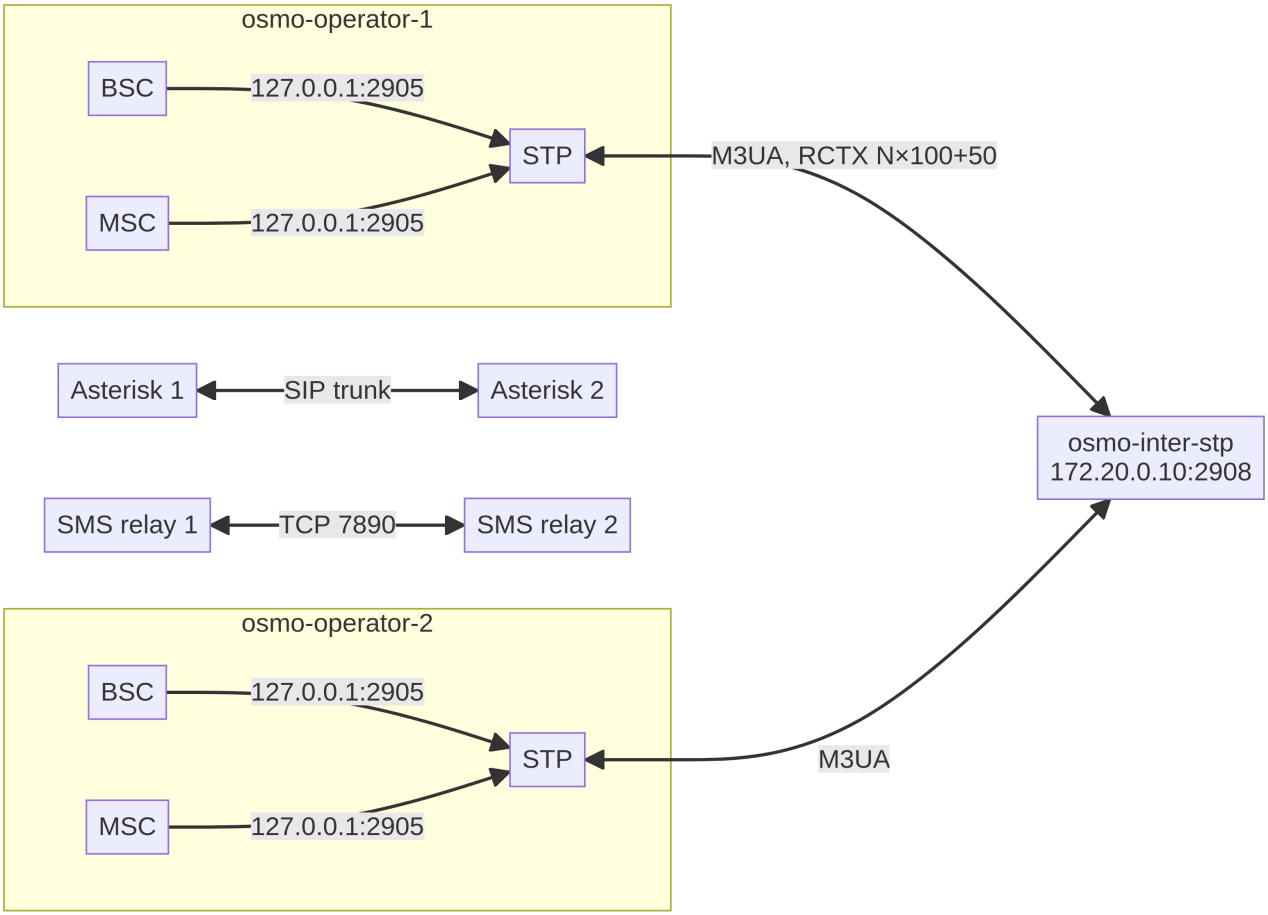
2.19.1 osmo-operator — multi-operator, SS7 and WAN

The standalone bench is one operator. The bench also runs **N operators (1 to 9)**, each with its own core network, joined by a central **inter-STP** that routes M3UA/SCCP between them, on one host (Docker) or across machines (WAN). No configuration file is edited by hand: pjsip, dialplan, SMS routing and the inter-STP config are all generated from N.

← README · Details: wiki/WAN, wiki/Payload-ISM, wiki/Start-native §7-wan, wiki/Reference §9

2.19.1.1 On one host (Docker)

```
sudo ./launch/start-docker.sh          # choose bridge mode, enter N operators
sudo ./launch/start-docker.sh stop
```



Docker network	range	role
`gsm-inter`	`172.20.0.0/24`	interop backbone (inter-STP M3UA), inter-STP at `.10`, operator N at `.(10+N)`
`gsm-net-opN`	`172.20.N.0/24`	operator N private network

Start order matters: the inter-STP must listen on `:2908` **before** the operator STPs connect.

2.19.1.2 Point codes and routing contexts

The single-host plan is `N.23.<role>`. As soon as operators are spread over several **nodes**, the node goes into the point code, otherwise three nodes would present the same address to the hub:

plan	point code	routing context
single host	`N.23.<role>` (inter-STP `0.23.0`)	`rctx_inter = N×100+50`
WAN	`1.<node><op>.<role>`, role 1=MSC 2=STP 3=BSC (inter-STP `0.0.0`)	`node×1000 + op×100 + 50`

The inter-STP uses **AS without a routing key** (catch-all per operator) and routes on the DPC alone. Each operator STP sends unknown PCs to `as-inter` through a `0.0.0/0` catch-all route.



2.19.1.3 SMS and voice between operators

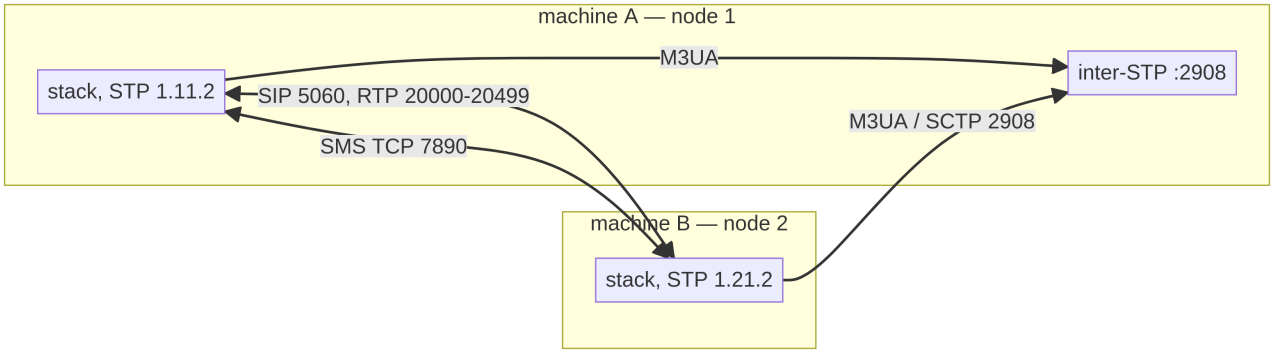
- **SMS**: `sms-interop-relay.py` watches the `proto-smsc-daemon` MO log, parses the GSM 03.40 TPDU, looks up `sms-routing.conf` (longest prefix), and forwards over TCP 7890 to the target relay. That relay resolves MSISDN→IMSI on its HLR and calls `proto-smsc-sendmt`.
- **Voice**: a SIP trunk between the operators' Asterisk instances.

Numbering on the WAN works like a country code. Node N has prefix `NN` (11, 22…), and a mobile's MSISDN is `<node>00<op><ms>`. From node 1, dial `22 + 20011` to reach the first mobile of node 2. The sending node strips the prefix.

2.19.1.4 Across machines (WAN)

```
./start-native.sh --wan                                    # interactive
./start-native.sh --wan-nodes "1:IP:11 2:IP:22" --wan-id 1 # scriptable
./start-native.sh --wan mynode2.conf                       # from another node's card
./start-native.sh --gen-conf                               # write this node's card, start nothing
./start-native.sh --regen-conf …                           # rebuild the WAN table from scratch
./start-native.sh --virtualbox=3                           # mesh with VirtualBox VMs
./start-native.sh --air-mesh                               # mesh the BURSTS: a shared radio medium
./start-native.sh --payload-ism                            # SMS text over an ISM radio link
```

`--node N` picks the node number **at launch**, so one ISO serves all nine nodes. One node runs the inter-STP (`--interstp`), the others point at it with `--hub-ip`.



Over the Internet, put the nodes in a **WireGuard** tunnel: SCTP does not cross consumer NAT, and the table address must be both reachable and local. The full method (tunnel, firewall, start order, layer-by-layer checks, common failures) is in wiki/WAN.

2.19.1.5 Diagnosing PROHIB routes

```
sudo checks/diag-interstp.sh        # from the hub: who is attached, PROHIB routes, flapping
sudo checks/diag-stp-operator.sh    # from an operator node
sudo checks/ss7_check.sh
```

A healthy operator STP route table:

```
N.23.1/14   as-dyn-…  avail  avail  dyn   ← MSC (dynamic)
N.23.3/14   as-dyn-…  avail  avail  dyn   ← BSC (dynamic)
0.0.0/0     as-inter  avail  avail        ← catch-all to the inter-STP
```

symptom	likely cause	fix
route `0.0.0/0` PROHIB	inter-STP down, or wrong IP	check the hub IP and the backbone IP
PROHIB towards a node that was switched off	stale entry in `/etc/osmo-wan.conf`	`--regen-conf` on the node that runs the hub

symptom	likely cause	fix
no dynamic routes	MSC/BSC not reaching the local STP	check that the ASPs point at `127.0.0.1`
ASP DOWN on the inter-STP	Docker race	`docker restart` the operator

The SS7 console (`ss7-console.py`, navigation/QUICKSTART.md) sends MAP queries across the interconnect.

Test it: `cd /opt/GSM/tests && ./banc-max.sh --multi` (README-tests).

2.20 /opt/GSM/osmo-operator/doc/README-tests.md

9224 octets, 181 lignes

2.20.1 osmo-operator — tests and reference results

Two families of tests:

- **bbaranoff/tests** (cloned in `/opt/GSM/tests`) climbs the bench **element by element**, from core to voice, and says how high it got. This is the end-to-end test suite.
- **`checks/`** in this repository is a set of read-only diagnostics of a running stack (daemons, SS7, interconnect, WAN, directory).

← README · Details: wiki/Resultats, wiki/Start-native

2.20.1.1 Reference run: `tout-en-un-20261016-164539.qmd`

The current reference is `tout-en-un-20261016-164539.qmd`, a Quarto run report of two `banc-max.sh` runs in **DSP mode** (layer 1 = real TI mask ROM). A healthy bench must reproduce these lines. Render it with `quarto render tout-en-un-20261016-164539.qmd` (HTML, self-contained).

	run 2026-10-04 14:04	run 2026-10-16 16:32 (reference)
Ladder	**25/25**, 0 failures	**27/27**, 0 failures
Layer 1	`--dsp`	`--dsp`
Coverage (layer 1)	33 handled, 1 degraded, 1 not handled, 2 not observed	31 handled, 1 degraded, 1 not handled, 4 not observed
Signal path	7 ROM, 7 host	**14 ROM, 0 host**, 0 mixed
Skipped	4g, multi, liens, pytest-legacy	data, ms2, 4g, multi, liens, pytest-legacy

The 2026-10-16 ladder, rung by rung (quoted from `verdict.txt` in the report):

#	rung	result	evidence
1	demarrage	OK	full stop, then `start-native.sh --dsp`: core up in 16 s
2	coeur	OK	processes + VTY 4258 4239 4254 4242 4243
3	bts	OK	osmo-bts-trx, VTY 4241
4	services	OK	asterisk 5060, sip-connector, `/tmp/sendmt_socket`
5	ss7	OK	`ss7_check` 7 ✓ 0 ✗, `diag-stp-operator`
6	checks-existants	OK	`global_check`, `call-diag`, `audio-diag` with no failure

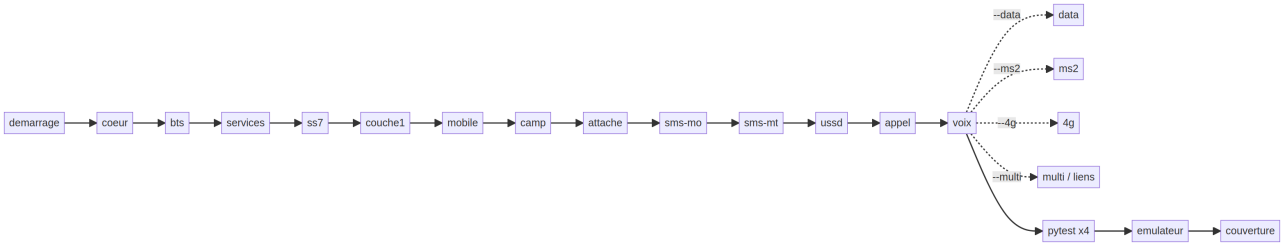
#	rung	result	evidence
7	couche1	OK	`c54x_exe --arm`, `qemu-system-arm`, `osmocon`, `pont_dsp.py`
8	mobile	OK	VTY 4247, IMSI 001010001000001
9	camp	OK	after 1 s, ARFCN 514 (DCS), CGI 001-01-1-6001
10	attache	OK	MM idle normal service, in the VLR, MSISDN 100101
11	sms-mo	OK	SMS to 100102, RP-ACK
12	sms-mt	OK	MSC delivered 1 → 2 (1 s)
13	ussd	OK	`*#100#` → 100101, `*#101#` → IMSI
14	appel	OK	CC ACTIVE after 3 s
15	voix	OK	1 kHz tone back after 0.54 s, +59.7 dB over silence, 100 % of the energy in 950–1050 Hz
16–20	data, ms2, 4g, multi, liens	skipped	need `--data`, `--ms2`, `--4g`, `--multi`
21–24	pytest infra / couche1 / mobile / reseau	OK	11 / 12 (+11 skipped) / 10 / 6 passed
25	pytest-legacy	skipped	needs `--legacy`
26	emulateur	OK	210 ISA examples: 207 ok, 3 FAIL, 22 not assembled; no BC divergence
27	couverture	OK	see below

Layer-1 coverage on the same run (`couverture.txt`): every step from FCCH detection to CHANNEL RELEASE is **handled by the ROM**, including A5 on the coprocessor and uplink TCH speech. The open points are:

function	verdict
TCH/F downlink quality (`B_BFI`)	DEGRADED: 26 proofs, 46 symptoms (really bad frames and FACCH steals, masked by `layer1.highram_patched`)
frequency hopping	NOT HANDLED (outside this setup)
MVKD/MVDK guard, uplink speech ring, host vocoding	NOT OBSERVED
DSP real-time margin	handled: 4.62 ms frame, worst DSP 3.37 ms, 0/1000 frames over budget

2.20.1.2 The ladder: `banc-max.sh`

The bench stacks a dozen elements that depend on each other: no core means no BTS, no layer 1 means no mobile, and without camping there is no attach, SMS or call. When something does not work, the useful question is **how far it got**. The last rung reached names the first faulty element.



One module per rung, in `modules/NN-name.sh`, played in numeric order. It never fixes anything: every verdict is a reading made at test time, with MSC counters taken as deltas.

```
cd /opt/GSM/tests
./banc-max.sh                       # default: = --dsp --restart --continue, ROM vocoder (io-handler l1phy)
BANC_VOCODEUR=hote ./banc-max.sh    # same, with gapk on the host instead of the ROM vocoder
./banc-max.sh --campagne            # grgsm then dsp (each restarted, up to voice) + PDF
./banc-max.sh --dsp --restart       # one mode, restart the stack in it
./banc-max.sh --grgsm               # one mode, on the running stack
./banc-max.sh --full                # both layer 1s, 4G and inter-operator, up to voice
./banc-max.sh --ms2 --data          # also the MS#1 -> MS#2 and GPRS rungs
./banc-max.sh --only=camp,attache   # only these rungs
./banc-max.sh --list                # the rungs, in order
```

Results land in `/root/banc-max-<date>/` (`verdict.txt`, `couverture.txt`, VTY captures, pytest reports, grafcets). Everything runs detached (`CALYPSO_NO_ATTACH=1`), never in an interactive tmux.

2.20.1.3 Run report: `tout-en-un.py`

`tout-en-un.py` gathers the `.log .sh .md .mmd .py .txt` of the tests and of the last `/root/banc-max-*` into **one Quarto/Markdown file**, shortened without loss: identical files are cited once, repeated lines become `×N`, and lines that differ only by numbers become a template plus the list of values. `.mmd` files are rendered as Mermaid diagrams.

```
./tout-en-un.py                              # /opt/GSM/tests + the last /root/banc-max-*
./tout-en-un.py /root/banc-max-20261016-163229  # a given run
SORTIE=/tmp/run.qmd EXCLURE="banc-max-*" ./tout-en-un.py
```

`qmd-worker/` serves these reports on the web (Cloudflare Worker).

2.20.1.4 pytest suites

`pytest/` reads the **live** stack (logs, VTY). The four suites are called by rungs 21–24, and each writes `test_results.md`, a grafcet and Mermaid diagrams:

file	suite
`test_10_infra.py`	processes, ports, VTY
`test_20_couche1_grgsm.py` / `test_21_couche1_dsp.py`	layer 1, per chain
`test_30_mobile.py`	the osmocom-bb mobile
`test_40_reseau.py`	network: LU, SMS, call
`test_50_actif.py`	active tests

```
cd /opt/GSM/tests/pytest && ./run_tests.sh
```

Other tools in the repository: `sms-rafale.sh` (SMS burst), `inject.py`, `log_timeline.py`, `rapport.py`, and `rom-tx/` (uplink encoding by the ROM, block injection).

2.20.1.5 `checks/`: diagnostics of a running stack

All read-only. `check_all.sh` chooses what makes sense on this node, runs from general to specific, and gives one verdict per check and an exit code.

```
sudo checks/check_all.sh           # everything relevant here
sudo checks/check_all.sh --dump    # + vty-debug-dump and its summary (daemons in DEBUG, several MB)
```

script	looks at
`global_check.sh`	every daemon: STP, HLR, MSC, BSC, BTS, PCU, SGSN, Asterisk, SMS relay
`ss7_check.sh`	signalling: ASPs, ASs, routes, inter-STP
`diag-stp-operator.sh`	the interconnect from an operator node: config vs running daemon, SCTP to the hub
`diag-interstp.sh`	the interconnect from the hub: who is attached, who is missing, PROHIB routes, flapping
`wan_ss7_check.sh`	what crosses between machines (only when `/etc/osmo-wan.conf` exists)
`annuaire.sh`	the subscriber directory, checked against what the network knows
`vty-debug-dump.sh` + `operator_summary.sh`	full VTY dump, then a per-operator summary

`wiki/Resultats.md` holds the reference `check_all.sh` runs (1 native operator; 3 operators with the inter-STP) and explains the known deviations.

Targeted diagnostics in `scripts/`:

```
scripts/call-diag.sh            # call path MS -> MSC -> MNCC -> sip-connector -> Asterisk
scripts/audio-diag.sh           # /dev/snd -> ALSA -> PulseAudio -> mobile l1phy
scripts/banc-repro.sh --calls 5 # a reproducible experiment for an intermittent fault
navigation/test-console.sh      # every command of the SS7 console, with a report
```

2.20.1.6 CI

The GitHub workflows (`.github/workflows/`) build the images and the ISO. They do not run the radio bench. The ladder needs a full stack with QEMU and is run on a real machine.

2.21 /opt/GSM/osmo-operator/doc/README-usage.md

2853 octets, 97 lignes

2.21.1 osmo-operator — using the bench

Day-to-day handling of a running bench: tmux, VTY, numbers, ports, Wireshark.

← README · Details: wiki/Home, wiki/Start-native, README-architecture

key	window

2.21.1.1 tmux

```
tmux -S /tmp/osmocom_tmux attach -t osmocom                      # native / ISO
docker exec -ti osmo-operator-1 tmux -S /tmp/osmocom_tmux attach -t osmocom   # Docker
```

key	window
`Ctrl-b 0`	faketrx
`Ctrl-b 1`	MS1 (trxcon + mobile)
`Ctrl-b 2`	Asterisk
`Ctrl-b 3`	SMSC (proto-smsc-daemon + relay)
`Ctrl-b w`	list all windows
`Ctrl-b d`	detach

2.21.1.2 VTY

port	component
`4239`	OsmoSTP
`4241`	OsmoBTS
`4242`	OsmoBSC
`4243`	OsmoMGW
`4254`	OsmoMSC
`4258`	OsmoHLR
`4247` / `4347`	osmocom-bb mobile (gr-gsm / DSP)

```
OsmoMSC# show subscriber all
OsmoMSC# subscriber msisdn 100101 sms sender msisdn 100102 send Hello

OsmoHLR# subscriber imsi 001010001000001 show
OsmoHLR# show gsup-clients
```

Subscribers are provisioned automatically at start-up by `run_modules/21-abonnes-hlr.sh`, following the numbering plan. `--regen` re-applies it.

2.21.1.3 Numbers

number	use
`<node>00<op><ms>` (e.g. `100101`)	GSM subscribers
`600`	echo test
`*#100#` / `*#101#`	USSD: own MSISDN / own IMSI
`<prefix><number>` (e.g. `2220011`)	a subscriber on another WAN node (README-ss7)

2.21.1.4 Ports on the wire

port	carries
`2905` SCTP	M3UA, local STP
`2908` SCTP	M3UA, inter-STP
`2427` UDP	MGCP (MGW)
`5060` UDP	SIP (Asterisk)
`2775` TCP	SMPP
`7890` TCP	inter-operator SMS relay

port	carries
`5700-5702` UDP	TRXD, between `osmo-bts-trx` and the bridge
`4729` UDP	GSMTAP, the Um interface
`4730` UDP	full SI set into the shunt (`si_bridge.py`)
`4731` UDP	SCH BSIC/FN out of the gr-gsm receiver
`6702` UDP	DL bursts into the DSP BSP (`--dsp`)
`8080` TCP	the web dashboard (`osmo-egprs-web`)

2.21.1.5 Wireshark

Started automatically with `sctp or udp port 4729`. Useful filters:

```
gsmtap                  # radio traffic (Um)
gsm_a.dtap              # MM / CC / SMS layer 3
gsm_map                 # MAP
m3ua / sccp             # SS7, multi-operator
```

A tcpdump wrapper is installed for GSMTAP (`udp/4729`) and M3UA captures. `tools/osmo-wireshark-root.sh` opens Wireshark with capture rights.

2.21.1.6 Sending things by hand

```
scripts/send-mt-sms.sh              # an MT SMS to a mobile
tools/at-cmd.py                     # AT commands to the emulated phone's modem
```

2.21.1.7 Dashboard

`osmo-egprs-web` on `:8080` (`systemctl restart osmo-egprs-web`). The desktop also runs a conky status panel (`tools/conky-osmo-status.sh`).

2.22 /opt/GSM/osmo-operator/doc/c54x_exe-depuis-ubuntu-vierge.md

9241 octets, 171 lignes

2.22.1 c54x_exe depuis une Ubuntu 24.04 vierge

Oct 20, 2026 · Bastien Baranoff

2.22.1.1 Ce que montre cette page

En une vingtaine de minutes, le cœur TMS320C54x émulé de `c54x_exe` charge la mask-ROM TI d'un vrai Calypso et l'exécute, sans QEMU, sans ARM, sans réseau GSM.

C'est le hello world du DSP : il prouve que le cœur exécute du vrai code TI. Il ne prouve pas que le DSP décode quoi que ce soit, puisque aucun burst n'est injecté. Le montage complet (ARM sous QEMU, BTS, pont, appel voix) est décrit en fin de page et vit dans osmo-operator.

Tout se fait en root ou avec `sudo` ; les chemins `/opt/GSM/…` sont ceux que les Makefile et les scripts attendent par défaut.

2.22.1.2 1. Prérequis (Ubuntu 24.04)

Une ligne `apt` suffit : le compilateur, `git`, `curl` pour les ROM, et `libosmocore` (qui contient `libosmocoding`, utilisée par `calypso_bsp.c`).

```
sudo apt update
sudo apt install -y build-essential git curl pkg-config libosmocore-dev
sudo mkdir -p /opt/GSM
```

Vérifier que `libosmocoding` est bien là :

```
pkg-config --modversion libosmocoding
```

Si `pkg-config` ne la trouve pas, la version de `libosmocore` du dépôt Ubuntu est trop ancienne ou incomplète ; installer alors `libosmocore-dev` depuis le dépôt nightly d'Osmocom (voir osmocom.org).

2.22.1.3 2. Les sources de qosmo

`c54x_exe` ne recopie pas le cœur C54x : il compile directement les sources de qosmo (`calypso_c54x.c` et ses voisins) avec ~120 lignes de doublures d'en-têtes QEMU. Il faut donc le dépôt sur disque, mais pas le construire.

```
cd /opt/GSM
git clone https://github.com/bbaranoff/qosmo
```

Aucun `configure`, aucun `make` ici. Si qosmo est ailleurs que `/opt/GSM/qosmo`, passer `QOSMO=/chemin` au `make` de l'étape suivante.

2.22.1.4 3. Cloner et compiler c54x_exe

La compilation prend moins d'une minute et produit un seul binaire, `c54x_exe`, à la racine du dépôt.

```
cd /opt/GSM
git clone https://github.com/bbaranoff/c54x_exe
cd c54x_exe
make QOSMO=/opt/GSM/qosmo \
     CFLAGS="-O3 -g -Wall -Werror=format -Werror=format-extra-args \
             -Wno-unused-function -Wno-unused-variable \
             -Wno-unused-but-set-variable -Wno-sign-compare"
```

Ce `CFLAGS` est celui du Dockerfile d'osmo-operator : il garde les erreurs de format bloquantes et rend muets les avertissements de variables inutilisées, nombreux dans les sources QEMU reprises telles quelles. Un `make` nu marche aussi mais sort beaucoup plus de bruit.

`make` reconstruit le binaire à chaque appel (cible `.PHONY`) ; pas besoin de `make clean` après un changement dans qosmo.

Vérifier :

```
./c54x_exe --help | head
```

2.22.1.5 4. Les ROM DSP, depuis FreeCalypso

La mask-ROM du C54x n'est pas dans le dépôt : elle vient du dump publié par le projet FreeCalypso (Mychaela Falconia), sur son FTP public. Le script `rom/fetch-rom.sh` la télécharge et la découpe en 7 sections au format attendu par `--rom-dir`.

```
cd /opt/GSM/c54x_exe
export FREECALYPSO_URL=ftp://ftp.freecalypso.org/pub/GSM/Calypso
bash rom/fetch-rom.sh --dest /opt/GSM --dest /opt/GSM/c54x_exe/rom
curl -fsS --retry 2 -o /opt/GSM/calypso_dsp.txt "$FREECALYPSO_URL/dsp-rom-3606-dump.txt"
cp /opt/GSM/calypso_dsp.txt rom/calypso_dsp.txt
```

`3606` est la version de ROM DSP du Calypso des Motorola C1xx ; c'est aussi la valeur que la L1 DSP écrit dans l'API RAM au boot, ce qui permettra de vérifier plus loin que la bonne ROM tourne.

Après le script, `/opt/GSM` doit contenir exactement ces 7 fichiers :

```
ls -l /opt/GSM/calypso_dsp.*.bin
# calypso_dsp.PROM0.bin  calypso_dsp.PROM1.bin  calypso_dsp.PROM2.bin
# calypso_dsp.PROM3.bin  calypso_dsp.DROM.bin   calypso_dsp.PDROM.bin
# calypso_dsp.Registers.bin
```

Si `curl` refuse le FTP (certaines builds le compilent sans), le dump se télécharge à la main depuis le même répertoire sur ftp.freecalypso.org et se passe à `fetch-rom.sh` en local.

2.22.1.6 5. Premier run : la ROM tourne

```
cd /opt/GSM/c54x_exe
./c54x_exe --trames 200
```

`--trames 200` fait tourner le DSP sur 200 trames TDMA (une trame = 4,615 ms de temps GSM), avec le budget d'instructions par défaut de 2300 par trame en mode autonome. Sans option, seules les erreurs du cœur passent sur stderr ; le bilan final dit combien de lignes ont été masquées et avec quel niveau `-v` les voir.

Ce qu'on doit voir, dans l'ordre :

1. Les 7 sections de ROM chargées depuis `/opt/GSM` (ou `--rom-dir`).
2. `c54x_reset()` : la ROM repart de `0xff80`, son vecteur de reset.
3. Des lignes `BRANCH-TRACE` autour de `PC=0xb41f` : c'est la boucle d'attente de la ROM TI, du vrai code que le cœur exécute.
4. Un bilan de fin avec `a_sch[3] = 0x771a`.

Pour voir une ligne par trame :

```
./c54x_exe --trames 50 --verbeux
```

Et pour les traces du cœur, `-v` à `-vvvvvv` (voir `--help`).

2.22.1.7 6. Lire le résultat

`a_sch[3] = 0x771a` est une valeur constante d'un run à l'autre : la ROM a booté, s'est parquée en IDLE et a laissé cet état dans sa fenêtre API. C'est la preuve que le cœur exécute fidèlement la ROM jusque-là.

Ce que ça ne prouve pas : le décodage. L'API RAM est vierge, aucun burst n'arrive, le DSP tourne à vide. Un décodeur dont la sortie ne dépend pas de l'entrée ne décode pas ; faire varier l'entrée demande le montage complet de la section 8.

Point de repère utile : le README de `qemu-calypso` donne `0xf8d8` pour ce même mot. La différence vient de l'entrée (QEMU fournit une API RAM initialisée par l'ARM, ici elle est vide), pas d'un bug : deux entrées différentes, deux constantes différentes.

2.22.1.8 7. Erreurs classiques

Symptôme	Cause	Remède
`make` : `qemu/osdep.h` ou `calypso_c54x.c` introuvable	qosmo n'est pas à `/opt/GSM/qosmo`	`make QOSMO=/chemin/vers/qosmo`
`make` : `osmocom/coding/...` introuvable	`libosmocore-dev` absent ou trop vieux	section 1, dépôt nightly Osmocom
`fetch-rom.sh` échoue ou `curl: (1) Protocol "ftp" not supported`	`curl` compilé sans FTP, ou pare-feu	télécharger à la main, puis `fetch-rom.sh` sur le fichier local
`c54x_exe` : ROM introuvable	les `.bin` ne sont pas dans `/opt/GSM`	`--rom-dir /chemin` ou refaire la section 4
Le bilan ne montre pas `0x771a`	mauvaise ROM, ou sections mal découpées	comparer `rom/calypso_dsp.txt` au dump 3606 ; refaire la section 4
`PC=0xb41f` n'apparaît jamais, le cœur sort en erreur	régression dans qosmo	`cd /opt/GSM/qosmo && git log -3`, remonter au commit du README

2.22.1.9 8. La suite : injecter des bursts

Pour que le DSP décode vraiment, il faut l'ARM (le firmware `layer1.highram.elf` d'osmocom-bb sous QEMU `qosmo`), une BTS et le pont qui lui livre des bursts. Deux chemins :

Le banc complet. Il pose tout, dans les mêmes chemins `/opt/GSM/…` que cette page :

```
cd /opt/GSM
git clone https://github.com/bbaranoff/osmo-operator
cd osmo-operator
sudo ./install.sh
sudo ./start-direct.sh --dsp
```

Le `--dsp` remplace la couche 1 gr-gsm par `c54x_exe` ; `--menu` pose la même question interactivement. Le wiki d'osmo-operator décrit les options, les ports et le plan de numérotation.

Exe par exe. Pour voir chaque étape produire son log : le DSP (`c54x_exe --arm -v`), puis QEMU avec `CALYPSO_DSP_EXTERN=1`, puis `osmocon`, puis `mobile`. La séquence, avec les lignes de journal attendues à chaque pas, est dans LAUNCH.md et le README de c54x_exe.

À partir de là, `a_sch` change avec l'entrée, et la question « est-ce que ça décode ? » a une réponse mesurable.

2.22.1.10 9. Le firmware Calypso : la toolchain arm-elf (gnuarm)

Le firmware d'osmocom-bb (`layer1.highram.elf`, `trx.highram.elf`…) se compile avec la toolchain historique `arm-elf` (gcc 4.5.2, binutils 2.21.1, newlib 1.19) du gnu-arm-installer, installée dans `/root/gnuarm/install`. Sur Ubuntu 24.04 son `cc1` réclame `libmpfr.so.4`, que le système ne fournit plus (il a `libmpfr.so.6`) ; un lien suffit :

```
ln -sf /usr/lib/x86_64-linux-gnu/libmpfr.so.6 /usr/local/lib/libmpfr.so.4 && ldconfig
/root/gnuarm/install/bin/arm-elf-gcc --version      # gcc 4.5.2
```

L'arbre `osmocom-bb-transceiver` (branche jolly/testing, appli `trx`) se compile aussi avec le `arm-none-eabi-gcc` du système, avec les adaptations de `patches/osmocom-bb-transceiver-qemu-bench.patch` (préfixe, édition de liens par gcc, `-fcommon`, `solve_envs.py` en python 3) :

```
cd /opt/GSM/osmocom-bb-transceiver/src/target/firmware
make BOARDS=compal_e88 APPLICATIONS=trx ENVIRONMENTS=highram CROSS_COMPILE=arm-none-eabi-
cp board/compal_e88/trx.highram.{elf,bin} /opt/GSM/firmware/board/compal_e88/
```

Tous les patchs appliqués aux arbres osmocom-bb (mobile, transceiver, burst-ind) sont dans `patches/` d'osmo-operator et appliqués par le Dockerfile et `install_modules/40-patches.sh`.

2.23 /opt/GSM/osmo-operator/doc/c54x_exe-from-scratch-ubuntu.en.md

8836 octets, 171 lignes

2.23.1 c54x_exe from a clean Ubuntu 24.04

Oct 20, 2026 · Bastien Baranoff

2.23.1.1 What this page shows

In about twenty minutes, the emulated TMS320C54x core of `c54x_exe` loads the TI mask ROM of a real Calypso and runs it, without QEMU, without the ARM, without a GSM network.

This is the DSP's hello world: it proves the core executes real TI code. It does not prove the DSP decodes anything, since no burst is injected. The full setup (ARM under QEMU, BTS, bridge, voice call) is described at the end of the page and lives in osmo-operator.

Everything runs as root or with `sudo`; the `/opt/GSM/…` paths are the ones the Makefiles and scripts expect by default.

2.23.1.2 1. Prerequisites (Ubuntu 24.04)

One `apt` line is enough: the compiler, `git`, `curl` for the ROMs, and `libosmocore` (which contains `libosmocoding`, used by `calypso_bsp.c`).

```
sudo apt update
sudo apt install -y build-essential git curl pkg-config libosmocore-dev
sudo mkdir -p /opt/GSM
```

Check that `libosmocoding` is there:

```
pkg-config --modversion libosmocoding
```

If `pkg-config` cannot find it, the `libosmocore` of the Ubuntu repository is too old or incomplete; install `libosmocore-dev` from the Osmocom nightly repository instead (see osmocom.org).

2.23.1.3 2. The qosmo sources

`c54x_exe` does not copy the C54x core: it compiles the qosmo sources directly (`calypso_c54x.c` and its neighbours) with ~120 lines of QEMU header stand-ins. The repository must be on disk, but it does not need to be built.

```
cd /opt/GSM
git clone https://github.com/bbaranoff/qosmo
```

No `configure`, no `make` here. If qosmo is somewhere other than `/opt/GSM/qosmo`, pass `QOSMO=/path` to the `make` of the next step.

2.23.1.4 3. Clone and build c54x_exe

The build takes less than a minute and produces a single binary, `c54x_exe`, at the root of the repository.

```
cd /opt/GSM
git clone https://github.com/bbaranoff/c54x_exe
cd c54x_exe
make QOSMO=/opt/GSM/qosmo \
     CFLAGS="-O3 -g -Wall -Werror=format -Werror=format-extra-args \
             -Wno-unused-function -Wno-unused-variable \
             -Wno-unused-but-set-variable -Wno-sign-compare"
```

This `CFLAGS` is the one of the osmo-operator Dockerfile: it keeps format errors fatal and silences the unused-variable warnings, numerous in the QEMU sources taken as they are. A bare `make` works too but is much noisier.

`make` rebuilds the binary on every call (`.PHONY` target); no `make clean` is needed after a change in qosmo.

Check:

```
./c54x_exe --help | head
```

2.23.1.5 4. The DSP ROMs, from FreeCalypso

The C54x mask ROM is not in the repository: it comes from the dump published by the FreeCalypso project (Mychaela Falconia) on its public FTP. The `rom/fetch-rom.sh` script downloads it and splits it into the 7 sections `--rom-dir` expects.

```
cd /opt/GSM/c54x_exe
export FREECALYPSO_URL=ftp://ftp.freecalypso.org/pub/GSM/Calypso
bash rom/fetch-rom.sh --dest /opt/GSM --dest /opt/GSM/c54x_exe/rom
curl -fsS --retry 2 -o /opt/GSM/calypso_dsp.txt "$FREECALYPSO_URL/dsp-rom-3606-dump.txt"
cp /opt/GSM/calypso_dsp.txt rom/calypso_dsp.txt
```

`3606` is the DSP ROM version of the Calypso in the Motorola C1xx; it is also the value the DSP L1 writes into the API RAM at boot, which will later confirm that the right ROM is running.

After the script, `/opt/GSM` must contain exactly these 7 files:

```
ls -l /opt/GSM/calypso_dsp.*.bin
# calypso_dsp.PROM0.bin  calypso_dsp.PROM1.bin  calypso_dsp.PROM2.bin
# calypso_dsp.PROM3.bin  calypso_dsp.DROM.bin   calypso_dsp.PDROM.bin
# calypso_dsp.Registers.bin
```

If `curl` refuses FTP (some builds come without it), download the dump by hand from the same directory on ftp.freecalypso.org and give the local file to `fetch-rom.sh`.

2.23.1.6 5. First run: the ROM runs

```
cd /opt/GSM/c54x_exe
./c54x_exe --trames 200
```

`--trames 200` runs the DSP for 200 TDMA frames (one frame = 4.615 ms of GSM time), with the default budget of 2300 instructions per frame in standalone mode. Without options only core errors go to stderr; the final summary says how many lines were hidden and which `-v` level shows them.

What you should see, in order:

1. The 7 ROM sections loaded from `/opt/GSM` (or `--rom-dir`).
2. `c54x_reset()`: the ROM restarts at `0xff80`, its reset vector.
3. `BRANCH-TRACE` lines around `PC=0xb41f`: the TI ROM's wait loop, real code executed by the core.
4. A final summary with `a_sch[3] = 0x771a`.

For one line per frame:

```
./c54x_exe --trames 50 --verbeux
```

And for core traces, `-v` to `-vvvvvv` (see `--help`).

2.23.1.7 6. Reading the result

`a_sch[3] = 0x771a` is a constant from one run to the next: the ROM booted, parked itself in IDLE and left this state in its API window. It is the proof that the core executes the ROM faithfully up to that point.

What it does not prove: decoding. The API RAM is blank, no burst arrives, the DSP runs idle. A decoder whose output does not depend on its input does not decode; varying the input requires the full setup of section 8.

Useful reference: the `qemu-calypso` README gives `0xf8d8` for the same word. The difference comes from the input (QEMU provides an API RAM initialised by the ARM, here it is empty), not from a bug: two different inputs, two different constants.

2.23.1.8 7. Common errors

Symptom	Cause	Fix
`make`: `qemu/osdep.h` or `calypso_c54x.c` not found	qosmo is not at `/opt/GSM/qosmo`	`make QOSMO=/path/to/qosmo`
`make`: `osmocom/coding/...` not found	`libosmocore-dev` missing or too old	section 1, Osmocom nightly repository
`fetch-rom.sh` fails or `curl: (1) Protocol "ftp" not supported`	`curl` built without FTP, or a firewall	download by hand, then `fetch-rom.sh` on the local file
`c54x_exe`: ROM not found	the `.bin` files are not in `/opt/GSM`	`--rom-dir /path` or redo section 4
The summary does not show `0x771a`	wrong ROM, or badly split sections	compare `rom/calypso_dsp.txt` with the 3606 dump; redo section 4
`PC=0xb41f` never appears, the core exits with an error	regression in qosmo	`cd /opt/GSM/qosmo && git log -3`, go back to the README's commit

2.23.1.9 8. Next: injecting bursts

For the DSP to really decode, you need the ARM (the osmocom-bb `layer1.highram.elf` firmware under the `qosmo` QEMU), a BTS and the bridge that delivers bursts to it. Two paths:

The complete bench. It installs everything, in the same `/opt/GSM/…` paths as this page:

```
cd /opt/GSM
git clone https://github.com/bbaranoff/osmo-operator
cd osmo-operator
sudo ./install.sh
sudo ./start-direct.sh --dsp
```

`--dsp` replaces the gr-gsm layer 1 with `c54x_exe`; `--menu` asks the same question interactively. The osmo-operator wiki describes the options, the ports and the numbering plan.

Executable by executable. To watch each step produce its log: the DSP (`c54x_exe --arm -v`), then QEMU with `CALYPSO_DSP_EXTERN=1`, then `osmocon`, then `mobile`. The sequence, with the expected log lines at each step, is in LAUNCH.md and the c54x_exe README.

From there, `a_sch` changes with the input, and the question "does it decode?" has a measurable answer.

2.23.1.10 9. The Calypso firmware: the arm-elf toolchain (gnuarm)

The osmocom-bb firmware (`layer1.highram.elf`, `trx.highram.elf`…) is built with the historical `arm-elf` toolchain (gcc 4.5.2, binutils 2.21.1, newlib 1.19) from the gnu-arm-installer, installed in `/root/gnuarm/install`. On Ubuntu 24.04 its `cc1` asks for `libmpfr.so.4`, which the system no longer ships (it has `libmpfr.so.6`); a symlink is enough:

```
ln -sf /usr/lib/x86_64-linux-gnu/libmpfr.so.6 /usr/local/lib/libmpfr.so.4 && ldconfig
/root/gnuarm/install/bin/arm-elf-gcc --version      # gcc 4.5.2
```

The `osmocom-bb-transceiver` tree (jolly/testing branch, `trx` application) also builds with the system's `arm-none-eabi-gcc`, with the adaptations of `patches/osmocom-bb-transceiver-qemu-bench.patch` (prefix, linking through gcc, `-fcommon`, `solve_envs.py` ported to Python 3):

```
cd /opt/GSM/osmocom-bb-transceiver/src/target/firmware
make BOARDS=compal_e88 APPLICATIONS=trx ENVIRONMENTS=highram CROSS_COMPILE=arm-none-eabi-
cp board/compal_e88/trx.highram.{elf,bin} /opt/GSM/firmware/board/compal_e88/
```

Every patch applied to the osmocom-bb trees (mobile, transceiver, burst-ind) lives in `patches/` of osmo-operator and is applied by the Dockerfile and `install_modules/40-patches.sh`.

2.24 /opt/GSM/osmo-operator/doc/install-from-scratch.en-notes.md

4257 octets, 75 lignes

2.24.1 English notes per Dockerfile stage (read by tools/dockerfile-to-install-md.sh –lang en)

One section per stage name (`## <stage>`); the text is inserted under the stage heading of the English install guide. The Dockerfile comments themselves are in French and are reproduced only in the French guide.

2.24.1.1 base

Ubuntu 24.04 (noble) and everything that is long to install and rarely changes: the whole apt package list (one list for the entire project, `apt-fast` to download it), Wireshark allowed for non-root capture, the `osmo-deb` helper that caches every compiled tree as a `.deb` in `/var/cache/osmo-debs`, the Asterisk and ALSA configuration files, the `osmocom` system user and the `/opt/GSM` tree that every script expects. On a bare machine the `.deb` cache is empty: every `osmo-deb install` fails and the build branch runs.

2.24.1.2 osmo-core

The Osmocom stack compiled from source, in dependency order and at pinned versions (libosmocore, libosmo-netif, libosmo-abis, libosmo-sigtran, libsmpp34, libgtpnl, osmo-hlr, osmo-mgw, osmo-ggsn, osmo-sgsn, osmo-msc, osmo-bsc, osmo-trx, osmo-bts, osmo-pcu, osmo-sip-connector, libosmo-gprs), then the project patches kept in `patches/`: osmo-trx IPC device aligned on the TDMA frame and a per-RA RACH modulation table, and the "force RAND" toy patches for osmo-bts, osmo-hlr and osmo-msc (the RAND of the authentication challenge is forced only when the `RAND` environment variable is set). The stage ends with osmo-gapk (voice codecs), the GSUP SMSC protocol library, the SMS coding utilities and libosmo-dsp.

2.24.1.3 bb

osmocom-bb (the `mobile` application, `osmocon`, layer23) with the patches from `patches/osmocom-bb-*` (clock generator frame tolerance, PLMN handling, SIM PLMN from BCD), the Calypso firmware images (`layer1.highram.elf` and friends) and the two gcc flavours the old trees need (gcc-9 selected through `update-alternatives` while building, gcc-11 restored afterwards). It also builds the `osmocom-bb-transceiver` tree (Sylvain Munaut's jolly/testing branch: the `trx` firmware application and the host `transceiver` program that turns the phone into a BTS transceiver) with `patches/osmocom-bb-transceiver-qemu-bench.patch`, and the `burst-ind` tree.

2.24.1.4 qemu

The single QEMU tree of the project, `/opt/GSM/qosmo`: a QEMU fork with the Calypso machine (ARM core, TPU/TSP, DSP API RAM, LCD and keypad emulation) built with `--enable-l1-grgsm`, the gr-gsm layer 1 shunt. The C54x DSP itself is no longer run inside QEMU: see the `l1` stage.

2.24.1.5 l1

The two layer-1 engines that run outside QEMU: `c54x_exe` (the TMS320C54x core emulated from the qosmo sources, loading the real TI mask ROM, see the appendix) and `grgsm_exe` (the gr-gsm based layer 1). The DSP ROM images come from the FreeCalypso dump and are split into the seven sections `c54x_exe` reads.

2.24.1.6 grgsm-venv

GNU Radio 3.10, gr-osmosdr and gr-gsm installed in the Python virtual environment `/root/.env`, with the project patches for gr-gsm (optional Qt GUI import, receiver publishing BSIC and frame number). The venv is activated from root's profile at the end of the final stage.

2.24.1.7 node

Node 22 and the web dashboard `osmo-egprs-web` (the operations console served as a systemd unit), plus its npm dependencies.

2.24.1.8 lte

The 4G side of the bench: srsRAN_4G built with the ZeroMQ radio (no SDR needed) and Open5GS, both under `/opt/LTE`, with their configuration files.

2.24.1.9 osmocom-nitb

The assembly stage, the image that is actually published. Nothing is compiled here: the `.deb` packages produced by the previous stages are installed (`dpkg -i`), then the project files are laid out: the `/etc/osmocom` configurations, the scripts under `/usr/local/bin`, the systemd units, the PulseAudio system daemon configuration (GSM audio), the runtime directories, the shell prompt, the desktop pieces and the `/opt/GSM` helper scripts. On a single machine the `COPY --from=<stage>` instructions are no-ops: everything is already installed where the previous stages put it. The container entrypoint and stop signal are Docker metadata only.

2.24.1.10 debs

A `scratch` stage holding only `/var/cache/osmo-debs`, used by the CI to extract the compiled packages without materialising the full image. Nothing to do on a machine.

2.25 /opt/GSM/osmo-operator/doc/install-from-scratch.en.md

46532 octets, 1144 lignes

2.25.1 Installing from scratch (generated from the Dockerfile)

> Generated by `tools/dockerfile-to-install-md.sh --lang en` on 2026-10-20 from `Dockerfile`. Do not edit by hand: fix the Dockerfile, the script or `doc/install-from-scratch.en-notes.md`, then regenerate. The French version (`doc/install-from-scratch.md`) carries the full rationale written in the Dockerfile comments; this one keeps the commands and a short English note per stage.

2.25.1.1 How to read this page

The osmo-operator Dockerfile is the complete, tested description of the installation. This page replays it step by step on a bare machine (Ubuntu 24.04, as root), without Docker:

- every `FROM ... AS <name>` of the Dockerfile is a chapter; on a single machine the stages simply follow each other and the `COPY --from=<stage>` instructions are not needed (everything is already in place);
- every `RUN` is a bash block to paste as is into a root shell, in order;
- every `ARG`/`ENV` becomes an `export` to set before the blocks that follow (in Docker an `ARG` does not survive a `FROM`; here it just stays in the shell);
- every `COPY` becomes a `cp` from a checkout of the osmo-operator repository (`$REPO`, `/opt/GSM/osmo-operator` by default).

The `.deb` package cache (`osmo-deb install <name> <version>`) is an optimisation of the Docker build: without the cache the command fails and the `if ! osmo-deb install ...; then ... fi` branch builds from source, which is exactly what a bare machine needs. `osmo-deb pack` then wraps the result (`make install`); without `/var/cache/osmo-debs` it just installs.

```
export REPO=/opt/GSM/osmo-operator      # checkout of the repository (git clone
↪  https://github.com/bbaranoff/osmo-operator)
export ROOT=/opt/GSM
mkdir -p "$ROOT"
```

2.25.1.2 Table of contents

1. base (from `ubuntu:24.04`)
2. osmo-core (from `base`)
3. bb (from `osmo-core`)
4. qemu (from `base`)
5. l1 (from `qemu`)
6. grgsm-venv (from `osmo-core`)
7. node (from `base`)
8. lte (from `base`)
9. osmocom-nitb (from `base`)
10. debs (from `scratch`)

2.25.1.3 Stage 1 : base

Ubuntu 24.04 (noble) and everything that is long to install and rarely changes: the whole apt package list (one list for the entire project, `apt-fast` to download it), Wireshark allowed for non-root capture, the `osmo-deb` helper that caches every compiled tree as a `.deb` in `/var/cache/osmo-debs`, the Asterisk and ALSA configuration files, the `osmocom` system user and the `/opt/GSM` tree that every script expects. On a bare machine the `.deb` cache is empty: every `osmo-deb install` fails and the build branch runs.

In Docker: `FROM ubuntu:24.04`. On a machine, carry on in the same system.

```
export DEBIAN_FRONTEND=noninteractive
```

```
export ROOT=/opt/GSM
```

```
export container=docker
export OSMO_IMAGE=1
export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig
export LD_LIBRARY_PATH=/usr/local/lib
export GIT_TERMINAL_PROMPT=0
```

```
mkdir -p "$(dirname /usr/local/sbin/apt-fast-install)" && cp -r "$REPO"/packaging/apt-fast-install.sh
↪  /usr/local/sbin/apt-fast-install
```

```
export OSMO_UBUNTU_MIRROR=""


chmod 755 /usr/local/sbin/apt-fast-install && apt-fast-install \
    && rm -rf /var/lib/apt/lists/*


export DEBIAN_FRONTEND=noninteractive


echo 'wireshark-common wireshark-common/install-setuid boolean true' | debconf-set-selections


if [ -f /etc/apt/sources.list.d/ubuntu.sources ]; then \
        sed -i 's/^Types: deb$/Types: deb deb-src/' /etc/apt/sources.list.d/ubuntu.sources; \
    else \
        sed -nE 's|^deb (http\S+) (\S+) .*|deb-src \1 \2 main restricted universe multiverse|p' \
            /etc/apt/sources.list | sort -u > /etc/apt/sources.list.d/deb-src.list; \
    fi


apt-fast update && apt-fast install -y --no-install-recommends \
    # Outils de build
    build-essential git gcc g++ make cmake autoconf automake libtool pkg-config wget curl \
    # Dépendances Osmocom Core & Network
    libtalloc-dev libpcsclite-dev libsctp-dev libmnl-dev liburing-dev asterisk-moh* \
    libdbi-dev libdbd-sqlite3 libsqlite3-dev sqlite3 libc-ares-dev libgnutls28-dev \
    # Audio, Radio & SIP
    libortp-dev libfftw3-dev libusb-1.0-0-dev libsofia-sip-ua-dev libsofia-sip-ua-glib-dev \
    # Python & Outils système
    python3 python3-dev python3-scapy ca-certificates tmux systemd systemd-sysv \
    # Debug — gdb-multiarch pour attacher au gdb-stub QEMU (ARM Calypso)
    gdb-multiarch \
    # ALSA — requis par osmo-gapk pour l'I/O audio matériel
    # (libasound2t64 : nom noble de libasound2, transition time_t 64 bits)
    libasound2-dev libasound2t64 alsa-utils \
    # libgsm — codec GSM-FR natif (accélère gapk en mode gsmfr)
    libgsm1-dev libgsm1 \
    iptables iproute2 asterisk ffmpeg \
    # Sync build-iso : psmisc (pkill/killall, cleanup) + pulseaudio (chaîne audio gapk/parec)
    psmisc pulseaudio pulseaudio-utils binutils-arm-none-eabi \
    # Toolchains alternatives : osmocom-bb jolly/testing et fixeria/burst_ind ne
    # compilent qu'avec gcc-9 ; gcc-11 reste le compilateur par defaut du reste
    # (gcc-13 est celui de noble : il compile la pile Osmocom en tete de fichier,
    # les alternatives ci-dessous ne sont posees qu apres).
    # (Ex-bloc apt dedie, supprime : les `update-alternatives` plus bas echouaient
    #  « alternative path /usr/bin/gcc-9 doesn't exist » sans ces paquets.)
    gcc-9 g++-9 gcc-11 g++-11 \
    # log4cpp : etait installe a part avec le build-dep gnuradio (gr-gsm).
    liblog4cpp5-dev \
    # QEMU Calypso (bbaranoff/qosmO) : venv + numpy/scipy pour l'outillage
    # DSP, glib/pixman/slirp pour la cible arm-softmmu, ninja pour meson, socat
    # pour les PTY de scripts/run.sh. (Ex-bloc apt-get juste avant le build QEMU.)
    python3-venv python3-pip python3-numpy python3-scipy \
    libglib2.0-dev libpixman-1-dev libslirp-dev socat ninja-build \
    # Ex-Dockerfile.run — outillage d'exploitation du conteneur :
    #   telnet   -> repli VTY (run_modules/21-abonnes-hlr.sh) + transport dashboard
    #   whiptail -> menus de tools/vty-menu.sh et de start-docker.sh
    #   xz-utils -> `tar -xJf` du tarball Node 22 (bloc dashboard, en fin de fichier)
    #   libasound2-plugins -> greffon ALSA->PulseAudio, EXIGE par configs/asound.conf
    #        (`type pulse` sur gsm_out, gsm_in ET pcm.!default) : sans lui, toute
    #        ouverture ALSA (gapk, mobile) echoue et l'audio est mort des 2 cotes
    #   tshark/tcpdump/libcap2-bin -> capture ; libcap2-bin fournit le setcap ci-dessous
    telnet nano whiptail xz-utils libasound2-plugins \
    tcpdump tshark libcap2-bin \
    # Deps GNU Radio 3.10 (gr-gsm, venv /root/.env plus bas) : meme resolution
    # apt que le reste, pas de second RUN apt. (Le && ferme la liste pour
```

```
    # l extracteur de install_modules/10-deps.sh.)
    && apt-fast build-dep -y gnuradio \
    && rm -rf /var/lib/apt/lists/*


setcap cap_net_raw,cap_net_admin+eip "$(command -v dumpcap)"


export OSMO_DEB_REFRESH=0


mkdir -p "$(dirname /usr/local/sbin/osmo-deb)" && cp -r "$REPO"/packaging/osmo-deb.sh /usr/local/sbin/osmo-deb


mkdir -p /var/cache/osmo-debs/ && cp -r "$REPO"/.deb-cache/ /var/cache/osmo-debs/


chmod 755 /usr/local/sbin/osmo-deb && osmo-deb list || true


git config --global http.postBuffer 524288000
```

Docker metadata, no equivalent on a machine: `SHELL ["/bin/bash", "-c"]`.

```
mkdir -p /etc/asterisk/ && cp -r "$REPO"/configs/*conf /etc/asterisk/


mkdir -p "$(dirname /etc/asound.conf)" && cp -r "$REPO"/configs/asound.conf /etc/asound.conf


cd ${ROOT}


groupadd osmocom && useradd -r -g osmocom -s /sbin/nologin -d /var/lib/osmocom osmocom && \
    mkdir -p /var/lib/osmocom && chown osmocom:osmocom /var/lib/osmocom
```

2.25.1.4 Stage 2 : osmo-core

The Osmocom stack compiled from source, in dependency order and at pinned versions (libosmocore, libosmo-netif, libosmo-abis, libosmo-sigtran, libsmpp34, libgtpnl, osmo-hlr, osmo-mgw, osmo-ggsn, osmo-sgsn, osmo-msc, osmo-bsc, osmo-trx, osmo-bts, osmo-pcu, osmo-sip-connector, libosmo-gprs), then the project patches kept in `patches/`: osmo-trx IPC device aligned on the TDMA frame and a per-RA RACH modulation table, and the "force RAND" toy patches for osmo-bts, osmo-hlr and osmo-msc (the RAND of the authentication challenge is forced only when the `RAND` environment variable is set). The stage ends with osmo-gapk (voice codecs), the GSUP SMSC protocol library, the SMS coding utilities and libosmo-dsp.

In Docker: `FROM base`. *On a machine, carry on in the same system.*

```
export ROOT=/opt/GSM


export OSMO_DEB_REFRESH=0
```

```
for repo in \
    libosmocore:1.12.1 \
    libosmo-netif:1.7.0 \
    libosmo-abis:2.1.0 \
    libosmo-sigtran:2.2.1 \
    libsmpp34:1.14.5 \
    libgtpnl:1.3.3 \
    osmo-hlr:1.9.2 \
    osmo-mgw:1.15.0 \
    osmo-ggsn:1.14.0 \
    osmo-sgsn:1.13.1 \
    osmo-msc:1.15.0 \
    osmo-bsc:1.14.0 \
    osmo-trx:1.7.2 \
    osmo-bts:1.10.0 \
    osmo-pcu:1.5.2 \
    osmo-sip-connector:1.7.2 \
    libosmo-gprs:0.2.1; \
    do \
    name=$(echo $repo | cut -d: -f1) && \
    version=$(echo $repo | cut -d: -f2) && \
    \
    if [[ "$name" =~ ^libosmo ]]; then \
        GIT_URL="https://gitea.osmocom.org/osmocom/$name"; \
    else \
        GIT_URL="https://gitea.osmocom.org/cellular-infrastructure/$name"; \
    fi && \
    \
    # Le paquet du cache d abord : s il est la, ni clone ni compilation.
    if osmo-deb install "$name" "$version"; then continue; fi && \
    \
    cd ${ROOT} && \
    git clone "$GIT_URL" && cd "$name" && \
    git checkout "$version" && \
    \
    autoreconf -fi && \
    EXTRA_FLAGS="" && \
    if [ "$name" = "libosmo-abis" ]; then EXTRA_FLAGS="--disable-dahdi"; fi && \
    if [ "$name" = "osmo-msc" ]; then EXTRA_FLAGS="--enable-smpp"; fi && \
    if [ "$name" = "osmo-mgw" ]; then EXTRA_FLAGS="--enable-alsa"; fi && \
    if [ "$name" = "osmo-trx" ]; then EXTRA_FLAGS="--with-ipc"; fi && \
    if [ "$name" = "osmo-bts" ]; then EXTRA_FLAGS="--enable-virtual --enable-trx"; fi && \
    if [ "$name" = "osmo-ggsn" ]; then EXTRA_FLAGS="--enable-gtp-linux"; fi && \
    \
    ./configure $EXTRA_FLAGS && \
    make -j$(nproc) && \
    # make install sous DESTDIR -> .deb dans le cache -> dpkg -i dans la racine
    osmo-deb pack "$name" "$version" make install && \
    ldconfig \
    || { echo "ECHEC build osmocom: $name"; exit 1; }; \
    done


mkdir -p "$(dirname /tmp/osmo-trx-ipc-ts-frame-align.patch)" && cp -r
 ↪  "$REPO"/patches/osmo-trx-ipc-ts-frame-align.patch /tmp/osmo-trx-ipc-ts-frame-align.patch


mkdir -p "$(dirname /tmp/osmo-trx-rach-per-ra-table.patch)" && cp -r
 ↪  "$REPO"/patches/osmo-trx-rach-per-ra-table.patch /tmp/osmo-trx-rach-per-ra-table.patch


if ! osmo-deb install osmo-trx 1.7.2+ipc; then \
      { [ -d ${ROOT}/osmo-trx ] || { cd ${ROOT} \
          && git clone https://gitea.osmocom.org/cellular-infrastructure/osmo-trx \
          && git -C osmo-trx checkout 1.7.2; }; } \
      && git -C ${ROOT}/osmo-trx apply /tmp/osmo-trx-ipc-ts-frame-align.patch \
```

```
      && git -C ${ROOT}/osmo-trx apply /tmp/osmo-trx-rach-per-ra-table.patch \
      && cd ${ROOT}/osmo-trx \
      && autoreconf -fi \
      && ./configure --with-ipc \
      && make -j$(nproc) \
      && osmo-deb pack osmo-trx 1.7.2+ipc make install \
      && ldconfig; \
    fi


mkdir -p "$(dirname /tmp/osmo-bts-force-rand-toy.patch)" && cp -r "$REPO"/patches/osmo-bts-force-rand-toy.patch
 ↪  /tmp/osmo-bts-force-rand-toy.patch


if ! osmo-deb install osmo-bts 1.10.0+rand; then \
      { [ -d ${ROOT}/osmo-bts ] || { cd ${ROOT} \
          && git clone https://gitea.osmocom.org/cellular-infrastructure/osmo-bts \
          && git -C osmo-bts checkout 1.10.0; }; } \
      && git -C ${ROOT}/osmo-bts apply /tmp/osmo-bts-force-rand-toy.patch \
      && cd ${ROOT}/osmo-bts \
      && autoreconf -fi \
      && ./configure --enable-virtual --enable-trx \
      && make -j$(nproc) \
      && osmo-deb pack osmo-bts 1.10.0+rand make install \
      && ldconfig; \
    fi


mkdir -p "$(dirname /tmp/osmo-hlr-force-rand-toy.patch)" && cp -r "$REPO"/patches/osmo-hlr-force-rand-toy.patch
 ↪  /tmp/osmo-hlr-force-rand-toy.patch


if ! osmo-deb install osmo-hlr 1.9.2+rand; then \
      { [ -d ${ROOT}/osmo-hlr ] || { cd ${ROOT} \
          && git clone https://gitea.osmocom.org/cellular-infrastructure/osmo-hlr \
          && git -C osmo-hlr checkout 1.9.2; }; } \
      && git -C ${ROOT}/osmo-hlr apply /tmp/osmo-hlr-force-rand-toy.patch \
      && cd ${ROOT}/osmo-hlr \
      && autoreconf -fi \
      && ./configure \
      && make -j$(nproc) \
      && osmo-deb pack osmo-hlr 1.9.2+rand make install \
      && ldconfig; \
    fi


mkdir -p "$(dirname /tmp/osmo-msc-force-rand-toy.patch)" && cp -r "$REPO"/patches/osmo-msc-force-rand-toy.patch
 ↪  /tmp/osmo-msc-force-rand-toy.patch


if ! osmo-deb install osmo-msc 1.15.0+rand; then \
      { [ -d ${ROOT}/osmo-msc ] || { cd ${ROOT} \
          && git clone https://gitea.osmocom.org/cellular-infrastructure/osmo-msc \
          && git -C osmo-msc checkout 1.15.0; }; } \
      && git -C ${ROOT}/osmo-msc apply /tmp/osmo-msc-force-rand-toy.patch \
      && cd ${ROOT}/osmo-msc \
      && autoreconf -fi \
      && ./configure --enable-smpp \
      && make -j$(nproc) \
      && osmo-deb pack osmo-msc 1.15.0+rand make install \
      && ldconfig; \
    fi
```

```
if ! osmo-deb install osmo-gapk 0.git; then \
      cd ${ROOT} && \
      git clone https://gitea.osmocom.org/osmocom/gapk osmo-gapk && \
      cd osmo-gapk && \
      autoreconf -fi && \
      ./configure --enable-alsa && \
      make -j$(nproc) && \
      osmo-deb pack osmo-gapk 0.git make install && \
      ldconfig; \
    fi


if ! osmo-deb install gsup-smsc-proto 0.git; then \
      cd ${ROOT} && \
      git clone https://gitea.osmocom.org/themwi/gsup-smsc-proto && \
      cd gsup-smsc-proto && \
      ./configure --with-osmo=/usr/local && \
      make -j$(nproc) && \
      osmo-deb pack gsup-smsc-proto 0.git \
        sh -c 'for i in daemon sendmt; do make -C "$i" DESTDIR="$DESTDIR" install || exit 1; done' && \
      ldconfig; \
    fi


if ! osmo-deb install sms-coding-utils 0.r1; then \
      cd ${ROOT} && \
      wget -q https://www.freecalypso.org/pub/GSM/FreeCalypso/sms-coding-utils-latest.tar.bz2 && \
      tar xf sms-coding-utils-latest.tar.bz2 && \
      cd sms-coding-utils-r1 && \
      ./configure && \
      make -j$(nproc) && \
      osmo-deb pack sms-coding-utils 0.r1 \
        sh -c 'mkdir -p "$DESTDIR/usr/local/bin" && make install DESTDIR="$DESTDIR"'; \
    fi


if ! osmo-deb install libosmo-dsp 0.git; then \
      cd /opt/GSM \
      && git clone https://gitea.osmocom.org/sdr/libosmo-dsp.git \
      && cd libosmo-dsp \
      && autoreconf -fi \
      && ./configure \
      && make -j$(nproc) \
      && osmo-deb pack libosmo-dsp 0.git make install \
      && ldconfig; \
    fi
```

2.25.1.5 Stage 3 : bb

osmocom-bb (the `mobile` application, `osmocon`, layer23) with the patches from `patches/osmocom-bb-*` (clock generator frame tolerance, PLMN handling, SIM PLMN from BCD), the Calypso firmware images (`layer1.highram.elf` and friends) and the two gcc flavours the old trees need (gcc-9 selected through `update-alternatives` while building, gcc-11 restored afterwards). It also builds the `osmocom-bb-transceiver` tree (Sylvain Munaut's jolly/testing branch: the `trx` firmware application and the host `transceiver` program that turns the phone into a BTS transceiver) with `patches/osmocom-bb-transceiver-qemu-bench.patch`, and the `burst-ind` tree.

In Docker: `FROM osmo-core`. *On a machine, carry on in the same system.*

```
export ROOT=/opt/GSM


export OSMO_DEB_REFRESH=0
```

```
mkdir -p "$(dirname /tmp/osmocom-bb-clck-gen-frame-tolerance.patch)" && cp -r
 ↪  "$REPO"/patches/osmocom-bb-clck-gen-frame-tolerance.patch /tmp/osmocom-bb-clck-gen-frame-tolerance.patch


mkdir -p "$(dirname /tmp/osmocom-bb-plmn-a6-reinsert.patch)" && cp -r
 ↪  "$REPO"/patches/osmocom-bb-plmn-a6-reinsert.patch /tmp/osmocom-bb-plmn-a6-reinsert.patch


mkdir -p "$(dirname /tmp/osmocom-bb-sim-plmn-from-bcd.patch)" && cp -r
 ↪  "$REPO"/patches/osmocom-bb-sim-plmn-from-bcd.patch /tmp/osmocom-bb-sim-plmn-from-bcd.patch


if ! osmo-deb install osmocom-bb 0.git; then \
      cd ${ROOT} && \
      git clone https://gitea.osmocom.org/phone-side/osmocom-bb && \
      git -C ${ROOT}/osmocom-bb apply /tmp/osmocom-bb-clck-gen-frame-tolerance.patch && \
      git -C ${ROOT}/osmocom-bb apply /tmp/osmocom-bb-plmn-a6-reinsert.patch && \
      git -C ${ROOT}/osmocom-bb apply /tmp/osmocom-bb-sim-plmn-from-bcd.patch && \
      cd osmocom-bb/src && \
      # Build complet : firmware (layer1.bin/.elf pour Calypso) + outils host
      # (mobile, trxcon, virtphy, ccch_scan). Le firmware est nécessaire pour
      # le mode PHY_MODE=qemu où QEMU émule un Calypso et exécute layer1.
      make nofirmware && \
      osmo-deb snapshot osmocom-bb 0.git ${ROOT}/osmocom-bb; \
    fi


if ! osmo-deb install calypso-firmware 0.git; then \
      GIT_TERMINAL_PROMPT=0 git clone --depth 1 \
        https://github.com/bbaranoff/firmware /opt/GSM/firmware \
      && osmo-deb snapshot calypso-firmware 0.git /opt/GSM/firmware; \
    fi


update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-9 90 \
       --slave /usr/bin/g++ g++ /usr/bin/g++-9 \
    && update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-11 110 \
       --slave /usr/bin/g++ g++ /usr/bin/g++-11


update-alternatives --set gcc /usr/bin/gcc-9


mkdir -p "$(dirname /tmp/osmocom-bb-transceiver-qemu-bench.patch)" && cp -r
 ↪  "$REPO"/patches/osmocom-bb-transceiver-qemu-bench.patch /tmp/osmocom-bb-transceiver-qemu-bench.patch


if ! osmo-deb install osmocom-bb-transceiver 0.git; then \
      git clone --branch jolly/testing --depth 1 \
        https://gitea.osmocom.org/phone-side/osmocom-bb.git \
        /opt/GSM/osmocom-bb-transceiver \
      && git -C /opt/GSM/osmocom-bb-transceiver apply /tmp/osmocom-bb-transceiver-qemu-bench.patch \
      && cd /opt/GSM/osmocom-bb-transceiver/src \
      && make HOST_layer23_CONFARGS=--enable-transceiver nofirmware -j$(nproc) \
      && cp /opt/GSM/osmocom-bb-transceiver/src/host/layer23/src/transceiver/transceiver \
         /usr/local/bin/transceiver \
      && osmo-deb snapshot osmocom-bb-transceiver 0.git /usr/local/bin/transceiver \
            /opt/GSM/osmocom-bb-transceiver; \
    fi
```

```
if ! osmo-deb install osmocom-bb-burst-ind 0.git; then \
      git clone --branch fixeria/burst_ind --depth 1 \
        https://gitea.osmocom.org/phone-side/osmocom-bb.git \
        /opt/GSM/osmocom-bb-burst_ind \
      && cd /opt/GSM/osmocom-bb-burst_ind/src \
      && make nofirmware -j$(nproc) \
      && cp /opt/GSM/osmocom-bb-burst_ind/src/host/layer23/src/misc/ccch_scan \
         /usr/local/bin/ccch_scan \
      && { cp /opt/GSM/osmocom-bb-burst_ind/src/host/layer23/src/misc/bcch_scan \
         /usr/local/bin/bcch_scan 2>/dev/null || true; } \
      && { cp /opt/GSM/osmocom-bb-burst_ind/src/host/layer23/src/misc/cell_log \
         /usr/local/bin/cell_log 2>/dev/null || true; } \
      && osmo-deb snapshot osmocom-bb-burst-ind 0.git /opt/GSM/osmocom-bb-burst_ind \
           $(ls /usr/local/bin/ccch_scan /usr/local/bin/bcch_scan /usr/local/bin/cell_log 2>/dev/null); \
    fi


update-alternatives --set gcc /usr/bin/gcc-11
```

2.25.1.6 Stage 4 : qemu

The single QEMU tree of the project, `/opt/GSM/qosmo`: a QEMU fork with the Calypso machine (ARM core, TPU/TSP, DSP API RAM, LCD and keypad emulation) built with `--enable-l1-grgsm`, the gr-gsm layer 1 shunt. The C54x DSP itself is no longer run inside QEMU: see the `l1` stage.

In Docker: `FROM base`. *On a machine, carry on in the same system.*

```
export ROOT=/opt/GSM


export OSMO_DEB_REFRESH=0


if ! osmo-deb install qosmo 0.git; then \
      git clone https://github.com/bbaranoff/qosmO /opt/GSM/qosmo \
      && cd /opt/GSM/qosmo \
      && python3 -m venv /root/.venv-qemu \
      && . /root/.venv-qemu/bin/activate \
      && pip install --no-cache-dir tomli \
      && mkdir -p build && cd build \
      && ../configure --target-list=arm-softmmu --enable-l1-grgsm \
             --prefix=/opt/GSM/qemu-install --disable-werror --disable-docs \
      && make -j$(nproc) \
      && make install \
      && cp /opt/GSM/qemu-install/bin/qemu-system-arm /usr/local/bin/qemu-system-arm \
      && cp /opt/GSM/qemu-install/bin/qosmo /usr/local/bin/qosmo \
      && osmo-deb snapshot qosmo 0.git /opt/GSM/qosmo /opt/GSM/qemu-install \
             /root/.venv-qemu /usr/local/bin/qemu-system-arm /usr/local/bin/qosmo; \
    fi
```

2.25.1.7 Stage 5 : l1

The two layer-1 engines that run outside QEMU: `c54x_exe` (the TMS320C54x core emulated from the qosmo sources, loading the real TI mask ROM, see the appendix) and `grgsm_exe` (the gr-gsm based layer 1). The DSP ROM images come from the FreeCalypso dump and are split into the seven sections `c54x_exe` reads.

In Docker: `FROM qemu`. *On a machine, carry on in the same system.*

```
export ROOT=/opt/GSM


export OSMO_DEB_REFRESH=0
```

Copy between Docker stages (`--from=osmo-core /usr/local /usr/local`): on a single machine `/usr/local` is already in place.

```
ldconfig


export FREECALYPSO_URL=ftp://ftp.freecalypso.org/pub/GSM/Calypso


if ! osmo-deb install c54x-exe 0.git; then \
      git clone https://github.com/bbaranoff/c54x_exe /opt/GSM/c54x_exe \
      && cd /opt/GSM/c54x_exe \
      && make QOSMO=/opt/GSM/qosmo \
             CFLAGS="-O3 -g -Wall -Werror=format -Werror=format-extra-args -Wno-unused-function
 ↪  -Wno-unused-variable -Wno-unused-but-set-variable -Wno-sign-compare" \
      && FREECALYPSO_URL="$FREECALYPSO_URL" bash rom/fetch-rom.sh --dest /opt/GSM --dest /opt/GSM/c54x_exe/rom \
      && curl -fsS --retry 2 -o /opt/GSM/calypso_dsp.txt "$FREECALYPSO_URL/dsp-rom-3606-dump.txt" \
      && cp /opt/GSM/calypso_dsp.txt rom/calypso_dsp.txt \
      && osmo-deb snapshot c54x-exe 0.git /opt/GSM/c54x_exe \
             /opt/GSM/calypso_dsp.PROM0.bin /opt/GSM/calypso_dsp.PROM1.bin /opt/GSM/calypso_dsp.PROM2.bin \
             /opt/GSM/calypso_dsp.PROM3.bin /opt/GSM/calypso_dsp.DROM.bin /opt/GSM/calypso_dsp.PDROM.bin \
             /opt/GSM/calypso_dsp.Registers.bin /opt/GSM/calypso_dsp.txt; \
    fi


if ! osmo-deb install grgsm-exe 0.git; then \
      git clone https://github.com/bbaranoff/grgsm_exE /opt/GSM/grgsm_exe \
      && cd /opt/GSM/grgsm_exe \
      && make QOSMO=/opt/GSM/qosmo \
      && osmo-deb snapshot grgsm-exe 0.git /opt/GSM/grgsm_exe; \
    fi
```

2.25.1.8 Stage 6 : grgsm-venv

GNU Radio 3.10, gr-osmosdr and gr-gsm installed in the Python virtual environment `/root/.env`, with the project patches for gr-gsm (optional Qt GUI import, receiver publishing BSIC and frame number). The venv is activated from root's profile at the end of the final stage.

In Docker: `FROM osmo-core`. On a machine, carry on in the same system.

```
export ROOT=/opt/GSM


export OSMO_DEB_REFRESH=0


mkdir -p "$(dirname /tmp/grgsm-receiver-publish-bsic-fn.patch)" && cp -r
 ↪  "$REPO"/patches/grgsm-receiver-publish-bsic-fn.patch /tmp/grgsm-receiver-publish-bsic-fn.patch


if ! osmo-deb install grgsm-venv 0.git; then \
      curl -fsSL
       ↪  https://gist.githubusercontent.com/bbaranoff/3683811057933af0954b661821e950d1/raw/fcdb4092483ec383440b67fc002db0c1583
       ↪  | bash \
      && git -C /opt/GSM/gr-gsm apply /tmp/grgsm-receiver-publish-bsic-fn.patch \
      && cd /opt/GSM/gr-gsm/build \
      && make -j"$(nproc)" \
      && make install \
      && . ~/.env/bin/activate && pip install matplotlib \
      && osmo-deb snapshot grgsm-venv 0.git /root/.env /etc/ld.so.conf.d/gnuradio.conf; \
    fi && ldconfig
```

2.25.1.9 Stage 7 : node

Node 22 and the web dashboard `osmo-egprs-web` (the operations console served as a systemd unit), plus its npm dependencies.

In Docker: `FROM base`. On a machine, carry on in the same system.

```
export ROOT=/opt/GSM


export OSMO_DEB_REFRESH=0


if [ ! -d "/opt/GSM/osmo-egprs-web/" ]; then \
      mkdir -p /opt/GSM && \
      git clone -b main https://github.com/bbaranoff/osmo-egprs-web /opt/GSM/osmo-egprs-web; \
    fi


export NODE_VERSION=v22.23.2


case "$(uname -m)" in x86_64) _na=x64 ;; aarch64) _na=arm64 ;; *) echo "arch node inconnue: $(uname -m)" >&2; exit
↪  1 ;; esac && \
    curl -fsSL "https://nodejs.org/dist/${NODE_VERSION}/node-${NODE_VERSION}-linux-${_na}.tar.xz" \
        -o /tmp/node.tar.xz && \
    mkdir -p /opt/node && \
    tar -xJf /tmp/node.tar.xz -C /opt/node --strip-components=1 && \
    rm -f /tmp/node.tar.xz && \
    ln -sf /opt/node/bin/node /usr/local/bin/node && \
    ln -sf /opt/node/bin/npm  /usr/local/bin/npm && \
    ln -sf /opt/node/bin/npx  /usr/local/bin/npx && \
    node --version


mkdir -p "$(dirname /etc/systemd/system/osmo-egprs-web.service)" && cp -r "$REPO"/services/osmo-egprs-web.service
↪  /etc/systemd/system/osmo-egprs-web.service


ln -sf /etc/systemd/system/osmo-egprs-web.service \
        /etc/systemd/system/multi-user.target.wants/osmo-egprs-web.service


mkdir -p "$(dirname /opt/GSM/osmo-egprs-web/osmo-egprs-web.service)" && cp -r
↪  "$REPO"/services/osmo-egprs-web.service /opt/GSM/osmo-egprs-web/osmo-egprs-web.service


test -s /opt/GSM/osmo-egprs-web/install-web-service.sh


cd /opt/GSM/osmo-egprs-web && npm install --omit=dev --no-audit --no-fund || true


bash /opt/GSM/osmo-egprs-web/install-web-service.sh || true
```

2.25.1.10 Stage 8 : lte

The 4G side of the bench: srsRAN_4G built with the ZeroMQ radio (no SDR needed) and Open5GS, both under `/opt/LTE`, with their configuration files.

In Docker: `FROM base`. On a machine, carry on in the same system.

```
export ROOT=/opt/GSM


export OSMO_DEB_REFRESH=0


git clone https://github.com/bbaranoff/osmo-operator /opt/GSM/osmo-operator


mkdir -p "$(dirname /usr/local/sbin/osmo-lte-install)" && cp -r "$REPO"/tools/osmo-lte-install.sh
 ↪  /usr/local/sbin/osmo-lte-install


chmod 755 /usr/local/sbin/osmo-lte-install && \
    OSMO_REPO=/opt/GSM/osmo-operator osmo-lte-install --deps && \
    { osmo-deb install libzmq 4.3.5+git || true; } && \
    { osmo-deb install srsgui 0.1+git || true; } && \
    SRS_DEB_VER="$(osmo-lte-install --srs-deb-version)" && \
    if ! osmo-deb install srsran "$SRS_DEB_VER"; then \
        OSMO_DEB=1 OSMO_REPO=/opt/GSM/osmo-operator osmo-lte-install --build || { echo "ECHEC build srsRAN"; exit
         ↪  1; }; \
    fi && \
    if ! osmo-deb install open5gs 2.8.0+git; then \
        OSMO_DEB=1 OSMO_REPO=/opt/GSM/osmo-operator osmo-lte-install --build || { echo "ECHEC build Open5GS"; exit
         ↪  1; }; \
    fi && \
    OSMO_REPO=/opt/GSM/osmo-operator osmo-lte-install --configs --launchers && \
    ldconfig && test -x /usr/local/bin/srsenb && test -x /opt/LTE/open5gs/install/bin/open5gs-mmed \
    && test -s /opt/LTE/open5gs/install/etc/open5gs/mme.yaml && grep -q '^  sgsap:'
     ↪  /opt/LTE/open5gs/install/etc/open5gs/mme.yaml \
    && test -s /opt/LTE/open5gs/install/etc/freeDiameter/mme.conf && test -s
     ↪  /opt/LTE/open5gs/install/etc/open5gs/tls/mme.crt \
    && test -s /root/.config/srsran/enb.conf && test -x /usr/local/bin/osmo-lte && test -x /usr/local/bin/osmo-epc


mkdir -p "$(dirname /tmp/pmbootstrap-osmo-bench-qemu.patch)" && cp -r
 ↪  "$REPO"/patches/pmbootstrap-osmo-bench-qemu.patch /tmp/pmbootstrap-osmo-bench-qemu.patch


if ! osmo-deb install pmbootstrap 0.git; then \
        mkdir -p /opt/user_interface/pmos && \
        git clone https://gitlab.postmarketos.org/postmarketOS/pmbootstrap.git
         ↪  /opt/user_interface/pmos/pmbootstrap && \
        cd /opt/user_interface/pmos/pmbootstrap && \
        git apply /tmp/pmbootstrap-osmo-bench-qemu.patch && \
        grep -q osmo_bench_args pmb/commands/qemu.py && \
        osmo-deb snapshot pmbootstrap 0.git /opt/user_interface/pmos/pmbootstrap; \
    fi && test -f /opt/user_interface/pmos/pmbootstrap/pmbootstrap.py
```

2.25.1.11 Stage 9 : osmocom-nitb

The assembly stage, the image that is actually published. Nothing is compiled here: the `.deb` packages produced by the previous stages are installed (`dpkg -i`), then the project files are laid out: the `/etc/osmocom` configurations, the scripts under `/usr/local/bin`, the systemd units, the PulseAudio system daemon configuration (GSM audio), the runtime directories, the shell prompt, the desktop pieces and the `/opt/GSM` helper scripts. On a single machine the `COPY --from=<stage>` instructions are no-ops: everything is already installed where the previous stages put it. The container entrypoint and stop signal are Docker metadata only.

In Docker: `FROM base`. On a machine, carry on in the same system.

```
export ROOT=/opt/GSM
```

Copy between Docker stages (`--from=bb /var/cache/osmo-debs/ /var/cache/osmo-debs/`): on a single machine `/var/cache/osmo-debs/` is already in place.

Copy between Docker stages (`--from=l1 /var/cache/osmo-debs/ /var/cache/osmo-debs/`): on a single machine `/var/cache/osmo-debs/` is already in place.

Copy between Docker stages (`--from=grgsm-venv /var/cache/osmo-debs/ /var/cache/osmo-debs/`): on a single machine `/var/cache/osmo-debs/` is already in place.

Copy between Docker stages (`--from=lte /var/cache/osmo-debs/ /var/cache/osmo-debs/`): on a single machine `/var/cache/osmo-debs/` is already in place.

```
set -eux; \
    dpkg -i --force-overwrite /var/cache/osmo-debs/osmo-build-*.deb; \
    dpkg -i --force-overwrite \
        /var/cache/osmo-debs/osmo-build-osmo-trx_1.7.2+ipc~*.deb \
        /var/cache/osmo-debs/osmo-build-osmo-bts_1.10.0+rand~*.deb \
        /var/cache/osmo-debs/osmo-build-osmo-hlr_1.9.2+rand~*.deb \
        /var/cache/osmo-debs/osmo-build-osmo-msc_1.15.0+rand~*.deb; \
    ldconfig
```

Copy between Docker stages (`--from=node /opt/node /opt/node`): on a single machine `/opt/node` is already in place.

Copy between Docker stages (`--from=node /opt/GSM/osmo-egprs-web /opt/GSM/osmo-egprs-web`): on a single machine `/opt/GSM/osmo-egprs-web` is already in place.

Copy between Docker stages (`--from=node /etc/systemd/system/osmo-egprs-web.service /etc/systemd/system/osmo-egprs-web.service`): on a single machine `/etc/systemd/system/osmo-egprs-web.service` is already in place.

```
ln -sf /opt/node/bin/node /usr/local/bin/node \
    && ln -sf /opt/node/bin/npm  /usr/local/bin/npm \
    && ln -sf /opt/node/bin/npx  /usr/local/bin/npx \
    && ln -sf /etc/systemd/system/osmo-egprs-web.service \
        /etc/systemd/system/multi-user.target.wants/osmo-egprs-web.service \
    && node --version
```

```
git clone https://github.com/bbaranoff/osmo-operator /opt/GSM/osmo-operator
```

```
mkdir -p "$(dirname /usr/local/sbin/osmo-lte-install)" && cp -r "$REPO"/tools/osmo-lte-install.sh
 ↪  /usr/local/sbin/osmo-lte-install
```

```
chmod 755 /usr/local/sbin/osmo-lte-install
```

Copy between Docker stages (`--from=lte /opt/LTE /opt/LTE`): on a single machine `/opt/LTE` is already in place.

```
OSMO_REPO=/opt/GSM/osmo-operator osmo-lte-install --deps && \
    OSMO_REPO=/opt/GSM/osmo-operator osmo-lte-install --configs --launchers && \
    ldconfig && test -x /usr/local/bin/srsenb \
    && test -x /opt/LTE/open5gs/install/bin/open5gs-mmed \
    && test -s /opt/LTE/open5gs/install/etc/open5gs/mme.yaml && grep -q '^  sgsap:'
     ↪  /opt/LTE/open5gs/install/etc/open5gs/mme.yaml \
    && test -s /opt/LTE/open5gs/install/etc/freeDiameter/mme.conf \
    && test -s /opt/LTE/open5gs/install/etc/open5gs/tls/mme.crt \
    && test -s /root/.config/srsran/enb.conf && test -x /usr/local/bin/osmo-lte && test -x /usr/local/bin/osmo-epc


update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-9 90 \
       --slave /usr/bin/g++ g++ /usr/bin/g++-9 \
    && update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-11 110 \
       --slave /usr/bin/g++ g++ /usr/bin/g++-11 \
    && update-alternatives --set gcc /usr/bin/gcc-11


cd /etc/osmocom


mkdir -p /etc/osmocom/ && cp -r "$REPO"/scripts/. /etc/osmocom/


mkdir -p /etc/osmocom/ && cp -r "$REPO"/configs/*cfg /etc/osmocom/


mv /etc/osmocom/run.sh /root/run.sh


chmod +x /etc/osmocom/gapk-start.sh && \
    ln -sf /etc/osmocom/gapk-start.sh /usr/local/bin/gapk-start.sh && \
    mkdir -p /var/lib/gapk


cp -f /usr/local/bin/osmo* /usr/bin/ || true && \
    cp -f /usr/local/bin/proto-smsc-* /usr/bin/ || true && \
    cp -f /usr/local/bin/sms-* /usr/bin/ || true && \
    cp -f /usr/local/bin/gen-sms-* /usr/bin/ || true && \
    # Si tu as des fichiers .service dans configs/
    cp /etc/osmocom/configs/*.service /lib/systemd/system/ 2>/dev/null || true


which proto-smsc-daemon && which proto-smsc-sendmt && \
    echo "proto-smsc binaries OK" || \
    echo "WARNING: proto-smsc binaries not found in PATH"


sed -i 's/^CPUScheduling/#CPUScheduling/g' /lib/systemd/system/osmo-*.service && \
    sed -i 's/User=osmocom/User=root/g' /lib/systemd/system/osmo-ggsn.service && \
    sed -i 's/User=osmocom/User=root/g' /lib/systemd/system/osmo-sgsn.service && \
    chmod +x /etc/osmocom/*.sh


mkdir -p "$(dirname /etc/systemd/system/osmo-bts-trx.service)" && cp -r "$REPO"/services/osmo-bts-trx.service
 ↪  /etc/systemd/system/osmo-bts-trx.service


systemctl enable osmo-bts-trx.service && \
    passwd -d root && \
    systemctl mask getty@tty1.service serial-getty@tty1.service


mkdir -p /root/.osmocom/bb/ \
    && install -m755 ${ROOT}/osmocom-bb/src/host/trxcon/src/trxcon           /usr/local/bin/trxcon \
    && install -m755 ${ROOT}/osmocom-bb/src/host/layer23/src/mobile/mobile   /usr/local/bin/mobile \
    && install -m755 ${ROOT}/osmocom-bb/src/host/virt_phy/src/virtphy        /usr/local/bin/virtphy
```

```
echo "alias faketrx='python3 ${ROOT}/osmocom-bb/src/target/trx_toolkit/fake_trx.py'" >> ~/.bashrc


printf '%s\n' '' \
    '### calypso-prompt ###' \
    "export PS1='\[\033[1;31m\]\u\[\033[0m\]@\[\033[1;34m\]\h\[\033[0m\]:\[\033[1;32m\]\w\[\033[0m\]☎# '" \
    '### end calypso-prompt ###' >> /root/.bashrc


mkdir -p "$(dirname /root/.osmocom/bb/mobile.cfg)" && cp -r "$REPO"/configs/mobile.cfg
 ↪  /root/.osmocom/bb/mobile.cfg


chmod +x /root/run.sh


mkdir -p /var/log/osmocom /var/run/smsc \
        /etc/asterisk /var/lib/asterisk /var/log/asterisk /var/run/asterisk \
        /root/.osmocom/bb && \
    chmod 755 /etc/asterisk /var/lib/asterisk /var/log/asterisk \
        /var/run/asterisk /var/log/osmocom


set -eux; \
    usermod -aG pulse-access root 2>/dev/null || true; \
    test -f /etc/pulse/system.pa; \
    sed -i 's|^#\?load-module module-native-protocol-unix.*|load-module module-native-protocol-unix
     ↪  auth-anonymous=1 socket=/var/run/pulse/native|' /etc/pulse/system.pa; \
    grep -q 'auth-anonymous=1' /etc/pulse/system.pa \
      || echo 'load-module module-native-protocol-unix auth-anonymous=1 socket=/var/run/pulse/native' >>
       ↪  /etc/pulse/system.pa; \
    grep -q 'sink_name=gsm_audio' /etc/pulse/system.pa \
      || echo 'load-module module-null-sink sink_name=gsm_audio format=s16le rate=8000 channels=1
       ↪  sink_properties=device.description=GSM_Audio' >> /etc/pulse/system.pa; \
    grep -q 'sink_name=gsm_mic' /etc/pulse/system.pa \
      || echo 'load-module module-null-sink sink_name=gsm_mic format=s16le rate=8000 channels=1
       ↪  sink_properties=device.description=GSM_Mic' >> /etc/pulse/system.pa; \
    grep -q 'sink_name=osmo_tts_off' /etc/pulse/system.pa \
      || echo 'load-module module-null-sink sink_name=osmo_tts_off format=s16le rate=8000 channels=1
       ↪  sink_properties=device.description=TTS_off' >> /etc/pulse/system.pa; \
    sed -i 's|^load-module module-suspend-on-idle|#load-module module-suspend-on-idle|' /etc/pulse/system.pa; \
    mkdir -p /var/run/pulse && chown -R pulse:pulse /var/run/pulse


echo 'source ~/.env/bin/activate' >> ~/.bashrc


mkdir -p /opt/GSM/ && cp -r "$REPO"/opt-gsm/. /opt/GSM/
```

Docker metadata, no equivalent on a machine: `STOPSIGNAL SIGRTMIN+3`.

Docker metadata, no equivalent on a machine: `ENTRYPOINT ["/etc/osmocom/entrypoint.sh"]`.

Docker metadata, no equivalent on a machine: `CMD ["/bin/bash"]`.

2.25.1.12 Stage 10 : debs

A `scratch` stage holding only `/var/cache/osmo-debs`, used by the CI to extract the compiled packages without materialising the full image. Nothing to do on a machine.

Docker stage without content of its own (`scratch` image): nothing to do on a machine.

Copy between Docker stages (`--from=osmocom-nitb /var/cache/osmo-debs/ /`): on a single machine `/var/cache/osmo-debs/` is already in place.

2.25.2 Appendix: c54x_exe-from-scratch-ubuntu.en.md

Contents of `doc/c54x_exe-from-scratch-ubuntu.en.md`, included as is.

2.25.2.1 c54x_exe from a clean Ubuntu 24.04

Oct 20, 2026 · Bastien Baranoff

2.25.2.1.1 What this page shows

In about twenty minutes, the emulated TMS320C54x core of `c54x_exe` loads the TI mask ROM of a real Calypso and runs it, without QEMU, without the ARM, without a GSM network.

This is the DSP's hello world: it proves the core executes real TI code. It does not prove the DSP decodes anything, since no burst is injected. The full setup (ARM under QEMU, BTS, bridge, voice call) is described at the end of the page and lives in osmo-operator.

Everything runs as root or with `sudo`; the `/opt/GSM/…` paths are the ones the Makefiles and scripts expect by default.

2.25.2.1.2 1. Prerequisites (Ubuntu 24.04)

One `apt` line is enough: the compiler, `git`, `curl` for the ROMs, and `libosmocore` (which contains `libosmocoding`, used by `calypso_bsp.c`).

```
sudo apt update
sudo apt install -y build-essential git curl pkg-config libosmocore-dev
sudo mkdir -p /opt/GSM
```

Check that `libosmocoding` is there:

```
pkg-config --modversion libosmocoding
```

If `pkg-config` cannot find it, the `libosmocore` of the Ubuntu repository is too old or incomplete; install `libosmocore-dev` from the Osmocom nightly repository instead (see osmocom.org).

2.25.2.1.3 2. The qosmo sources

`c54x_exe` does not copy the C54x core: it compiles the qosmo sources directly (`calypso_c54x.c` and its neighbours) with ~120 lines of QEMU header stand-ins. The repository must be on disk, but it does not need to be built.

```
cd /opt/GSM
git clone https://github.com/bbaranoff/qosmo
```

No `configure`, no `make` here. If qosmo is somewhere other than `/opt/GSM/qosmo`, pass `QOSMO=/path` to the `make` of the next step.

2.25.2.1.4 3. Clone and build c54x_exe

The build takes less than a minute and produces a single binary, `c54x_exe`, at the root of the repository.

```
cd /opt/GSM
git clone https://github.com/bbaranoff/c54x_exe
cd c54x_exe
make QOSMO=/opt/GSM/qosmo \
     CFLAGS="-O3 -g -Wall -Werror=format -Werror=format-extra-args \
             -Wno-unused-function -Wno-unused-variable \
             -Wno-unused-but-set-variable -Wno-sign-compare"
```

This `CFLAGS` is the one of the osmo-operator Dockerfile: it keeps format errors fatal and silences the unused-variable warnings, numerous in the QEMU sources taken as they are. A bare `make` works too but is much noisier.

`make` rebuilds the binary on every call (`.PHONY` target); no `make clean` is needed after a change in qosmo.

Check:

```
./c54x_exe --help | head
```

2.25.2.1.5 4. The DSP ROMs, from FreeCalypso

The C54x mask ROM is not in the repository: it comes from the dump published by the FreeCalypso project (Mychaela Falconia) on its public FTP. The `rom/fetch-rom.sh` script downloads it and splits it into the 7 sections `--rom-dir` expects.

```
cd /opt/GSM/c54x_exe
export FREECALYPSO_URL=ftp://ftp.freecalypso.org/pub/GSM/Calypso
bash rom/fetch-rom.sh --dest /opt/GSM --dest /opt/GSM/c54x_exe/rom
curl -fsS --retry 2 -o /opt/GSM/calypso_dsp.txt "$FREECALYPSO_URL/dsp-rom-3606-dump.txt"
cp /opt/GSM/calypso_dsp.txt rom/calypso_dsp.txt
```

`3606` is the DSP ROM version of the Calypso in the Motorola C1xx; it is also the value the DSP L1 writes into the API RAM at boot, which will later confirm that the right ROM is running.

After the script, `/opt/GSM` must contain exactly these 7 files:

```
ls -l /opt/GSM/calypso_dsp.*.bin
## calypso_dsp.PROM0.bin  calypso_dsp.PROM1.bin  calypso_dsp.PROM2.bin
## calypso_dsp.PROM3.bin  calypso_dsp.DROM.bin   calypso_dsp.PDROM.bin
## calypso_dsp.Registers.bin
```

If `curl` refuses FTP (some builds come without it), download the dump by hand from the same directory on ftp.freecalypso.org and give the local file to `fetch-rom.sh`.

2.25.2.1.6 5. First run: the ROM runs

```
cd /opt/GSM/c54x_exe
./c54x_exe --trames 200
```

`--trames 200` runs the DSP for 200 TDMA frames (one frame = 4.615 ms of GSM time), with the default budget of 2300 instructions per frame in standalone mode. Without options only core errors go to stderr; the final summary says how many lines were hidden and which `-v` level shows them.

What you should see, in order:

1. The 7 ROM sections loaded from `/opt/GSM` (or `--rom-dir`).
2. `c54x_reset()`: the ROM restarts at `0xff80`, its reset vector.
3. `BRANCH-TRACE` lines around `PC=0xb41f`: the TI ROM's wait loop, real code executed by the core.
4. A final summary with `a_sch[3] = 0x771a`.

For one line per frame:

```
./c54x_exe --trames 50 --verbeux
```

And for core traces, `-v` to `-vvvvvv` (see `--help`).

2.25.2.1.7 6. Reading the result

`a_sch[3] = 0x771a` is a constant from one run to the next: the ROM booted, parked itself in IDLE and left this state in its API window. It is the proof that the core executes the ROM faithfully up to that point.

What it does not prove: decoding. The API RAM is blank, no burst arrives, the DSP runs idle. A decoder whose output does not depend on its input does not decode; varying the input requires the full setup of section 8.

Useful reference: the `qemu-calypso` README gives `0xf8d8` for the same word. The difference comes from the input (QEMU provides an API RAM initialised by the ARM, here it is empty), not from a bug: two different inputs, two different constants.

2.25.2.1.8 7. Common errors

Symptom	Cause	Fix
`make`: `qemu/osdep.h` or `calypso_c54x.c` not found	qosmo is not at `/opt/GSM/qosmo`	`make QOSMO=/path/to/qosmo`
`make`: `osmocom/coding/...` not found	`libosmocore-dev` missing or too old	section 1, Osmocom nightly repository
`fetch-rom.sh` fails or `curl: (1) Protocol "ftp" not supported`	`curl` built without FTP, or a firewall	download by hand, then `fetch-rom.sh` on the local file
`c54x_exe`: ROM not found	the `.bin` files are not in `/opt/GSM`	`--rom-dir /path` or redo section 4
The summary does not show `0x771a`	wrong ROM, or badly split sections	compare `rom/calypso_dsp.txt` with the 3606 dump; redo section 4
`PC=0xb41f` never appears, the core exits with an error	regression in qosmo	`cd /opt/GSM/qosmo && git log -3`, go back to the README's commit

2.25.2.1.9 8. Next: injecting bursts

For the DSP to really decode, you need the ARM (the osmocom-bb `layer1.highram.elf` firmware under the `qosmo` QEMU), a BTS and the bridge that delivers bursts to it. Two paths:

The complete bench. It installs everything, in the same `/opt/GSM/…` paths as this page:

```
cd /opt/GSM
git clone https://github.com/bbaranoff/osmo-operator
cd osmo-operator
sudo ./install.sh
sudo ./start-direct.sh --dsp
```

`--dsp` replaces the gr-gsm layer 1 with `c54x_exe`; `--menu` asks the same question interactively. The osmo-operator wiki describes the options, the ports and the numbering plan.

Executable by executable. To watch each step produce its log: the DSP (`c54x_exe --arm -v`), then QEMU with `CALYPSO_DSP_EXTERN=1`, then `osmocon`, then `mobile`. The sequence, with the expected log lines at each step, is in LAUNCH.md and the c54x_exe README.

From there, `a_sch` changes with the input, and the question "does it decode?" has a measurable answer.

2.25.2.1.10 9. The Calypso firmware: the arm-elf toolchain (gnuarm)

The osmocom-bb firmware (`layer1.highram.elf`, `trx.highram.elf`…) is built with the historical `arm-elf` toolchain (gcc 4.5.2, binutils 2.21.1, newlib 1.19) from the gnu-arm-installer, installed in `/root/gnuarm/install`. On Ubuntu 24.04 its `cc1` asks for `libmpfr.so.4`, which the system no longer ships (it has `libmpfr.so.6`); a symlink is enough:

```
ln -sf /usr/lib/x86_64-linux-gnu/libmpfr.so.6 /usr/local/lib/libmpfr.so.4 && ldconfig
/root/gnuarm/install/bin/arm-elf-gcc --version      # gcc 4.5.2
```

The `osmocom-bb-transceiver` tree (jolly/testing branch, `trx` application) also builds with the system's `arm-none-eabi-gcc`, with the adaptations of `patches/osmocom-bb-transceiver-qemu-bench.patch` (prefix, linking through gcc, `-fcommon`, `solve_envs.py` ported to Python 3):

```
cd /opt/GSM/osmocom-bb-transceiver/src/target/firmware
make BOARDS=compal_e88 APPLICATIONS=trx ENVIRONMENTS=highram CROSS_COMPILE=arm-none-eabi-
cp board/compal_e88/trx.highram.{elf,bin} /opt/GSM/firmware/board/compal_e88/
```

Every patch applied to the osmocom-bb trees (mobile, transceiver, burst-ind) lives in `patches/` of osmo-operator and is applied by the Dockerfile and `install_modules/40-patches.sh`.

2.26 /opt/GSM/osmo-operator/doc/install-from-scratch.md

87927 octets, 1555 lignes

2.26.1 Installation depuis zero (generee depuis le Dockerfile)

> Fichier genere par `tools/dockerfile-to-install-md.sh` le 2026-10-20 depuis `Dockerfile`. Ne pas editer a la main : corriger le Dockerfile ou le script, puis regenerer.

2.26.1.1 Comment lire cette page

Le Dockerfile d'osmo-operator est la description complete et testee de l'installation. Cette page le rejoue etape par etape sur une machine nue (Ubuntu 24.04, en root), sans Docker :

- chaque `FROM ... AS <nom>` du Dockerfile est un chapitre ; sur une seule machine les etapes se suivent simplement, les `COPY --from=<etape>` ne sont plus necessaires (tout est deja en place) ;
- chaque `RUN` est un bloc bash a coller tel quel dans un shell root, dans l'ordre ;
- chaque `ARG`/`ENV` devient un `export` a poser avant les blocs qui suivent (les `ARG` ne traversent pas un `FROM` dans Docker, ici on les garde dans le shell) ;
- chaque `COPY` devient un `cp` depuis une copie du depot osmo-operator (`$REPO`, par defaut `/opt/GSM/osmo-operator`).

Le cache de paquets `.deb` (`osmo-deb install <nom> <version>`) est une optimisation du build Docker : sans cache, la commande echoue et la branche `if ! osmo-deb install ...; then ... fi` compile depuis les sources, ce qui est exactement ce qu'on veut sur une machine nue. `osmo-deb pack` emballe ensuite le resultat (`make install`) ; sans `/var/cache/osmo-debs` il se contente d'installer.

```
export REPO=/opt/GSM/osmo-operator      # copie du depot (git clone https://github.com/bbaranoff/osmo-operator)
export ROOT=/opt/GSM
mkdir -p "$ROOT"
```

2.26.1.2 Sommaire

1. base (a partir de `ubuntu:24.04`)
2. osmo-core (a partir de `base`)
3. bb (a partir de `osmo-core`)
4. qemu (a partir de `base`)
5. l1 (a partir de `qemu`)
6. grgsm-venv (a partir de `osmo-core`)
7. node (a partir de `base`)
8. lte (a partir de `base`)
9. osmocom-nitb (a partir de `base`)
10. debs (a partir de `scratch`)

2.26.1.3 Etape 1 : base

Dockerfile — IMAGE DE BASE osmocom-nitb (COUCHE STABLE)

Tout ce qui est LONG et rarement modifie vit ici : la TOTALITE des paquets apt (une seule liste, plus aucun apt dans Dockerfile.run), la pile Osmocom compilee depuis les sources, osmocom-bb + le firmware Calypso, QEMU et son device IPC, gr-gsm/GNU Radio, le runtime Node et le dashboard web, la config PulseAudio, les units systemd, les repertoires, le prompt. ~11 Go et ~40 min de build : on ne la rebatit que quand une dependance change.

[2026-09-03] BASE ubuntu:24.04 (noble) - python 3.12, gcc 13 par defaut, bibliotheques renommees *t64 (libasound2t64, libgnutls30t64…). L ISO (build-iso.sh) part du MEME noble : le venv /root/.env et les .so de /usr/local sont copies tels quels dans le rootfs, ils doivent trouver la meme glibc et le meme python. Changer la base ici, c est la changer la-bas.

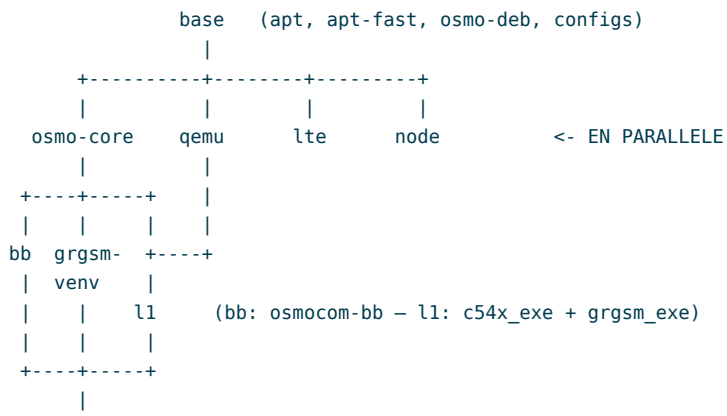
CACHE .deb - packaging/osmo-deb.sh. Chaque dossier compile ci-dessous sort en paquet .deb dans /var/cache/osmo-debs (COPY depuis .deb-cache/, que build.sh synchronise avec /var/cache/osmo-debs de l HOTE). Au rebuild, `osmo-deb install` pose le paquet et saute le clone + la compilation ; `osmo-deb pack` et `osmo-deb snapshot` fabriquent le paquet la premiere fois. build.sh –no-cache passe OSMO_DEB_REFRESH=1 : tout est recompile et le cache reecrit. LES SOURCES RESTENT DANS /opt/GSM : `osmo-deb pack` embarque l arbre courant (/opt/GSM/, .git et objets compris) dans le paquet, et les snapshots nomment l arbre. Un rebuild depuis le cache redonne donc la meme image, avec ses ateliers - pas une image ou seul /usr/local subsiste.

L'iteration quotidienne se fait dans Dockerfile.run, qui repart de cette image (`FROM osmocom-nitb`) et n'y rafraichit que les scripts, les configs, le pont et les arbres git qosmo / osmo-operator — en secondes, pas en 40 minutes.

Cette image reste AUTONOME : elle a son propre ENTRYPOINT et start-nitb.sh la lance seule. Ne pas retirer ses COPY de configs/scripts sous pretexte que Dockerfile.run les refait : le recouvrement est voulu des deux cotes.

[2026-09-25] LE BUILD EST UN GRAPHE, PLUS UNE CHAINE — stages BuildKit

Ce fichier etait une seule cible de ~75 etapes : une chaine, donc un seul chemin d execution, meme sur une machine a 32 coeurs. Or quatre des cinq compilations longues n ont AUCUNE dependance entre elles - la pile Osmocom, QEMU (qosmo), le venv GNU Radio, la 4G (srsRAN + Open5GS). Elles s additionnaient pour rien.



```
osmocom-nitb      (assemblage : dpkg -i, puis configs)
      |
    debs          (FROM scratch : le cache .deb pour la CI)
```

CE QUI REND LE DECOUPAGE POSSIBLE : packaging/osmo-deb.sh. Chaque bloc sort deja en .deb qui contient A LA FOIS les fichiers installes ET l arbre de sources de /opt/GSM (cmd_pack, « L ARBRE DE SOURCES PART AVEC »). Un stage parallele n a donc pas a exporter un systeme de fichiers : il exporte ses .deb, et l assemblage les repose avec un `dpkg -i`. C est EXACTEMENT ce que fait deja iso_modules/50-injection-image.sh pour le rootfs de l ISO — le precedent existe dans ce depot, on le reutilise.

TROIS STAGES NE SONT PAS PARALLELES A osmo-core, et c est voulu — tous les trois se lient a libosmocore d une facon ou d une autre :

```
- osmocom-bb : `make nofirmware` lie mobile/trxcon a libosmocore ;
- c54x_exe   : son Makefile fait `pkg-config --libs libosmocoding libosmocore` ;
- gr-gsm     : son CMakeLists cherche libosmocore/libosmocoding/libosmogsm.
```

Les en sortir donnerait un build qui casse, pas un build plus rapide. Ils tournent en revanche EN PARALLELE ENTRE EUX, une fois osmo-core fini.

CE QUI NE CHANGE PAS : les versions, les patchs, les ./configure, le contenu de /opt/GSM, et –target osmocom-nitb (build.sh, compose.yaml) reste OBLIGATOIRE puisque `debs` est toujours la derniere etape du fichier.

Dans Docker : `FROM ubuntu:24.04`. Sur une machine, on continue dans le meme systeme.

ROOT : ou vivent les sources dans l'image. Chemin FIXE et assume — dans un conteneur, il n'y a rien a rendre portable.

```
export DEBIAN_FRONTEND=noninteractive
```

```
export ROOT=/opt/GSM
```

OSMO_IMAGE=1 : « image de ce depot, son reseau est pose par start-docker.sh / compose » - network/osmo-ip-plan.sh ne touche alors a aucune adresse. Un conteneur quelconque (ubuntu:24.04 nu) ne l a pas et s adresse comme un natif.

```
export container=docker
export OSMO_IMAGE=1
export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig
export LD_LIBRARY_PATH=/usr/local/lib
export GIT_TERMINAL_PROMPT=0
```

2.26.1.3.1 apt-fast : les téléchargements apt en parallèle

Cette image installe plusieurs centaines de paquets ; apt les récupère un par un, apt-fast les met en parallèle via aria2. Le gain porte sur le téléchargement, pas sur dpkg — l'installation reste séquentielle.

UN SEUL installeur pour tous les environnements : packaging/apt-fast-install.sh (cette image, Dockerfile.stp, le chroot de l ISO, l hote via build.sh). Il pose aussi /etc/apt/apt.conf.d/90osmo-operator, les reglages de telechargement communs. Repli integre : sans GitHub, apt-fast appelle apt-get et le build continue, plus lentement.

LE CACHE APT DU BUILD. [2026-09-04] Chaque etape apt de ce fichier monte /var/cache/apt/archives en cache BuildKit (id osmo-apt-archives, partage avec Dockerfile.stp) : les .deb telecharges une fois y restent, sur l hote, dans le cache de BuildKit - une reconstruction ne retelecharge que ce qui a change. apt-fast-install retire docker-clean et pose Keep-Downloaded-Packages, sinon apt effacerait les paquets a peine installes. Necessite BuildKit (docker compose, ou docker-buildx : build.sh les installe) - le builder historique ne connait pas –mount.

```
mkdir -p "$(dirname /usr/local/sbin/apt-fast-install)" && cp -r "$REPO"/packaging/apt-fast-install.sh
↪  /usr/local/sbin/apt-fast-install
```

Le miroir Ubuntu mesure par build.sh (packaging/apt-mirror.sh) : vide = les sources de l image de base (archive.ubuntu.com), qui peut etre inutilisable depuis l hote (3 Ko/s le 2026-09-04) sans qu apt le dise.

```
export OSMO_UBUNTU_MIRROR=""


chmod 755 /usr/local/sbin/apt-fast-install && apt-fast-install \
    && rm -rf /var/lib/apt/lists/*


export DEBIAN_FRONTEND=noninteractive
```

2.26.1.3.2 1. Dépendances système — TOUTES ici, y compris celles qu'installait

```
   Dockerfile.run. Une seule liste, un seul endroit où la faire évoluer :
   l'image d'exécution n'a plus à connaître apt du tout.
```

tshark tire wireshark-common, qui pose une question debconf (capture non-root) et bloquerait un build non interactif : on y répond d'avance.

⚠ COUPLAGE EXTERIEUR — install_modules/10-deps.sh EXTRAIT cette liste du Dockerfile (« la liste fait autorite dans le Dockerfile ») avec

```
  awk '/^RUN apt-get update && apt-get install/,/[^\\]$/'
```

Ce motif ne connait QUE `apt-get` : depuis le passage a apt-fast, il ne capture plus que le bloc d'amorcage (aria2 curl ca-certificates) et l'installation NATIVE (hors docker) repart avec une liste tronquee — inst_deps_verify echoue sur build-essential/libtalloc-dev/libsctp-dev. Le correctif est dans 10-deps.sh (accepter `apt-(get|fast)` dans l'awk ET dans le sed qui suit), pas ici : ne pas repasser cette liste en apt-get pour contourner le probleme.

```
echo 'wireshark-common wireshark-common/install-setuid boolean true' | debconf-set-selections
```

deb-src AVANT l unique `apt-fast update` : le build-dep gnuradio (gr-gsm, fin du bloc ci-dessous) lit les index Sources. Il avait son propre RUN avec un deuxieme update + une deuxieme resolution apt : fusionne ici. Noble ecrit ses sources en deb822 (ubuntu.sources, ligne “Types: deb”) ; l ancien /etc/apt/sources.list (jammy) reste gere au cas ou la base change (gnuradio est dans universe : tous les composants).

```
if [ -f /etc/apt/sources.list.d/ubuntu.sources ]; then \
        sed -i 's/^Types: deb$/Types: deb deb-src/' /etc/apt/sources.list.d/ubuntu.sources; \
    else \
        sed -nE 's|^deb (http\S+) (\S+) .*|deb-src \1 \2 main restricted universe multiverse|p' \
            /etc/apt/sources.list | sort -u > /etc/apt/sources.list.d/deb-src.list; \
    fi


apt-fast update && apt-fast install -y --no-install-recommends \
    # Outils de build
    build-essential git gcc g++ make cmake autoconf automake libtool pkg-config wget curl \
    # Dépendances Osmocom Core & Network
    libtalloc-dev libpcsclite-dev libsctp-dev libmnl-dev liburing-dev asterisk-moh* \
    libdbi-dev libdbd-sqlite3 libsqlite3-dev sqlite3 libc-ares-dev libgnutls28-dev \
    # Audio, Radio & SIP
    libortp-dev libfftw3-dev libusb-1.0-0-dev libsofia-sip-ua-dev libsofia-sip-ua-glib-dev \
    # Python & Outils système
    python3 python3-dev python3-scapy ca-certificates tmux systemd systemd-sysv \
    # Debug — gdb-multiarch pour attacher au gdb-stub QEMU (ARM Calypso)
    gdb-multiarch \
    # ALSA — requis par osmo-gapk pour l'I/O audio matériel
    # (libasound2t64 : nom noble de libasound2, transition time_t 64 bits)
    libasound2-dev libasound2t64 alsa-utils \
    # libgsm — codec GSM-FR natif (accélère gapk en mode gsmfr)
    libgsm1-dev libgsm1 \
```

```
    iptables iproute2 asterisk ffmpeg \
    # Sync build-iso : psmisc (pkill/killall, cleanup) + pulseaudio (chaîne audio gapk/parec)
    psmisc pulseaudio pulseaudio-utils binutils-arm-none-eabi \
    # Toolchains alternatives : osmocom-bb jolly/testing et fixeria/burst_ind ne
    # compilent qu'avec gcc-9 ; gcc-11 reste le compilateur par defaut du reste
    # (gcc-13 est celui de noble : il compile la pile Osmocom en tete de fichier,
    # les alternatives ci-dessous ne sont posees qu apres).
    # (Ex-bloc apt dedie, supprime : les `update-alternatives` plus bas echouaient
    #  « alternative path /usr/bin/gcc-9 doesn't exist » sans ces paquets.)
    gcc-9 g++-9 gcc-11 g++-11 \
    # log4cpp : etait installe a part avec le build-dep gnuradio (gr-gsm).
    liblog4cpp5-dev \
    # QEMU Calypso (bbaranoff/qosmO) : venv + numpy/scipy pour l'outillage
    # DSP, glib/pixman/slirp pour la cible arm-softmmu, ninja pour meson, socat
    # pour les PTY de scripts/run.sh. (Ex-bloc apt-get juste avant le build QEMU.)
    python3-venv python3-pip python3-numpy python3-scipy \
    libglib2.0-dev libpixman-1-dev libslirp-dev socat ninja-build \
    # Ex-Dockerfile.run — outillage d'exploitation du conteneur :
    #   telnet   -> repli VTY (run_modules/21-abonnes-hlr.sh) + transport dashboard
    #   whiptail -> menus de tools/vty-menu.sh et de start-docker.sh
    #   xz-utils -> `tar -xJf` du tarball Node 22 (bloc dashboard, en fin de fichier)
    #   libasound2-plugins -> greffon ALSA->PulseAudio, EXIGE par configs/asound.conf
    #        (`type pulse` sur gsm_out, gsm_in ET pcm.!default) : sans lui, toute
    #        ouverture ALSA (gapk, mobile) echoue et l'audio est mort des 2 cotes
    #   tshark/tcpdump/libcap2-bin -> capture ; libcap2-bin fournit le setcap ci-dessous
    telnet nano whiptail xz-utils libasound2-plugins \
    tcpdump tshark libcap2-bin \
    # Deps GNU Radio 3.10 (gr-gsm, venv /root/.env plus bas) : meme resolution
    # apt que le reste, pas de second RUN apt. (Le && ferme la liste pour
    # l extracteur de install_modules/10-deps.sh.)
    && apt-fast build-dep -y gnuradio \
    && rm -rf /var/lib/apt/lists/*
```

Capture non-root pour le dashboard et tcpdump. Position OBLIGATOIRE : dumpcap n'existe qu'une fois wireshark-common installe (tire par tshark, juste au-dessus).

```
setcap cap_net_raw,cap_net_admin+eip "$(command -v dumpcap)"
```

2.26.1.3.3 Le cache .deb : l outil, puis les paquets deja construits

.deb-cache/ est le miroir, dans le contexte de build, de /var/cache/osmo-debs de l hote (build.sh l y recopie avant, et ramene les nouveaux paquets apres). Vide au premier build : tout se compile et se met en cache. Plein ensuite : chaque etape ci-dessous pose son paquet et ne compile rien. OSMO_DEB_REFRESH=1 (build.sh –no-cache) : le cache est ignore et reecrit.

```
export OSMO_DEB_REFRESH=0
```

```
mkdir -p "$(dirname /usr/local/sbin/osmo-deb)" && cp -r "$REPO"/packaging/osmo-deb.sh /usr/local/sbin/osmo-deb
```

```
mkdir -p /var/cache/osmo-debs/ && cp -r "$REPO"/.deb-cache/ /var/cache/osmo-debs/
```

```
chmod 755 /usr/local/sbin/osmo-deb && osmo-deb list || true
```

2.26.1.3.4 git : plus de forçage HTTP/1.1

[2026-09-03] RETIRE. Le contournement http.version=HTTP/1.1 (contre “expected flush after ref listing”, un flux HTTP/2 coupe par un intermediaire) etait pose ici en global et repete a chaque clone. Il n a plus lieu d etre : git parle HTTP/2 normalement, et forcer HTTP/1.1 ralentissait tous les clones pour un incident de reseau local. GIT_TERMINAL_PROMPT=0 (ENV, plus haut) reste : sans terminal, un clone qui veut un login doit ECHOUER net, pas attendre. postBuffer large pour les gros clones.

```
git config --global http.postBuffer 524288000
```

Metadonnee Docker, sans equivalent sur une machine : `SHELL ["/bin/bash", "-c"]`.

```
mkdir -p /etc/asterisk/ && cp -r "$REPO"/configs/*conf /etc/asterisk/
```

asound.conf DOIT AUSSI ETRE A LA RACINE. Le COPY ci-dessus le depose dans /etc/asterisk/ - ALSA ne regarde jamais la : il lit /etc/asound.conf (ou ~/.asoundrc). Sans ce second exemplaire, `aplay -D gsm_out` repond

```
    ALSA lib pcm.c: (snd_pcm_open_noupdate) Unknown PCM gsm_out
```

et le mobile du conteneur est MUET, sans qu aucun log ne parle d audio. start-docker.sh l.469 monte bien celui de l hote (-v …:/etc/asound.conf), mais un conteneur lance autrement - docker run a la main, diagnostic - n a rien.

```
mkdir -p "$(dirname /etc/asound.conf)" && cp -r "$REPO"/configs/asound.conf /etc/asound.conf


cd ${ROOT}
```

2.26.1.3.5 2. Création de l’utilisateur osmocom

```
groupadd osmocom && useradd -r -g osmocom -s /sbin/nologin -d /var/lib/osmocom osmocom && \
    mkdir -p /var/lib/osmocom && chown osmocom:osmocom /var/lib/osmocom
```

2.26.1.4 Etape 2 : osmo-core

STAGE osmo-core — la pile Osmocom : 17 depots, 4 patchs, gapk, SMSC, libosmo-dsp

Dans Docker : `FROM base`. *Sur une machine, on continue dans le meme systeme.*

ARG ne traverse pas un FROM : les stages qui appellent osmo-deb ou ${ROOT} doivent les redeclarer, sinon ROOT est vide et OSMO_DEB_REFRESH=1 (build.sh –no-cache) ne descend pas jusqu ici.

```
export ROOT=/opt/GSM


export OSMO_DEB_REFRESH=0
```

2.26.1.4.1 3. Compilation de la pile Osmocom (Ordre respecté)

```
for repo in \
    libosmocore:1.12.1 \
    libosmo-netif:1.7.0 \
    libosmo-abis:2.1.0 \
    libosmo-sigtran:2.2.1 \
    libsmpp34:1.14.5 \
    libgtpnl:1.3.3 \
    osmo-hlr:1.9.2 \
    osmo-mgw:1.15.0 \
    osmo-ggsn:1.14.0 \
    osmo-sgsn:1.13.1 \
    osmo-msc:1.15.0 \
    osmo-bsc:1.14.0 \
    osmo-trx:1.7.2 \
    osmo-bts:1.10.0 \
    osmo-pcu:1.5.2 \
    osmo-sip-connector:1.7.2 \
    libosmo-gprs:0.2.1; \
    do \
    name=$(echo $repo | cut -d: -f1) && \
    version=$(echo $repo | cut -d: -f2) && \
    \
    if [[ "$name" =~ ^libosmo ]]; then \
        GIT_URL="https://gitea.osmocom.org/osmocom/$name"; \
    else \
        GIT_URL="https://gitea.osmocom.org/cellular-infrastructure/$name"; \
    fi && \
    \
    # Le paquet du cache d abord : s il est la, ni clone ni compilation.
    if osmo-deb install "$name" "$version"; then continue; fi && \
    \
    cd ${ROOT} && \
    git clone "$GIT_URL" && cd "$name" && \
    git checkout "$version" && \
    \
    autoreconf -fi && \
    EXTRA_FLAGS="" && \
    if [ "$name" = "libosmo-abis" ]; then EXTRA_FLAGS="--disable-dahdi"; fi && \
    if [ "$name" = "osmo-msc" ]; then EXTRA_FLAGS="--enable-smpp"; fi && \
    if [ "$name" = "osmo-mgw" ]; then EXTRA_FLAGS="--enable-alsa"; fi && \
    if [ "$name" = "osmo-trx" ]; then EXTRA_FLAGS="--with-ipc"; fi && \
    if [ "$name" = "osmo-bts" ]; then EXTRA_FLAGS="--enable-virtual --enable-trx"; fi && \
    if [ "$name" = "osmo-ggsn" ]; then EXTRA_FLAGS="--enable-gtp-linux"; fi && \
    \
    ./configure $EXTRA_FLAGS && \
    make -j$(nproc) && \
    # make install sous DESTDIR -> .deb dans le cache -> dpkg -i dans la racine
    osmo-deb pack "$name" "$version" make install && \
    ldconfig \
    || { echo "ECHEC build osmocom: $name"; exit 1; }; \
    done
```

2.26.1.4.2 Patch osmo-trx IPC : alignement ts_initial sur la trame TDMA (fix RACH/LU)

Le device IPC (calypso-ipc-device) commit des buffers RX a des timestamps multiples de 2500 (CALYPSO_SHM_BUFSIZE) ; sans arrondi, ts_initial%5000 valait 0 ou 2500 -> device-TN0 mappe sur osmo-TN4 -> la RACH (UL) ratait le slot TS0/RACH -> NOPE/-110 -> Location Update bloquee. Le patch arrondit ts_initial a la frontiere de trame (8*625=5000 @ 4 SPS). Patch maintenu dans patches/.

2.26.1.4.3 Patch osmo-trx RACH UL : table de modulation per-RA (fix LU, no-hardcode)

La vraie RA du mobile (d_rach@0x0474, plombee via /dev/shm/calypso_rach par QEMU) varie a chaque burst ; le device rejouait un RA=3 fixe -> request-reference de l'IMM ASSIGN jamais matchee -> LU en boucle. sigProcLib pre-genere la modulation Laurent EXACTE d'osmo-trx pour chaque RA 0x00..0x0f (-> /root/rach_ref_RA.cs16), le device selectionne le ref de la VRAIE RA. Ajoute aussi les logs RACH-DET (Transceiver.cpp) + le lien libosmocoding dans COMMON_LDADD (Makefile.am, requis pour gsm0503_rach_ext_encode). Touche Makefile.am -> autoreconf+configure requis.

```
mkdir -p "$(dirname /tmp/osmo-trx-ipc-ts-frame-align.patch)" && cp -r
 ↪  "$REPO"/patches/osmo-trx-ipc-ts-frame-align.patch /tmp/osmo-trx-ipc-ts-frame-align.patch


mkdir -p "$(dirname /tmp/osmo-trx-rach-per-ra-table.patch)" && cp -r
 ↪  "$REPO"/patches/osmo-trx-rach-per-ra-table.patch /tmp/osmo-trx-rach-per-ra-table.patch
```

Meme nom de paquet que dans la boucle (osmo-trx), version 1.7.2+ipc : dpkg fait une mise a jour, les fichiers sont les memes. Si la boucle est sortie du cache, l arbre source n existe pas : on le reclone avant d appliquer les patchs.

```
if ! osmo-deb install osmo-trx 1.7.2+ipc; then \
      { [ -d ${ROOT}/osmo-trx ] || { cd ${ROOT} \
          && git clone https://gitea.osmocom.org/cellular-infrastructure/osmo-trx \
          && git -C osmo-trx checkout 1.7.2; }; } \
      && git -C ${ROOT}/osmo-trx apply /tmp/osmo-trx-ipc-ts-frame-align.patch \
      && git -C ${ROOT}/osmo-trx apply /tmp/osmo-trx-rach-per-ra-table.patch \
      && cd ${ROOT}/osmo-trx \
      && autoreconf -fi \
      && ./configure --with-ipc \
      && make -j$(nproc) \
      && osmo-deb pack osmo-trx 1.7.2+ipc make install \
      && ldconfig; \
    fi
```

2.26.1.4.4 Patch osmo-bts RAND : forcer le defi d'authentification

osmo-bts relaie normalement l'AUTHENTICATION REQUEST du reseau telle quelle. Ce patch, dans rsl_rx_rll() (src/common/rsl.c), reecrit le RAND de cette trame quand la variable d'environnement RAND est posee sur le processus BTS :

```
    RAND non defini -> rien de touche (comportement normal) ;
    RAND=0          -> RAND nul sur 16 octets (defi fixe COMP128v1,
                       defi fixe connu, Ki = prefixe connu
                       + 2 octets a retrouver) ;
    RAND=<hexa>     -> jusqu'a 16 octets lus en hexa, completes de zeros.
```

« seulement quand precise, mais force quand precise ». Ne touche que rsl.c, donc pas de Makefile.am : autoreconf/configure quand meme, l'arbre reclone arrive sans ./configure. Meme nom de paquet (osmo-bts), version 1.10.0+rand : dpkg fait une mise a jour. Patch maintenu dans patches/.

```
mkdir -p "$(dirname /tmp/osmo-bts-force-rand-toy.patch)" && cp -r "$REPO"/patches/osmo-bts-force-rand-toy.patch
 ↪  /tmp/osmo-bts-force-rand-toy.patch


if ! osmo-deb install osmo-bts 1.10.0+rand; then \
      { [ -d ${ROOT}/osmo-bts ] || { cd ${ROOT} \
          && git clone https://gitea.osmocom.org/cellular-infrastructure/osmo-bts \
          && git -C osmo-bts checkout 1.10.0; }; } \
      && git -C ${ROOT}/osmo-bts apply /tmp/osmo-bts-force-rand-toy.patch \
      && cd ${ROOT}/osmo-bts \
      && autoreconf -fi \
      && ./configure --enable-virtual --enable-trx \
      && make -j$(nproc) \
      && osmo-deb pack osmo-bts 1.10.0+rand make install \
      && ldconfig; \
    fi
```

2.26.1.4.5 Patch osmo-hlr RAND : forcer le RAND du vecteur d'auth

auc.c/auc_compute_vectors() : gate minimal. Si la variable d'environnement RAND est posee sur le process HLR, le RAND tire pour CHAQUE vecteur est force a 0 (16 octets) AVANT osmo_auth_gen_vec2 : SRES et Kc sont alors calcules sur RAND=0 (le HLR a le Ki). C'est l'etage indispensable pour une LU ACCEPTEE : osmo-msc envoie ce RAND et compare au SRES du meme vecteur -> match. Le seul gate osmo-msc (cote tx) donnerait un mismatch -> LU reject. RAND absent -> reseau normal. Ne touche pas Makefile.am ; autoreconf/configure quand meme (arbre reclone sans ./configure). Meme nom de paquet (osmo-hlr), version 1.9.2+rand : dpkg met a jour. Patch maintenu dans patches/.

```
mkdir -p "$(dirname /tmp/osmo-hlr-force-rand-toy.patch)" && cp -r "$REPO"/patches/osmo-hlr-force-rand-toy.patch
↪  /tmp/osmo-hlr-force-rand-toy.patch


if ! osmo-deb install osmo-hlr 1.9.2+rand; then \
      { [ -d ${ROOT}/osmo-hlr ] || { cd ${ROOT} \
          && git clone https://gitea.osmocom.org/cellular-infrastructure/osmo-hlr \
          && git -C osmo-hlr checkout 1.9.2; }; } \
      && git -C ${ROOT}/osmo-hlr apply /tmp/osmo-hlr-force-rand-toy.patch \
      && cd ${ROOT}/osmo-hlr \
      && autoreconf -fi \
      && ./configure \
      && make -j$(nproc) \
      && osmo-deb pack osmo-hlr 1.9.2+rand make install \
      && ldconfig; \
    fi
```

2.26.1.4.6 Patch osmo-msc RAND : forcer le defi d'authentification

gsm48_tx_mm_auth_req() (src/libmsc/gsm_04_08.c) : gate minimal. Si la variable d'environnement RAND est posee sur le processus MSC, le RAND de l'AUTH REQUEST descendante est force (0 sur 16 octets dans notre cas ; changer l'initialiseur forced_rand[] pour un autre RAND). RAND absent -> comportement normal du reseau. Ne touche pas Makefile.am ; autoreconf/configure quand meme (arbre reclone sans ./configure). Meme nom de paquet (osmo-msc), version 1.15.0+rand : dpkg met a jour. Patch maintenu dans patches/. Meme gate cote BSC (openbsc gsm_04_08.c).

```
mkdir -p "$(dirname /tmp/osmo-msc-force-rand-toy.patch)" && cp -r "$REPO"/patches/osmo-msc-force-rand-toy.patch
↪  /tmp/osmo-msc-force-rand-toy.patch


if ! osmo-deb install osmo-msc 1.15.0+rand; then \
      { [ -d ${ROOT}/osmo-msc ] || { cd ${ROOT} \
          && git clone https://gitea.osmocom.org/cellular-infrastructure/osmo-msc \
          && git -C osmo-msc checkout 1.15.0; }; } \
      && git -C ${ROOT}/osmo-msc apply /tmp/osmo-msc-force-rand-toy.patch \
      && cd ${ROOT}/osmo-msc \
      && autoreconf -fi \
      && ./configure --enable-smpp \
      && make -j$(nproc) \
      && osmo-deb pack osmo-msc 1.15.0+rand make install \
      && ldconfig; \
    fi
```

2.26.1.4.7 Patch gapk : sonde sur la sortie ALSA (GAPK_ALSA_PROBE)

Diagnostic pur, inerte tant que GAPK_ALSA_PROBE != 1. Tranche une question que rien d'autre ne permet de trancher : quand l'ecouteur du MS est muet, le PCM remis a snd_pcm_writei est-il deja nul (defaut en amont, dans la chaine gapk) ou non nul (defaut en aval, ALSA/PulseAudio) ? osmo_gapk_pq_execute ne journalise rien ici, car il ne se plaint que d'un retour negatif et pq_cb_alsa_output rend « succes » meme pour une ecriture vide. Patch maintenu dans patches/ (regenere si pq_alsa.c change).

[2026-08-12] LE PATCH RESTE DANS LE DEPOT, LE BUILD NE L'APPLIQUE PLUS. Demande explicite : garder patches/gapk-pq-alsa-output-probe.patch versionne (il tranche encore la question amont/aval quand

l'ecouteur est muet) mais ne plus le poser sur le gapk construit — un `git apply` dans le build casse l'image des que pq_alsa.c bouge en amont, et la sonde n'est plus la question du jour (descendant muet resolu par CALYPSO_PULSE_LATENCY_MSEC, 240 depuis le 2026-09-09 : cf. start-native.sh, le tampon du greffon ALSA-PulseAudio). POUR LE REMETTRE, deux gestes, dans cet ordre :

```
1. decommenter la ligne COPY ci-dessous ;
2. reinserer dans le RUN, entre le `git clone` et le `cd osmo-gapk` :
     git -C ${ROOT}/osmo-gapk apply /tmp/gapk-pq-alsa-output-probe.patch && \
   (elle ne peut pas rester en commentaire : une ligne `#` au milieu d'une
    continuation `\` couperait la chaine shell du RUN).
```

Verifier d'abord que le patch s'applique toujours :

```
git -C <clone gapk> apply --check patches/gapk-pq-alsa-output-probe.patch
```

POUR L'APPLIQUER A CHAUD SANS REBUILD : le poser dans le conteneur sur un clone de gapk, puis reconstruire gapk seul. COPY patches/gapk-pq-alsa-output-probe.patch /tmp/gapk-pq-alsa-output-probe.patch

```
if ! osmo-deb install osmo-gapk 0.git; then \
      cd ${ROOT} && \
      git clone https://gitea.osmocom.org/osmocom/gapk osmo-gapk && \
      cd osmo-gapk && \
      autoreconf -fi && \
      ./configure --enable-alsa && \
      make -j$(nproc) && \
      osmo-deb pack osmo-gapk 0.git make install && \
      ldconfig; \
    fi
```

2.26.1.4.8 gsup-smsc-proto : SMSC externe connecté à OsmoHLR via GSUP

Programmes : proto-smsc-daemon (réception MO SMS + relai MT via GSUP)

```
            proto-smsc-sendmt (injection MT SMS via socket UNIX local)
```

Dépendances build : libosmocore, libosmogsm, libosmo-gsup-client [2026-09-03] Son Makefile de tete lance `make DESTDIR= install` dans daemon/ et sendmt/ - DESTDIR force a vide, le staging d osmo-deb restait vide ("rien n a ete installe dans DESTDIR") alors que la compilation (gcc courant) etait bonne. On installe donc les deux sous-repertoires nous-memes, avec le DESTDIR qu osmo-deb pose : leurs Makefiles, eux, l honorent.

```
if ! osmo-deb install gsup-smsc-proto 0.git; then \
      cd ${ROOT} && \
      git clone https://gitea.osmocom.org/themwi/gsup-smsc-proto && \
      cd gsup-smsc-proto && \
      ./configure --with-osmo=/usr/local && \
      make -j$(nproc) && \
      osmo-deb pack gsup-smsc-proto 0.git \
        sh -c 'for i in daemon sendmt; do make -C "$i" DESTDIR="$DESTDIR" install || exit 1; done' && \
      ldconfig; \
    fi
```

2.26.1.4.9 sms-coding-utils : encodage/décodage SMS PDU (GSM 03.40)

sms-encode-text, gen-sms-deliver-pdu, sms-pdu-decode, etc. Son Makefile de tete pose `DESTDIR=` (vide) et le repasse aux sous-repertoires : une variable donnee SUR LA LIGNE DE COMMANDE de make l emporte et descend avec, d ou `make install DESTDIR=...` explicite (l ancien INSTDIR=… n existait dans aucun de ses Makefiles). bindir vaut /usr/local/bin par configure.

```
if ! osmo-deb install sms-coding-utils 0.r1; then \
      cd ${ROOT} && \
      wget -q https://www.freecalypso.org/pub/GSM/FreeCalypso/sms-coding-utils-latest.tar.bz2 && \
      tar xf sms-coding-utils-latest.tar.bz2 && \
      cd sms-coding-utils-r1 && \
      ./configure && \
      make -j$(nproc) && \
      osmo-deb pack sms-coding-utils 0.r1 \
        sh -c 'mkdir -p "$DESTDIR/usr/local/bin" && make install DESTDIR="$DESTDIR"'; \
    fi
```

2.26.1.4.10 libosmo-dsp (dépendance transceiver/burst_ind)

```
if ! osmo-deb install libosmo-dsp 0.git; then \
      cd /opt/GSM \
      && git clone https://gitea.osmocom.org/sdr/libosmo-dsp.git \
      && cd libosmo-dsp \
      && autoreconf -fi \
      && ./configure \
      && make -j$(nproc) \
      && osmo-deb pack libosmo-dsp 0.git make install \
      && ldconfig; \
    fi
```

2.26.1.5 Etape 3 : bb

STAGE bb — osmocom-bb, le firmware Calypso et les deux branches gcc-9

Dans Docker : `FROM osmo-core`. Sur une machine, on continue dans le meme systeme.

ARG ne traverse pas un FROM : les stages qui appellent osmo-deb ou ${ROOT} doivent les redeclarer, sinon ROOT est vide et OSMO_DEB_REFRESH=1 (build.sh –no-cache) ne descend pas jusqu ici.

```
export ROOT=/opt/GSM
```

```
export OSMO_DEB_REFRESH=0
```

2.26.1.5.1 Calypso build

2.26.1.5.2 Patch osmocon : filtre Kc — RETIRE le 2026-08-30

Ce patch faisait ecrire /dev/shm/calypso_kc a osmocon, en espionnant L1CTL_CRYPTO_REQ au passage sur le lien serie. Il etait la SEULE source du Kc a l'epoque ; il ne l'est plus, et il faisait desormais plus de mal que de bien.

POURQUOI ON LE RETIRE. Un espion voit passer la cle, mais pas sa fin de vie : il doit la DEVINER. Le patch la devinait sur DM_EST_REQ et DM_REL_REQ, et remettait le fichier a 32 zeros. L'intention est juste pour le RELEASE, fausse pour l'ESTABLISH – le mobile emet aussi un DM_EST_REQ pour ouvrir un lien SUPPLEMENTAIRE sur un canal DEJA chiffre (le SAPI 3 du SMS). La cle disparaissait donc en pleine session chiffree.

Pire, il y avait TROIS ecrivains sur ce fichier (osmocon, l1ctl_sock.c, le shunt DSP) sans le moindre arbitrage : le dernier qui ecrit gagne, et c'est l'effaceur. Mesure du 30/08 : fichier a 32 zeros, « A5=non » sur tout le canal dedie, premier appel mort.

CE QUI LE REMPLACE. calypso_dsp_shunt.c publie /dev/shm/calypso_kc_l1 depuis d_a5mode et a_kc du NDB – ce que le FIRMWARE a reellement charge dans le DSP (calypso/dsp.c:dsp_load_ciph_param). Ce n'est plus un espion qui devine, c'est l'etat de la couche 1, remis a zero par le firmware lui-meme quand il repart en clair. Et le fichier lui est PROPRE : plus de course a l'ecriture.

Le patch reste dans patches/ a titre documentaire, il n'est plus applique. L arbre ENTIER part dans le paquet (snapshot) : trx_toolkit, osmocon et les binaires host sont lus dans l arbre au runtime, pas seulement dans /usr/local. [2026-09-08] trx_toolkit/clck_gen.py : l horloge TDMA de fake_trx ne se remet a zero que pour un retard superieur a une trame (4,615 ms). Avant, un retard de 0,3 a 2 ms decalait la base de temps a chaque fois ; osmo-bts-trx compensait en boucle et la parole perdait des trames. Patch maintenu dans patches/, applique aussi par install_modules/40-patches.sh en natif. mobile : une SIM reinseree sans RPLMN (reconnexion SAP) restait en A6 et la recherche de PLMN finissait « unhandled » (patches/osmocom-bb-plmn-a6-reinsert.patch). PLMNsel / FPLMN lus avec osmo_plmn_to_bcd (ecriture) au lieu de from_bcd : 000-00 (patches/osmocom-bb-sim-plmn-from-bcd.patch). Les deux, aussi, par 40-patches.sh.

```
mkdir -p "$(dirname /tmp/osmocom-bb-clck-gen-frame-tolerance.patch)" && cp -r
 ↪  "$REPO"/patches/osmocom-bb-clck-gen-frame-tolerance.patch /tmp/osmocom-bb-clck-gen-frame-tolerance.patch


mkdir -p "$(dirname /tmp/osmocom-bb-plmn-a6-reinsert.patch)" && cp -r
 ↪  "$REPO"/patches/osmocom-bb-plmn-a6-reinsert.patch /tmp/osmocom-bb-plmn-a6-reinsert.patch


mkdir -p "$(dirname /tmp/osmocom-bb-sim-plmn-from-bcd.patch)" && cp -r
 ↪  "$REPO"/patches/osmocom-bb-sim-plmn-from-bcd.patch /tmp/osmocom-bb-sim-plmn-from-bcd.patch


if ! osmo-deb install osmocom-bb 0.git; then \
      cd ${ROOT} && \
      git clone https://gitea.osmocom.org/phone-side/osmocom-bb && \
      git -C ${ROOT}/osmocom-bb apply /tmp/osmocom-bb-clck-gen-frame-tolerance.patch && \
      git -C ${ROOT}/osmocom-bb apply /tmp/osmocom-bb-plmn-a6-reinsert.patch && \
      git -C ${ROOT}/osmocom-bb apply /tmp/osmocom-bb-sim-plmn-from-bcd.patch && \
      cd osmocom-bb/src && \
      # Build complet : firmware (layer1.bin/.elf pour Calypso) + outils host
      # (mobile, trxcon, virtphy, ccch_scan). Le firmware est nécessaire pour
      # le mode PHY_MODE=qemu où QEMU émule un Calypso et exécute layer1.
      make nofirmware && \
      osmo-deb snapshot osmocom-bb 0.git ${ROOT}/osmocom-bb; \
    fi
```

2.26.1.5.3 Note historique — patch fake_trx TRXD v0 (RETIRE)

Conserve verbatim : il documente une panne vecue. Il flottait dans Dockerfile.run juste au-dessus du bloc firmware, avec lequel il n'a AUCUN rapport — voisinage accidentel, pas lien de cause a effet.

2.26.1.5.4 Patch fake_trx : forcer TRXD v0

osmo-bts-trx 1.10+ négocie TRXD v1 via SETFORMAT, mais trxcon (OsmocomBB) ne supporte que v0. Le header v1 a 2 octets de plus → décalage des soft-bits → BER constant 55/456 sur chaque frame → FBSB_SEARCH échoue en boucle. Fix : forcer ver_req=0 dans la réponse SETFORMAT → le BTS reste en v0. (patch TRXD v0 retire : burst_fwd.py:66 re-encode deja par interface,

```
 trxcon recoit du v0 quoi qu'il arrive -> le forcage etait inutile)
```

2.26.1.5.5 Firmware Calypso prebuild (layer1.highram.*) — ex-Dockerfile.run

`make nofirmware` ci-dessus ne construit justement PAS le firmware : on prend les binaires prebuild de bbaranoff/firmware et on les depose la ou le mode PHY_MODE=qemu les cherche. ORDRE IMPOSE : ce bloc exige A LA FOIS ce clone ET l'arbre /opt/GSM/osmocom-bb ci-dessus — le placer plus haut ferait echouer les cp et PHY_MODE=qemu partirait sans layer1. (Le `rm -rf /opt/GSM/firmware` qui precedait le clone dans Dockerfile.run est

```
 supprime : ici le chemin n'existe pas encore, l'instruction etait morte.)
```

GIT_TERMINAL_PROMPT=0 : sans terminal (docker build), un clone qui demande un login doit echouer NET (message clair) au lieu d attendre un identifiant que personne ne tapera. –depth 1 : on ne veut que les binaires prebuild, pas l historique. (Le -c http.version=HTTP/1.1 qui etait ici est retire.)

```
if ! osmo-deb install calypso-firmware 0.git; then \
      GIT_TERMINAL_PROMPT=0 git clone --depth 1 \
        https://github.com/bbaranoff/firmware /opt/GSM/firmware \
      && osmo-deb snapshot calypso-firmware 0.git /opt/GSM/firmware; \
    fi
```

[2026-08-28] Les trois `cp` vers /opt/GSM/osmocom-bb/src/target/firmware qui suivaient sont SUPPRIMES. Ils entretenaient un deuxieme exemplaire du firmware que plus personne ne lit : depuis la normalisation, environnement/paths.env du depot qemu ne connait qu'un chemin, $GSM_ROOT/firmware, et c'est celui que QEMU (-kernel .elf) comme osmocon (romload .bin) recoivent. Verifie sur le banc 192.168.1.7 : les deux exemplaires etaient bit a bit identiques (md5 3363105f…), et seul /opt/GSM/firmware etait charge. Si vous recompilez le firmware dans osmocom-bb, deposez le resultat dans /opt/GSM/firmware : c'est desormais le seul endroit consulte.

2.26.1.5.6 GCC 9 pour osmocom-bb branches expérimentales (jolly/testing, burst_ind)

gcc-9 et gcc-11 sont installés avec le reste, plus haut : ici on ne fait que déclarer les alternatives, dont l'ordre compte pour osmocom-bb.

```
update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-9 90 \
       --slave /usr/bin/g++ g++ /usr/bin/g++-9 \
    && update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-11 110 \
       --slave /usr/bin/g++ g++ /usr/bin/g++-11


update-alternatives --set gcc /usr/bin/gcc-9
```

osmocom-bb jolly/testing → transceiver (BTS soft-SDR pour Calypso) Le binaire ET l arbre partent dans le paquet : les sources restent dans /opt/GSM au rebuild depuis le cache (voir l en-tete, CACHE .deb). [2026-10-19] patches/osmocom-bb-transceiver-qemu-bench.patch : l appli hote `transceiver` sert le Calypso EMULE (start-native.sh –osmo-trx, c54x_exe/run.sh OSMO_TRX=1) : –no-sync (pas de cellule a ecouter, mode BTS des le reset), –ts-mask (un seul timeslot), –l1ctl/–base-port, et SETTSC/SETFORMAT/ NOMTXPOWER acquiesces (osmo-bts-trx >= 1.3 les tient pour critiques). Applique aussi par install_modules/40-patches.sh en natif.

```
mkdir -p "$(dirname /tmp/osmocom-bb-transceiver-qemu-bench.patch)" && cp -r
 ↪  "$REPO"/patches/osmocom-bb-transceiver-qemu-bench.patch /tmp/osmocom-bb-transceiver-qemu-bench.patch


if ! osmo-deb install osmocom-bb-transceiver 0.git; then \
      git clone --branch jolly/testing --depth 1 \
        https://gitea.osmocom.org/phone-side/osmocom-bb.git \
        /opt/GSM/osmocom-bb-transceiver \
      && git -C /opt/GSM/osmocom-bb-transceiver apply /tmp/osmocom-bb-transceiver-qemu-bench.patch \
      && cd /opt/GSM/osmocom-bb-transceiver/src \
      && make HOST_layer23_CONFARGS=--enable-transceiver nofirmware -j$(nproc) \
      && cp /opt/GSM/osmocom-bb-transceiver/src/host/layer23/src/transceiver/transceiver \
         /usr/local/bin/transceiver \
```

```
      && osmo-deb snapshot osmocom-bb-transceiver 0.git /usr/local/bin/transceiver \
             /opt/GSM/osmocom-bb-transceiver; \
    fi
```

osmocom-bb fixeria/burst_ind → ccch_scan / bcch_scan / cell_log

```
if ! osmo-deb install osmocom-bb-burst-ind 0.git; then \
      git clone --branch fixeria/burst_ind --depth 1 \
        https://gitea.osmocom.org/phone-side/osmocom-bb.git \
        /opt/GSM/osmocom-bb-burst_ind \
      && cd /opt/GSM/osmocom-bb-burst_ind/src \
      && make nofirmware -j$(nproc) \
      && cp /opt/GSM/osmocom-bb-burst_ind/src/host/layer23/src/misc/ccch_scan \
         /usr/local/bin/ccch_scan \
      && { cp /opt/GSM/osmocom-bb-burst_ind/src/host/layer23/src/misc/bcch_scan \
         /usr/local/bin/bcch_scan 2>/dev/null || true; } \
      && { cp /opt/GSM/osmocom-bb-burst_ind/src/host/layer23/src/misc/cell_log \
         /usr/local/bin/cell_log 2>/dev/null || true; } \
      && osmo-deb snapshot osmocom-bb-burst-ind 0.git /opt/GSM/osmocom-bb-burst_ind \
           $(ls /usr/local/bin/ccch_scan /usr/local/bin/bcch_scan /usr/local/bin/cell_log 2>/dev/null); \
    fi
```

Retour au compilateur par defaut : ce stage ne sert qu a produire des .deb, mais le laisser sur gcc-9 piegerait quiconque en derive une etape.

```
update-alternatives --set gcc /usr/bin/gcc-11
```

2.26.1.6 Etape 4 : qemu

STAGE qemu — le seul arbre QEMU : /opt/GSM/qosmo (–enable-l1-grgsm)

Dans Docker : `FROM base`. Sur une machine, on continue dans le meme systeme.

ARG ne traverse pas un FROM : les stages qui appellent osmo-deb ou ${ROOT} doivent les redeclarer, sinon ROOT est vide et OSMO_DEB_REFRESH=1 (build.sh –no-cache) ne descend pas jusqu ici.

```
export ROOT=/opt/GSM
```

```
export OSMO_DEB_REFRESH=0
```

QEMU Calypso — RAN virtuel (baseband émulé)

Architecture :

```
  - QEMU émule un SoC Calypso (ARM7TDMI + DSP TMS320C54x)
  - L'ARM exécute le vrai firmware osmocom-bb layer1.highram.elf
  - Le DSP charge le ROM réel (calypso_dsp.txt) au boot
  - bridge.py relaie les bursts entre osmo-bts-trx (UDP 5700-5702)
    et la BSP du DSP (UDP 6702), avec QEMU comme maître d'horloge TDMA
  - Le mobile (layer23) se connecte directement au socket L1CTL
    publié par le firmware via la PTY série de QEMU
```

Voir scripts/run.sh PHY_MODE=qemu pour l'orchestration runtime.

(Le ‘apt-get install python3-venv python3-pip python3-numpy python3-scipy

```
 libglib2.0-dev libpixman-1-dev libslirp-dev socat ninja-build` qui etait ici
 a ete fusionne dans la liste apt-fast en tete de fichier : une seule liste,
 un seul endroit ou la faire evoluer. Aucun paquet perdu.)
```

[2026-09-25] qosmo + c54x_exe + grgsm_exe — remplacent qosmo-grgsm et qosmo-dsp

Les deux forks QEMU sont retires (cf. start-native.sh, environment/paths.env) :

```
  /opt/GSM/qosmo      UN seul arbre QEMU (bbaranoff/qosm0), --enable-l1-grgsm.
                      Il porte run.sh, run_modules, cfgs. Le meme binaire sert
                      au mode --dsp : CALYPSO_DSP_EXTERN=1 y coupe la L1 gr-gsm.
                      `ninja install` pose aussi le lanceur C `qosmo`.
  /opt/GSM/c54x_exe   le C54x HORS de QEMU (mask-ROM TI, sa ROM dans rom/).
                      Il compile les sources de /opt/GSM/qosmo : APRES qosmo.
  /opt/GSM/grgsm_exe  la couche 1 gr-gsm hors QEMU, memes sources qosmo.
```

Snapshot de l arbre qosmo ENTIER, build/ compris : QEMU_BIN pointe sur $OQC_ROOT/build/qemu-system-arm, et QEMU lit build/qemu-bundle pour se relocaliser (voir Dockerfile.lite). L ancien RUN « /opt/GSM/qemu/{build,*.py} » et calypso-ipc-device disparaissent avec qosmo-grgsm : plus rien ne les lit.

```
if ! osmo-deb install qosmo 0.git; then \
      git clone https://github.com/bbaranoff/qosm0 /opt/GSM/qosmo \
      && cd /opt/GSM/qosmo \
      && python3 -m venv /root/.venv-qemu \
      && . /root/.venv-qemu/bin/activate \
      && pip install --no-cache-dir tomli \
      && mkdir -p build && cd build \
      && ../configure --target-list=arm-softmmu --enable-l1-grgsm \
             --prefix=/opt/GSM/qemu-install --disable-werror --disable-docs \
      && make -j$(nproc) \
      && make install \
      && cp /opt/GSM/qemu-install/bin/qemu-system-arm /usr/local/bin/qemu-system-arm \
      && cp /opt/GSM/qemu-install/bin/qosmo /usr/local/bin/qosmo \
      && osmo-deb snapshot qosmo 0.git /opt/GSM/qosmo /opt/GSM/qemu-install \
             /root/.venv-qemu /usr/local/bin/qemu-system-arm /usr/local/bin/qosmo; \
    fi
```

2.26.1.7 Etape 5 : l1

STAGE l1 — les deux couches 1 hors QEMU : c54x_exe et grgsm_exe

Dans Docker : `FROM qemu`. Sur une machine, on continue dans le meme systeme.

ARG ne traverse pas un FROM : les stages qui appellent osmo-deb ou ${ROOT} doivent les redeclarer, sinon ROOT est vide et OSMO_DEB_REFRESH=1 (build.sh –no-cache) ne descend pas jusqu ici.

```
export ROOT=/opt/GSM
```

```
export OSMO_DEB_REFRESH=0
```

Elles compilent les sources de /opt/GSM/qosmo (deja la, ce stage en derive) et se lient a libosmocoding/libosmocore pour c54x_exe. On rapatrie donc /usr/local depuis osmo-core plutot que de faire deriver ce stage de lui : copier /usr/local (quelques centaines de Mo) coute bien moins que de copier l arbre de build de QEMU (~1,5 Go d objets) dans l autre sens.

Copie entre etapes Docker (`--from=osmo-core /usr/local /usr/local`) : sur une seule machine, `/usr/local` est deja en place.

```
ldconfig
```

c54x_exe : le Makefile met -march=native dans CFLAGS - dans une image qui tourne sur d autres CPU que celui du build, c est un SIGILL au demarrage. On garde ses drapeaux, sans celui-la. La ROM est aussi posee en /opt/GSM/calypso_dsp.*.bin, le –rom-dir par defaut de c54x_exe. [2026-10-10] La ROM n est plus dans le depot c54x_exe : le `cp rom/...` tombait sur « cannot stat rom/calypso_dsp.txt ». rom/fetch-rom.sh telecharge le dump FreeCalypso 3606, le convertit en .bin et verifie SHA256SUMS.3606 ; calypso_dsp.txt, c est le dump brut lui-meme, qu il ne garde pas : on le telecharge a part.

```
export FREECALYPSO_URL=ftp://ftp.freecalypso.org/pub/GSM/Calypso
```

```
if ! osmo-deb install c54x-exe 0.git; then \
      git clone https://github.com/bbaranoff/c54x_exe /opt/GSM/c54x_exe \
      && cd /opt/GSM/c54x_exe \
      && make QOSMO=/opt/GSM/qosmo \
             CFLAGS="-O3 -g -Wall -Werror=format -Werror=format-extra-args -Wno-unused-function
↪  -Wno-unused-variable -Wno-unused-but-set-variable -Wno-sign-compare" \
      && FREECALYPSO_URL="$FREECALYPSO_URL" bash rom/fetch-rom.sh --dest /opt/GSM --dest /opt/GSM/c54x_exe/rom \
      && curl -fsS --retry 2 -o /opt/GSM/calypso_dsp.txt "$FREECALYPSO_URL/dsp-rom-3606-dump.txt" \
      && cp /opt/GSM/calypso_dsp.txt rom/calypso_dsp.txt \
      && osmo-deb snapshot c54x-exe 0.git /opt/GSM/c54x_exe \
             /opt/GSM/calypso_dsp.PROM0.bin /opt/GSM/calypso_dsp.PROM1.bin /opt/GSM/calypso_dsp.PROM2.bin \
             /opt/GSM/calypso_dsp.PROM3.bin /opt/GSM/calypso_dsp.DROM.bin /opt/GSM/calypso_dsp.PDROM.bin \
             /opt/GSM/calypso_dsp.Registers.bin /opt/GSM/calypso_dsp.txt; \
    fi


if ! osmo-deb install grgsm-exe 0.git; then \
      git clone https://github.com/bbaranoff/grgsm_exE /opt/GSM/grgsm_exe \
      && cd /opt/GSM/grgsm_exe \
      && make QOSMO=/opt/GSM/qosmo \
      && osmo-deb snapshot grgsm-exe 0.git /opt/GSM/grgsm_exe; \
    fi
```

2.26.1.8 Etape 6 : grgsm-venv

STAGE grgsm-venv — GNU Radio 3.10 + gr-osmosdr + gr-gsm dans /root/.env

gr-gsm se lie a libosmocore/libosmocoding/libosmogsm (son CMakeLists les cherche par pkg-config) : ce stage DOIT donc partir d osmo-core, pas de base. Il reste parallele a `bb` et a `l1`, qui en derivent aussi.

Dans Docker : `FROM osmo-core`. Sur une machine, on continue dans le meme systeme.

ARG ne traverse pas un FROM : les stages qui appellent osmo-deb ou ${ROOT} doivent les redeclarer, sinon ROOT est vide et OSMO_DEB_REFRESH=1 (build.sh –no-cache) ne descend pas jusqu ici.

```
export ROOT=/opt/GSM


export OSMO_DEB_REFRESH=0
```

2.26.1.8.1 gr-gsm : GNU Radio 3.10 + gr-osmosdr + gr-gsm dans le venv /root/.env

(= moteur de démod du SI réel utilisé par si_bridge.py / grgsm_decode). Les deps GNU Radio (apt build-dep) sont posees par l unique bloc apt en tete de fichier, deb-src compris : plus aucun apt ici.

2.26.1.8.2 GNU Radio + gr-osmosdr + gr-gsm, en UN paquet : le venv /root/.env

On TÉLÉCHARGE et on exécute CE script (le gist, pinné au commit fcdb409). Il cree /root/.env et y installe les trois. Puis :

Patch gr-gsm : le receiver poste le BSIC/FN du SCH (decode_sch) sur le port `measurements` ET sur stdout (“SCHBSIC ”). Le shunt DSP le recoit (si_bridge.py parse le stdout de grgsm_decode -> UDP 4731 -> feed_sb) et encode le VRAI BSIC dans dispatch_sb (remplace SHUNT_CANNED_BSIC 63). Applique APRES le gist (qui clone+build gr-gsm propre), puis recompile/reinstalle dans le venv. Patch maintenu dans patches/ (regenere a chaque changement gr-gsm).

matplotlib est consomme par tools/fft_global.sh et tools/matrix.sh.

Le paquet du cache (grgsm-venv) est le venv COMPLET, patch et matplotlib compris : c est pour cela que les trois etapes tiennent dans un seul RUN. Les arbres /opt/GSM/{gnuradio,gr-osmosdr,gr-gsm} ne sont pas dans le paquet - rien ne les lit au runtime (le venv porte ses .so avec un RPATH sur /root/.env).

```
mkdir -p "$(dirname /tmp/grgsm-receiver-publish-bsic-fn.patch)" && cp -r
↪  "$REPO"/patches/grgsm-receiver-publish-bsic-fn.patch /tmp/grgsm-receiver-publish-bsic-fn.patch
```

```
if ! osmo-deb install grgsm-venv 0.git; then \
      curl -fsSL
       ↪  https://gist.githubusercontent.com/bbaranoff/3683811057933af0954b661821e950d1/raw/fcdb4092483ec383440b67fc002db0c1583
       ↪  | bash \
      && git -C /opt/GSM/gr-gsm apply /tmp/grgsm-receiver-publish-bsic-fn.patch \
      && cd /opt/GSM/gr-gsm/build \
      && make -j"$(nproc)" \
      && make install \
      && . ~/.env/bin/activate && pip install matplotlib \
      && osmo-deb snapshot grgsm-venv 0.git /root/.env /etc/ld.so.conf.d/gnuradio.conf; \
    fi && ldconfig
```

2.26.1.9 Etape 7 : node

STAGE node — Node 22 et le dashboard web osmo-egprs-web

Dans Docker : `FROM base`. Sur une machine, on continue dans le meme systeme.

ARG ne traverse pas un FROM : les stages qui appellent osmo-deb ou ${ROOT} doivent les redeclarer, sinon ROOT est vide et OSMO_DEB_REFRESH=1 (build.sh –no-cache) ne descend pas jusqu ici.

```
export ROOT=/opt/GSM
```

```
export OSMO_DEB_REFRESH=0
```

Node.js + dashboard web osmo-egprs-web (ex-Dockerfile.run, chaine complete)

Place EN FIN DE FICHIER a dessein : aucune etape du build ne depend de Node, et la fin de fichier preserve le cache des couches longues de compilation. L'ORDRE INTERNE DE CETTE CHAINE EST CONTRAINT — voir les ⚠ ci-dessous. /opt/GSM et pas /opt : c'est /opt/GSM que l'ISO recupere en bloc (docker cp "$CID:/opt/GSM"). Le dashboard vivait a cote, dans /opt, et devait donc etre reclone une seconde fois a la construction de l'image - deux sources pour le meme depot, qui n'avancaient pas ensemble.

```
if [ ! -d "/opt/GSM/osmo-egprs-web/" ]; then \
      mkdir -p /opt/GSM && \
      git clone -b main https://github.com/bbaranoff/osmo-egprs-web /opt/GSM/osmo-egprs-web; \
    fi
```

2.26.1.9.1 Runtime Node.js + service web osmo-egprs-web

Le dashboard /opt/GSM/osmo-egprs-web/server.js tourne en mode NATIF (telnet VTY local, pas de docker) et est servi sur :8080. Le DÉMARRAGE est géré par start-native.sh (`systemctl restart osmo-egprs-web`) — ici on ne fait qu'INSTALLER le runtime + les dépendances + le unit (enable au boot).

Node 22 (LTS « Jod ») — même famille majeure que l'ISO, qui installe nodesource setup_22.x (build-iso.sh). Garder les deux alignés : le dashboard est le même server.js des deux côtés.

```
export NODE_VERSION=v22.23.2
```

Le tarball suit l architecture de l image (x64 sur amd64, arm64 sur un build –platform linux/arm64 pour le Raspberry Pi) : un node x64 dans une image arm64 ne se lancerait pas, et le dashboard avec lui.

```
case "$(uname -m)" in x86_64) _na=x64 ;; aarch64) _na=arm64 ;; *) echo "arch node inconnue: $(uname -m)" >&2; exit
 ↪  1 ;; esac && \
    curl -fsSL "https://nodejs.org/dist/${NODE_VERSION}/node-${NODE_VERSION}-linux-${_na}.tar.xz" \
        -o /tmp/node.tar.xz && \
    mkdir -p /opt/node && \
    tar -xJf /tmp/node.tar.xz -C /opt/node --strip-components=1 && \
    rm -f /tmp/node.tar.xz && \
    ln -sf /opt/node/bin/node /usr/local/bin/node && \
    ln -sf /opt/node/bin/npm  /usr/local/bin/npm && \
    ln -sf /opt/node/bin/npx  /usr/local/bin/npx && \
    node --version
```

Unit systemd (fichier versionné dans le repo osmo-nitb-for-calypso, source unique). Le START reste géré par start-native.sh (`systemctl restart osmo-egprs-web`).

```
mkdir -p "$(dirname /etc/systemd/system/osmo-egprs-web.service)" && cp -r "$REPO"/services/osmo-egprs-web.service
 ↪  /etc/systemd/system/osmo-egprs-web.service


ln -sf /etc/systemd/system/osmo-egprs-web.service \
        /etc/systemd/system/multi-user.target.wants/osmo-egprs-web.service
```

2.26.1.9.2 Dashboard web : install-web-service.sh joue AU BOOT

[2026-08-12] Remplace le geste manuel `bash /opt/GSM/osmo-egprs-web/install-web-service.sh` qu'il fallait refaire dans chaque conteneur pour armer le HTTPS.

Ce qui reste au BUILD : node et le unit du service (plus haut), le `npm install` (juste en dessous). Ce qui passe au BOOT : le certificat TLS auto-signe — une cle privee generee au build serait la meme pour tous ceux qui tirent l'image. Le detail du raisonnement est dans services/osmo-egprs-web-install.service.

⚠ ORDRE : ce bloc DOIT rester APRES le clone de /opt/GSM/osmo-egprs-web (haut de cette section) — la COPY ci-dessous ecrit dans ce depot.

SOURCE UNIQUE DU UNIT. install-web-service.sh installe le unit en le copiant depuis /opt/GSM/osmo-egprs-web/osmo-egprs-web.service — la copie du depot osmo-egprs-web, qui avait DIVERGE de celle-ci (elle avait perdu `CAP_IFACE=any`, donc la capture du dashboard). On ecrase donc cette copie par services/osmo-egprs-web.service : les deux emplacements servent desormais le meme fichier, et le script ne peut plus reintroduire la regression.

```
mkdir -p "$(dirname /opt/GSM/osmo-egprs-web/osmo-egprs-web.service)" && cp -r
 ↪  "$REPO"/services/osmo-egprs-web.service /opt/GSM/osmo-egprs-web/osmo-egprs-web.service
```

Fail-fast : si l'amont retire le script, on le sait au build, pas au boot par un HTTPS muet.

```
test -s /opt/GSM/osmo-egprs-web/install-web-service.sh
```

Dependances JS (ws) — UN SEUL `npm install`, apres le clone. Non fatal (`|| true`) : node_modules est versionne dans le depot, un build hors-ligne reste valable. Mais la sortie est conservee, pas avalee.

```
cd /opt/GSM/osmo-egprs-web && npm install --omit=dev --no-audit --no-fund || true
```

2.26.1.9.3 L INSTALLATION EST JOUEE ICI, AU BUILD

[2026-08-31] On se contentait de VERIFIER que le script existe (le `test -s` ci-dessus) et on remettait tout au boot. L unite arrivait donc dans l image telle que le gabarit la portait, avec son ExecStart fige — et dans un conteneur, ou node vit en /usr/local/bin et non /usr/bin, systemd sortait en

```
    status=203/EXEC
```

en boucle, sans que rien ne parle d un chemin.

Le script sait resoudre node et poser l unite avec le bon chemin : on le joue donc au build, pour que l image contienne une unite deja juste.

`|| true` : dans un build il n y a PAS de systemd. Le script le sait et ne s arrete pas dessus (voir son en-tete), mais ses `systemctl` echouent forcement ; les avaler ici evite qu un detail d environnement fasse tomber une image de 11 Go. Ce qui compte - le unit, le certificat, le runtime - est ecrit avant ces appels.

⚠ ET IL N Y A PLUS DE REJEU AU BOOT. osmo-egprs-web-install.service a ete retire des trois chemins de deploiement, pour une raison simple : ce script TELECHARGE node quand il manque. Le laisser au demarrage faisait dependre d Internet un banc concu pour tourner isole - et un premier boot hors ligne repartait sans dashboard. Tout est desormais fige dans l image ; le demarrage ne fait plus que lancer osmo-egprs-web.service.

```
bash /opt/GSM/osmo-egprs-web/install-web-service.sh || true
```

2.26.1.10 Etape 8 : lte

STAGE lte — la 4G du banc : srsRAN_4G (ZeroMQ) et Open5GS dans /opt/LTE

Dans Docker : `FROM base`. Sur une machine, on continue dans le meme systeme.

ARG ne traverse pas un FROM : les stages qui appellent osmo-deb ou ${ROOT} doivent les redeclarer, sinon ROOT est vide et OSMO_DEB_REFRESH=1 (build.sh –no-cache) ne descend pas jusqu ici.

```
export ROOT=/opt/GSM
```

```
export OSMO_DEB_REFRESH=0
```

osmo-lte-install –configs lit $OSMO_REPO : ce stage a besoin du depot, et il ne derive pas de celui qui le clone. Le clone y est donc refait — c est quelques secondes, et l assemblage refait le sien, qui reste LA source pour le `git pull` de Dockerfile.run.

```
git clone https://github.com/bbaranoff/osmo-operator /opt/GSM/osmo-operator
```

2.26.1.10.1 LA 4G DU BANC : srsRAN_4G (ZeroMQ) et Open5GS, dans /opt/LTE

[2026-09-08] DEUX CHEMINS VERS LES MEMES PAQUETS (voir packaging/ snapshot-lte-debs.sh) : le raccourci photographie le natif de la machine de reference en osmo-build-libzmq / osmo-build-srsran / osmo-build-open5gs, et le cache .deb les rend a `osmo-deb install` ci-dessous - rien n est alors compile. Cache vide : on compile depuis les sources, par le MEME script que le natif (tools/osmo-lte-install.sh –build, OSMO_DEB=1 -> osmo-deb pack). libzmq vient d Ubuntu (libzmq3-dev, pose par `--deps` ci-dessous) : le /opt/LTE/libzmq compile a la main n a de raison d etre que sur le natif. Open5GS : prefixe /opt/LTE/open5gs/install (bin, etc/open5gs, var/log) - pas /root.

LES CONFIGS ET LES LANCEURS AUSSI (`--configs --launchers`, en fin de RUN). [2026-09-09] Ils n etaient poses que dans l ISO (88-lte-pmos.sh) et en natif (addition.sh), « pas ici : l image docker n a pas de HOME de session a servir ». Faux pour le conteneur : il tourne en root, et /root/.config/srsran EST le HOME que srsenb/srsue lisent. L image sortait donc avec les binaires 4G mais sans mme.yaml (donc sans SGs, donc sans CSFB), sans enb.conf, et sans la commande osmo-lte : une 4G qu on ne pouvait pas lancer. Et le .deb Open5GS n apporte AUCUN yaml (ninja install sous DESTDIR ne pose pas les exemples : 128 fichiers, tous dans bin/ et lib/) - les configs du depot sont la seule source. Elles viennent du clone GitHub de /opt/GSM/ osmo-operator (ligne « git clone » plus haut) : ce qui est pousse est ce que le conteneur voit. Les abonnes (configs/open5gs/dump, subscribers.json) ne sont pas poses au build : osmo-epc start les restaure dans MongoDB.

⚠ `--deps` D ABORD, TOUJOURS (pas seulement quand on compile). [2026-09-09] Le commentaire ci-dessus affirmait que libzmq3-dev etait « dans la liste apt plus haut » : il n y etait pas, ni lui ni libmbedtls-dev, libboost-program-options-dev, libconfig++-dev, meson, flex, bison, libmongoc. Le build tombait donc net des le cmake de srsRAN :

```
    Could NOT find MbedTLS (missing: MBEDTLS_LIBRARIES MBEDTLS_INCLUDE_DIRS)
```

La liste apt du haut de ce fichier est celle de la pile Osmocom (et elle est RELUE par install_modules/10-deps.sh pour le natif) : la 4G a la sienne, tenue dans tools/osmo-lte-install.sh (lte_deps), une seule fois pour les trois chemins. On l appelle, on ne la recopie pas. Inconditionnel parce que ces paquets sont aussi le RUNTIME : les .deb du cache ne portent que les binaires (osmo-deb pack = un tar), pas leurs dependances - srsenb sorti du cache reclame quand meme libmbedtls, libzmq5, libconfig++ et boost_program_options. `--deps` tire aussi mongodb-org (base d abonnes du HSS/PCRF) et n est fatal sur aucun paquet.

```
mkdir -p "$(dirname /usr/local/sbin/osmo-lte-install)" && cp -r "$REPO"/tools/osmo-lte-install.sh
 ↪  /usr/local/sbin/osmo-lte-install
```

```
chmod 755 /usr/local/sbin/osmo-lte-install && \
    OSMO_REPO=/opt/GSM/osmo-operator osmo-lte-install --deps && \
    { osmo-deb install libzmq 4.3.5+git || true; } && \
    { osmo-deb install srsgui 0.1+git || true; } && \
    SRS_DEB_VER="$(osmo-lte-install --srs-deb-version)" && \
    if ! osmo-deb install srsran "$SRS_DEB_VER"; then \
        OSMO_DEB=1 OSMO_REPO=/opt/GSM/osmo-operator osmo-lte-install --build || { echo "ECHEC build srsRAN"; exit
        ↪  1; }; \
    fi && \
    if ! osmo-deb install open5gs 2.8.0+git; then \
        OSMO_DEB=1 OSMO_REPO=/opt/GSM/osmo-operator osmo-lte-install --build || { echo "ECHEC build Open5GS"; exit
        ↪  1; }; \
    fi && \
    OSMO_REPO=/opt/GSM/osmo-operator osmo-lte-install --configs --launchers && \
    ldconfig && test -x /usr/local/bin/srsenb && test -x /opt/LTE/open5gs/install/bin/open5gs-mmed \
    && test -s /opt/LTE/open5gs/install/etc/open5gs/mme.yaml && grep -q '^  sgsap:'
    ↪  /opt/LTE/open5gs/install/etc/open5gs/mme.yaml \
    && test -s /opt/LTE/open5gs/install/etc/freeDiameter/mme.conf && test -s
    ↪  /opt/LTE/open5gs/install/etc/open5gs/tls/mme.crt \
    && test -s /root/.config/srsran/enb.conf && test -x /usr/local/bin/osmo-lte && test -x /usr/local/bin/osmo-epc
```

2.26.1.10.2 L UI SMARTPHONE : pmbootstrap PATCHE, dans /opt/user_interface/pmos

Le telephone du banc est une VM postmarketOS lancee par pmbootstrap (patche pour le modem serie PCI, le son du banc, la taille d ecran, et le port 5038 qui n est pas un adb : patches/pmbootstrap-osmo-bench-qemu.patch). Le clone et le patch sont faits ici, une fois, et sortent en osmo-build-pmbootstrap : l ISO (50-injection-image.sh) le pose tel quel, tools/osmo-pmos-install.sh le trouve en place et n a plus rien a cloner. Le NOYAU PPP, lui, ne se construit pas ici (pmbootstrap, non root, montages de boucle) : il arrive par osmo-build-pmos-kernel (packaging/build-pmos-kernel-deb.sh, ou GitHub bbaranoff/pmos_ppp_kernel), pose par 88-lte-pmos.sh.

```
mkdir -p "$(dirname /tmp/pmbootstrap-osmo-bench-qemu.patch)" && cp -r
 ↪  "$REPO"/patches/pmbootstrap-osmo-bench-qemu.patch /tmp/pmbootstrap-osmo-bench-qemu.patch


if ! osmo-deb install pmbootstrap 0.git; then \
        mkdir -p /opt/user_interface/pmos && \
        git clone https://gitlab.postmarketos.org/postmarketOS/pmbootstrap.git
         ↪  /opt/user_interface/pmos/pmbootstrap && \
        cd /opt/user_interface/pmos/pmbootstrap && \
        git apply /tmp/pmbootstrap-osmo-bench-qemu.patch && \
        grep -q osmo_bench_args pmb/commands/qemu.py && \
        osmo-deb snapshot pmbootstrap 0.git /opt/user_interface/pmos/pmbootstrap; \
    fi && test -f /opt/user_interface/pmos/pmbootstrap/pmbootstrap.py
```

2.26.1.11 Etape 9 : osmocom-nitb

========== STAGE osmocom-nitb — L ASSEMBLAGE. C est l image publiee. ==========
Rien ne se compile ici : on repose les .deb produits par les stages, puis on ajoute ce qu aucun paquet ne porte (Node, le dashboard, la 4G, le depot, les configs du contexte). Le nom osmocom-nitb reste sur CETTE etape : build.sh (compose_build) et compose.yaml passent –target osmocom-nitb.

Dans Docker : `FROM base`. *Sur une machine, on continue dans le meme systeme.*

```
export ROOT=/opt/GSM
```

2.26.1.11.1 1. Les paquets de tous les stages

bb DERIVE de osmo-core et l1 DERIVE de qemu : leur cache porte deja celui de leur parent, d ou cinq COPY et non sept.

Copie entre etapes Docker (`--from=bb /var/cache/osmo-debs/ /var/cache/osmo-debs/`) : sur une seule machine, `/var/cache/osmo-debs/` est deja en place.

Copie entre etapes Docker (`--from=l1 /var/cache/osmo-debs/ /var/cache/osmo-debs/`) : sur une seule machine, `/var/cache/osmo-debs/` est deja en place.

Copie entre etapes Docker (`--from=grgsm-venv /var/cache/osmo-debs/ /var/cache/osmo-debs/`) : sur une seule machine, `/var/cache/osmo-debs/` est deja en place.

Copie entre etapes Docker (`--from=lte /var/cache/osmo-debs/ /var/cache/osmo-debs/`) : sur une seule machine, `/var/cache/osmo-debs/` est deja en place.

(pas de COPY depuis `node` : ce stage n appelle jamais osmo-deb - Node est un

```
 tarball et le dashboard un clone git. Son cache n est que celui de base.)
```

⚠ L ORDRE DE POSE COMPTE POUR LES QUATRE PAQUETS PATCHES. osmo-trx, osmo-bts, osmo-hlr et osmo-msc existent en DEUX versions portant le MEME nom de paquet : celle de la boucle (osmo-trx 1.7.2) et la patchee (1.7.2+ipc). Dans la chaine lineaire d avant, la patchee etait posee apres, donc elle gagnait. Ici le glob les pose dans l ordre ASCII, ou « + » (0x2B) precede « ~ » (0x7E) : osmo-trx_1.7.2+ipc passe AVANT osmo-trx_1.7.2, et c est la version NON patchee qui finissait installee - RACH sans table per-RA, BTS/HLR/MSC sans le forcage du RAND, et une LU qui ne passe plus. Silencieux : le binaire est la, il est juste le mauvais. On repose donc les quatre explicitement, en dernier.

```
set -eux; \
    dpkg -i --force-overwrite /var/cache/osmo-debs/osmo-build-*.deb; \
    dpkg -i --force-overwrite \
        /var/cache/osmo-debs/osmo-build-osmo-trx_1.7.2+ipc~*.deb \
        /var/cache/osmo-debs/osmo-build-osmo-bts_1.10.0+rand~*.deb \
        /var/cache/osmo-debs/osmo-build-osmo-hlr_1.9.2+rand~*.deb \
        /var/cache/osmo-debs/osmo-build-osmo-msc_1.15.0+rand~*.deb; \
    ldconfig
```

2.26.1.11.2 2. Ce qu aucun .deb ne porte

Node et le dashboard : le runtime est un tarball detarre dans /opt/node, le dashboard un clone git avec son node_modules. Ni l un ni l autre ne passe par osmo-deb, et install-web-service.sh a deja ete joue dans le stage node.

Copie entre etapes Docker (`--from=node /opt/node /opt/node`) : sur une seule machine, `/opt/node` est deja en place.

Copie entre etapes Docker (`--from=node /opt/GSM/osmo-egprs-web /opt/GSM/osmo-egprs-web`) : sur une seule machine, `/opt/GSM/osmo-egprs-web` est deja en place.

Copie entre etapes Docker (`--from=node /etc/systemd/system/osmo-egprs-web.service /etc/systemd/system/osmo-egprs-web.service`) : sur une seule machine, `/etc/systemd/system/osmo-egprs-web.service` est deja en place.

```
ln -sf /opt/node/bin/node /usr/local/bin/node \
    && ln -sf /opt/node/bin/npm  /usr/local/bin/npm \
    && ln -sf /opt/node/bin/npx  /usr/local/bin/npx \
    && ln -sf /etc/systemd/system/osmo-egprs-web.service \
        /etc/systemd/system/multi-user.target.wants/osmo-egprs-web.service \
    && node --version
```

Chemin EXPLICITE. Sans lui, ce clone heritait du WORKDIR /etc/osmocom (pose beaucoup plus haut, jamais remis a ${ROOT}) et atterrissait dans /etc/osmocom/osmo-operator — masque au runtime par le montage de start-docker.sh l.505, et surtout PAS la ou Dockerfile.run va faire son `git pull`. Deux arbres, deux HEAD, un seul utilise. /opt/GSM/osmo-operator est le chemin nominal, teste en premier par build-iso.sh l.964 et update.sh l.329. On reste sur la branche par defaut du depot (main) : pas de checkout explicite, donc pas de ref a maintenir ici, et HEAD est attache — ce dont le `git pull` de Dockerfile.run a besoin.

```
git clone https://github.com/bbaranoff/osmo-operator /opt/GSM/osmo-operator
```

```
mkdir -p "$(dirname /usr/local/sbin/osmo-lte-install)" && cp -r "$REPO"/tools/osmo-lte-install.sh
 ↪  /usr/local/sbin/osmo-lte-install
```

```
chmod 755 /usr/local/sbin/osmo-lte-install
```

2.26.1.11.3 La 4G

/opt/LTE en entier, et pas seulement ce que portent les .deb : le prefixe d installation d Open5GS vit SOUS l arbre de sources (/opt/LTE/open5gs/install), et /opt/LTE n est pas sous /opt/GSM - osmo-deb pack n y ajoute donc aucune source (SRC_ROOT). Sans cette copie, l image aurait les binaires 4G sans un seul arbre pour les recompiler.

Copie entre etapes Docker (`--from=lte /opt/LTE /opt/LTE`) : sur une seule machine, `/opt/LTE` est deja en place.

⚠ `--deps` DOIT ETRE REJOUE ICI. Ce sont AUSSI les paquets de RUNTIME (libzmq5, libmbedtls, libconfig++, boost_program_options, mongodb-org…) et les .deb d osmo-deb ne declarent aucune dependance - leur en-tete le dit. Les poser dans le seul stage `lte` donnait une image ou srsenb et open5gs-mmed existent mais ne demarrent pas, faute de .so. Le montage du cache apt fait que rien n est retelecharge : les paquets sont deja dans le cache BuildKit, poses par le stage lte.

`--configs --launchers` ensuite, dans le meme ordre que le stage lte : les yaml d Open5GS, enb.conf et les deux lanceurs s ecrivent HORS de tout DESTDIR, donc aucun .deb ne les porte. Ils lisent /opt/GSM/osmo-operator, clone juste au-dessus.

```
OSMO_REPO=/opt/GSM/osmo-operator osmo-lte-install --deps && \
    OSMO_REPO=/opt/GSM/osmo-operator osmo-lte-install --configs --launchers && \
    ldconfig && test -x /usr/local/bin/srsenb \
    && test -x /opt/LTE/open5gs/install/bin/open5gs-mmed \
    && test -s /opt/LTE/open5gs/install/etc/open5gs/mme.yaml && grep -q '^  sgsap:'
     ↪  /opt/LTE/open5gs/install/etc/open5gs/mme.yaml \
    && test -s /opt/LTE/open5gs/install/etc/freeDiameter/mme.conf \
    && test -s /opt/LTE/open5gs/install/etc/open5gs/tls/mme.crt \
    && test -s /root/.config/srsran/enb.conf && test -x /usr/local/bin/osmo-lte && test -x /usr/local/bin/osmo-epc
```

Les alternatives gcc : elles ont ete DECLAREES dans le stage bb, qui ne transmet que ses .deb. On les redeclare ici pour que l image finale offre le meme choix, et on reste sur gcc-11 - le defaut d avant le decoupage.

```
update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-9 90 \
       --slave /usr/bin/g++ g++ /usr/bin/g++-9 \
    && update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-11 110 \
       --slave /usr/bin/g++ g++ /usr/bin/g++-11 \
    && update-alternatives --set gcc /usr/bin/gcc-11
```

2.26.1.11.4 4. Installation des fichiers du projet

```
cd /etc/osmocom
```

```
mkdir -p /etc/osmocom/ && cp -r "$REPO"/scripts/. /etc/osmocom/
```

```
mkdir -p /etc/osmocom/ && cp -r "$REPO"/configs/*cfg /etc/osmocom/
```

```
mv /etc/osmocom/run.sh /root/run.sh
```

Copie des binaires vers /usr/bin pour systemd et installation des .service (La `COPY scripts/gapk-start.sh /etc/osmocom/gapk-start.sh` qui etait ici est

```
 supprimee : `COPY scripts/. /etc/osmocom/` ci-dessus depose deja le fichier.
 Le RUN ci-dessous reste : le symlink et /var/lib/gapk sont uniques.)
```

```
chmod +x /etc/osmocom/gapk-start.sh && \
    ln -sf /etc/osmocom/gapk-start.sh /usr/local/bin/gapk-start.sh && \
    mkdir -p /var/lib/gapk
```

```
cp -f /usr/local/bin/osmo* /usr/bin/ || true && \
    cp -f /usr/local/bin/proto-smsc-* /usr/bin/ || true && \
    cp -f /usr/local/bin/sms-* /usr/bin/ || true && \
    cp -f /usr/local/bin/gen-sms-* /usr/bin/ || true && \
    # Si tu as des fichiers .service dans configs/
    cp /etc/osmocom/configs/*.service /lib/systemd/system/ 2>/dev/null || true
```

2.26.1.11.5 Vérification binaires proto-SMSC

Ex-Dockerfile.run. Premier point du build ou les deux chemins testes sont peuples (produits par gsup-smsc-proto, recopies dans /usr/bin juste au-dessus).

```
which proto-smsc-daemon && which proto-smsc-sendmt && \
    echo "proto-smsc binaries OK" || \
    echo "WARNING: proto-smsc binaries not found in PATH"
```

2.26.1.11.6 5. Fix Permissions & Systemd (Status 214/217)

```
sed -i 's/^CPUScheduling/#CPUScheduling/g' /lib/systemd/system/osmo-*.service && \
    sed -i 's/User=osmocom/User=root/g' /lib/systemd/system/osmo-ggsn.service && \
    sed -i 's/User=osmocom/User=root/g' /lib/systemd/system/osmo-sgsn.service && \
    chmod +x /etc/osmocom/*.sh
```

Activation du service et nettoyage Les unites systemd vivent dans services/ ; elles doivent atterrir dans /etc/systemd/system/, seul repertoire ou systemd les cherche.

```
mkdir -p "$(dirname /etc/systemd/system/osmo-bts-trx.service)" && cp -r "$REPO"/services/osmo-bts-trx.service
↪  /etc/systemd/system/osmo-bts-trx.service
```

```
systemctl enable osmo-bts-trx.service && \
    passwd -d root && \
    systemctl mask getty@tty1.service serial-getty@tty1.service
```

Binaires cote hote produits par osmocom-bb, deposes en une seule couche. `install -m755` plutot que `cp` : les droits sont explicites, pas herites.

```
mkdir -p /root/.osmocom/bb/ \
    && install -m755 ${ROOT}/osmocom-bb/src/host/trxcon/src/trxcon           /usr/local/bin/trxcon \
    && install -m755 ${ROOT}/osmocom-bb/src/host/layer23/src/mobile/mobile   /usr/local/bin/mobile \
    && install -m755 ${ROOT}/osmocom-bb/src/host/virt_phy/src/virtphy        /usr/local/bin/virtphy
```

ccch_scan n'est plus installe ici : la branche fixeria/burst_ind, en fin de fichier, ecrase de toute facon /usr/local/bin/ccch_scan par SA version (cp sans `|| true`, donc obligatoire). Le binaire de master ne survivait jamais — c'est la version burst_ind qui fait foi. Confort du shell interactif. Pas de `source` ici : chaque RUN est un shell neuf, ce serait sans effet — le fichier est lu a l'ouverture d'une session.

```
echo "alias faketrx='python3 ${ROOT}/osmocom-bb/src/target/trx_toolkit/fake_trx.py'" >> ~/.bashrc
```

Prompt : root(rouge)@(bleu)<nom-container=\h>(jaune) (vert)# \h = nom du container grâce à `docker run --hostname "$container_name"` (start-docker.sh). [refactor] LE PS1 ETAIT DEFINI DEUX FOIS, avec deux formats DIFFERENTS : ici et dans Dockerfile.run. Bash lit ~/.bashrc de haut en bas -> c'est celui de Dockerfile.run qui gagnait, et le prompt reellement affiche etait `user@host:cwd☎#`, pas celui decrit deux lignes plus haut. On ne garde QUE le gagnant, a l'identique, pour ne rien changer au prompt visible ; l'explication du `\\h` ci-dessus reste vraie pour les deux formats.

2.26.1.11.7 Prompt interactif (root rouge / hostname bleu / cwd vert / #)

```
printf '%s\n' '' \
    '### calypso-prompt ###' \
    "export PS1='\[\033[1;31m\]\u\[\033[0m\]@\[\033[1;34m\]\h\[\033[0m\]:\[\033[1;32m\]\w\[\033[0m\]☎# '" \
    '### end calypso-prompt ###' >> /root/.bashrc


mkdir -p "$(dirname /root/.osmocom/bb/mobile.cfg)" && cp -r "$REPO"/configs/mobile.cfg
 ↪  /root/.osmocom/bb/mobile.cfg


chmod +x /root/run.sh
```

Répertoires pour le proto-SMSC + arborescence Asterisk (ex-Dockerfile.run, ou ce mkdir faisait doublon avec celui-ci ; seul le chmod 755 etait unique). Position : apres l'installation d'asterisk (liste apt) et apres WORKDIR /etc/osmocom.

```
mkdir -p /var/log/osmocom /var/run/smsc \
        /etc/asterisk /var/lib/asterisk /var/log/asterisk /var/run/asterisk \
        /root/.osmocom/bb && \
    chmod 755 /etc/asterisk /var/lib/asterisk /var/log/asterisk \
        /var/run/asterisk /var/log/osmocom
```

2.26.1.11.8 PulseAudio system-mode : config bakée au build

(ex-Dockerfile.run — depend du paquet pulseaudio de la liste apt en tete ;

```
 place ici, tard, pour ne pas invalider le cache des ~200 lignes de compilation.)
```

Cause racine du warning "[audio] PulseAudio injoignable — audio dégradé" : le démon system-mode lancé au runtime tournait SANS auth-anonymous (le patch de start-native.sh arrivait après son démarrage) → 'pactl' renvoyait "Access denied", et un 2e démon ne pouvait pas démarrer ("Daemon already running"). On bake ici la config pour que TOUT démon soit joignable dès son lancement : socket anonyme + null-sinks gsm_audio/gsm_mic + root dans pulse-access.

[2026-08-12] gsm_mic AJOUTÉ ICI. Il n'était créé que par scripts/pulse-gsm-setup.sh, appelé tard dans run.sh (après osmo-start.sh) : un HLR qui rate coupe run.sh (set -e) et le sink n'existe jamais. Or configs/asound.conf fait pointer la CAPTURE gsm_in sur gsm_mic.monitor, et gapk_io abandonne lecture ET capture si la capture échoue → appel muet des deux côtés. Les deux sinks doivent naître avec le démon, pas avec un script.

```
set -eux; \
    usermod -aG pulse-access root 2>/dev/null || true; \
    test -f /etc/pulse/system.pa; \
    sed -i 's|^#\?load-module module-native-protocol-unix.*|load-module module-native-protocol-unix
     ↪  auth-anonymous=1 socket=/var/run/pulse/native|' /etc/pulse/system.pa; \
    grep -q 'auth-anonymous=1' /etc/pulse/system.pa \
      || echo 'load-module module-native-protocol-unix auth-anonymous=1 socket=/var/run/pulse/native' >>
       ↪  /etc/pulse/system.pa; \
    grep -q 'sink_name=gsm_audio' /etc/pulse/system.pa \
      || echo 'load-module module-null-sink sink_name=gsm_audio format=s16le rate=8000 channels=1
       ↪  sink_properties=device.description=GSM_Audio' >> /etc/pulse/system.pa; \
    grep -q 'sink_name=gsm_mic' /etc/pulse/system.pa \
      || echo 'load-module module-null-sink sink_name=gsm_mic format=s16le rate=8000 channels=1
       ↪  sink_properties=device.description=GSM_Mic' >> /etc/pulse/system.pa; \
    grep -q 'sink_name=osmo_tts_off' /etc/pulse/system.pa \
      || echo 'load-module module-null-sink sink_name=osmo_tts_off format=s16le rate=8000 channels=1
       ↪  sink_properties=device.description=TTS_off' >> /etc/pulse/system.pa; \
    sed -i 's|^load-module module-suspend-on-idle|#load-module module-suspend-on-idle|' /etc/pulse/system.pa; \
    mkdir -p /var/run/pulse && chown -R pulse:pulse /var/run/pulse
```

Dernier maillon de la chaine venv/gr-gsm (ex-Dockerfile.run) : le profil de root active le venv.

```
echo 'source ~/.env/bin/activate' >> ~/.bashrc
```

2.26.1.11.9 scripts bridge camping -> /opt/GSM (sinon /opt/GSM/qosmo/run.sh casse)

si_bridge.py (full SI set -> 4730 -> shunt feed_si), si_bridge_loop.sh, record_drain.py (iq_record.fifo -> record.cfile), grgsm_fft_live.py.

```
mkdir -p /opt/GSM/ && cp -r "$REPO"/opt-gsm/. /opt/GSM/
```

2.26.1.11.10 Metadonnees de l'image

Regroupees a la fin : elles decrivent le conteneur qui tournera, pas une etape de construction. SIGRTMIN+3 est le signal d'arret propre de systemd.

Metadonnee Docker, sans equivalent sur une machine : `STOPSIGNAL SIGRTMIN+3`.

Metadonnee Docker, sans equivalent sur une machine : `ENTRYPOINT ["/etc/osmocom/entrypoint.sh"]`.

Metadonnee Docker, sans equivalent sur une machine : `CMD ["/bin/bash"]`.

2.26.1.12 Etape 10 : debs

2.26.1.12.1 L'ETAPE `debs` : le cache .deb seul, rien d'autre

ATTENTION - `osmocom-nitb` N'EST PLUS LA DERNIERE ETAPE de ce fichier. Tout ce qui construit l'image doit donc dire `--target osmocom-nitb` : build.sh (compose_build), compose.yaml (build.target du service nitb), iso_modules/21-docker-build.sh, .github/workflows/docker.yml. Sans ce –target, docker batit `debs` et sort une image VIDE de 2 Go de paquets.

Pourquoi cette etape existe. En CI, l'image de base n'est plus chargee dans le docker du runner : elle part directement vers GHCR (buildx –push), parce que la materialiser localement demandait de tenir DEUX fois 11 Go et tuait le job en « exporting layers » (exit 143). Mais le workflow doit quand meme recuperer les .deb que le build vient de compiler, pour les rendre a actions/cache - et sans image locale, plus de `docker create` ni de `docker cp` (l'equivalent de debs_from_image, build.sh:306).

D'ou cette etape : une seconde passe `--target debs --output type=local` reutilise INTEGRALEMENT le cache de la premiere (meme builder, memes couches, tout en hit) et n'ecrit sur le disque du runner que /var/cache/osmo-debs - quelques Go de .deb au lieu de l'image entiere. Sur l'hote, build.sh continue de passer par debs_from_image : il a l'image sous la main, lui.

Etape Docker sans contenu propre (image `scratch`) : rien a faire sur une machine.

Copie entre etapes Docker (`--from=osmocom-nitb /var/cache/osmo-debs/ /`) : sur une seule machine, `/var/cache/osmo-debs/` est deja en place.

2.26.2 Annexe : c54x_exe-depuis-ubuntu-vierge.md

Contenu de `doc/c54x_exe-depuis-ubuntu-vierge.md`, repris tel quel.

2.26.2.1 c54x_exe depuis une Ubuntu 24.04 vierge

Oct 20, 2026 · Bastien Baranoff

2.26.2.1.1 Ce que montre cette page

En une vingtaine de minutes, le cœur TMS320C54x émulé de `c54x_exe` charge la mask-ROM TI d'un vrai Calypso et l'exécute, sans QEMU, sans ARM, sans réseau GSM.

C'est le hello world du DSP : il prouve que le cœur exécute du vrai code TI. Il ne prouve pas que le DSP décode quoi que ce soit, puisque aucun burst n'est injecté. Le montage complet (ARM sous QEMU, BTS, pont, appel voix) est décrit en fin de page et vit dans osmo-operator.

Tout se fait en root ou avec `sudo` ; les chemins `/opt/GSM/…` sont ceux que les Makefile et les scripts attendent par défaut.

2.26.2.1.2 1. Prérequis (Ubuntu 24.04)

Une ligne `apt` suffit : le compilateur, `git`, `curl` pour les ROM, et `libosmocore` (qui contient `libosmocoding`, utilisée par `calypso_bsp.c`).

```
sudo apt update
sudo apt install -y build-essential git curl pkg-config libosmocore-dev
sudo mkdir -p /opt/GSM
```

Vérifier que `libosmocoding` est bien là :

```
pkg-config --modversion libosmocoding
```

Si `pkg-config` ne la trouve pas, la version de `libosmocore` du dépôt Ubuntu est trop ancienne ou incomplète ; installer alors `libosmocore-dev` depuis le dépôt nightly d'Osmocom (voir osmocom.org).

2.26.2.1.3 2. Les sources de qosmo

`c54x_exe` ne recopie pas le cœur C54x : il compile directement les sources de qosmo (`calypso_c54x.c` et ses voisins) avec ~120 lignes de doublures d'en-têtes QEMU. Il faut donc le dépôt sur disque, mais pas le construire.

```
cd /opt/GSM
git clone https://github.com/bbaranoff/qosmo
```

Aucun `configure`, aucun `make` ici. Si qosmo est ailleurs que `/opt/GSM/qosmo`, passer `QOSMO=/chemin` au `make` de l'étape suivante.

2.26.2.1.4 3. Cloner et compiler c54x_exe

La compilation prend moins d'une minute et produit un seul binaire, `c54x_exe`, à la racine du dépôt.

```
cd /opt/GSM
git clone https://github.com/bbaranoff/c54x_exe
cd c54x_exe
make QOSMO=/opt/GSM/qosmo \
     CFLAGS="-O3 -g -Wall -Werror=format -Werror=format-extra-args \
             -Wno-unused-function -Wno-unused-variable \
             -Wno-unused-but-set-variable -Wno-sign-compare"
```

Ce `CFLAGS` est celui du Dockerfile d'osmo-operator : il garde les erreurs de format bloquantes et rend muets les avertissements de variables inutilisées, nombreux dans les sources QEMU reprises telles quelles. Un `make` nu marche aussi mais sort beaucoup plus de bruit.

`make` reconstruit le binaire à chaque appel (cible `.PHONY`) ; pas besoin de `make clean` après un changement dans qosmo.

Vérifier :

```
./c54x_exe --help | head
```

2.26.2.1.5 4. Les ROM DSP, depuis FreeCalypso

La mask-ROM du C54x n'est pas dans le dépôt : elle vient du dump publié par le projet FreeCalypso (Mychaela Falconia), sur son FTP public. Le script `rom/fetch-rom.sh` la télécharge et la découpe en 7 sections au format attendu par `--rom-dir`.

```
cd /opt/GSM/c54x_exe
export FREECALYPSO_URL=ftp://ftp.freecalypso.org/pub/GSM/Calypso
bash rom/fetch-rom.sh --dest /opt/GSM --dest /opt/GSM/c54x_exe/rom
curl -fsS --retry 2 -o /opt/GSM/calypso_dsp.txt "$FREECALYPSO_URL/dsp-rom-3606-dump.txt"
cp /opt/GSM/calypso_dsp.txt rom/calypso_dsp.txt
```

`3606` est la version de ROM DSP du Calypso des Motorola C1xx ; c'est aussi la valeur que la L1 DSP écrit dans l'API RAM au boot, ce qui permettra de vérifier plus loin que la bonne ROM tourne.

Après le script, `/opt/GSM` doit contenir exactement ces 7 fichiers :

```
ls -l /opt/GSM/calypso_dsp.*.bin
## calypso_dsp.PROM0.bin  calypso_dsp.PROM1.bin  calypso_dsp.PROM2.bin
## calypso_dsp.PROM3.bin  calypso_dsp.DROM.bin   calypso_dsp.PDROM.bin
## calypso_dsp.Registers.bin
```

Si `curl` refuse le FTP (certaines builds le compilent sans), le dump se télécharge à la main depuis le même répertoire sur ftp.freecalypso.org et se passe à `fetch-rom.sh` en local.

2.26.2.1.6 5. Premier run : la ROM tourne

```
cd /opt/GSM/c54x_exe
./c54x_exe --trames 200
```

`--trames 200` fait tourner le DSP sur 200 trames TDMA (une trame = 4,615 ms de temps GSM), avec le budget d'instructions par défaut de 2300 par trame en mode autonome. Sans option, seules les erreurs du cœur passent sur stderr ; le bilan final dit combien de lignes ont été masquées et avec quel niveau `-v` les voir.

Ce qu'on doit voir, dans l'ordre :

1. Les 7 sections de ROM chargées depuis `/opt/GSM` (ou `--rom-dir`).
2. `c54x_reset()` : la ROM repart de `0xff80`, son vecteur de reset.

3. Des lignes `BRANCH-TRACE` autour de `PC=0xb41f` : c'est la boucle d'attente de la ROM TI, du vrai code que le cœur exécute.
4. Un bilan de fin avec `a_sch[3] = 0x771a`.

Pour voir une ligne par trame :

```
./c54x_exe --trames 50 --verbeux
```

Et pour les traces du cœur, `-v` à `-vvvvvv` (voir `--help`).

2.26.2.1.7 6. Lire le résultat

`a_sch[3] = 0x771a` est une valeur constante d'un run à l'autre : la ROM a booté, s'est parquée en IDLE et a laissé cet état dans sa fenêtre API. C'est la preuve que le cœur exécute fidèlement la ROM jusque-là.

Ce que ça ne prouve pas : le décodage. L'API RAM est vierge, aucun burst n'arrive, le DSP tourne à vide. Un décodeur dont la sortie ne dépend pas de l'entrée ne décode pas ; faire varier l'entrée demande le montage complet de la section 8.

Point de repère utile : le README de `qemu-calypso` donne `0xf8d8` pour ce même mot. La différence vient de l'entrée (QEMU fournit une API RAM initialisée par l'ARM, ici elle est vide), pas d'un bug : deux entrées différentes, deux constantes différentes.

2.26.2.1.8 7. Erreurs classiques

Symptôme	Cause	Remède
`make` : `qemu/osdep.h` ou `calypso_c54x.c` introuvable	qosmo n'est pas à `/opt/GSM/qosmo`	`make QOSMO=/chemin/vers/qosmo`
`make` : `osmocom/coding/...` introuvable	`libosmocore-dev` absent ou trop vieux	section 1, dépôt nightly Osmocom
`fetch-rom.sh` échoue ou `curl: (1) Protocol "ftp" not supported`	`curl` compilé sans FTP, ou pare-feu	télécharger à la main, puis `fetch-rom.sh` sur le fichier local
`c54x_exe` : ROM introuvable	les `.bin` ne sont pas dans `/opt/GSM`	`--rom-dir /chemin` ou refaire la section 4
Le bilan ne montre pas `0x771a`	mauvaise ROM, ou sections mal découpées	comparer `rom/calypso_dsp.txt` au dump 3606 ; refaire la section 4
`PC=0xb41f` n'apparaît jamais, le cœur sort en erreur	régression dans qosmo	`cd /opt/GSM/qosmo && git log -3`, remonter au commit du README

2.26.2.1.9 8. La suite : injecter des bursts

Pour que le DSP décode vraiment, il faut l'ARM (le firmware `layer1.highram.elf` d'osmocom-bb sous QEMU `qosmo`), une BTS et le pont qui lui livre des bursts. Deux chemins :

Le banc complet. Il pose tout, dans les mêmes chemins `/opt/GSM/…` que cette page :

```
cd /opt/GSM
git clone https://github.com/bbaranoff/osmo-operator
cd osmo-operator
sudo ./install.sh
sudo ./start-direct.sh --dsp
```

Le `--dsp` remplace la couche 1 gr-gsm par `c54x_exe` ; `--menu` pose la même question interactivement. Le wiki d'osmo-operator décrit les options, les ports et le plan de numérotation.

Exe par exe. Pour voir chaque étape produire son log : le DSP (`c54x_exe --arm -v`), puis QEMU avec `CALYPSO_DSP_EXTERN=1`, puis `osmocon`, puis `mobile`. La séquence, avec les lignes de journal attendues à chaque pas, est dans LAUNCH.md et le README de c54x_exe.

À partir de là, `a_sch` change avec l'entrée, et la question « est-ce que ça décode ? » a une réponse mesurable.

2.26.2.1.10 9. Le firmware Calypso : la toolchain arm-elf (gnuarm)

Le firmware d'osmocom-bb (`layer1.highram.elf`, `trx.highram.elf`…) se compile avec la toolchain historique `arm-elf` (gcc 4.5.2, binutils 2.21.1, newlib 1.19) du gnu-arm-installer, installée dans `/root/gnuarm/install`. Sur Ubuntu 24.04 son `cc1` réclame `libmpfr.so.4`, que le système ne fournit plus (il a `libmpfr.so.6`) ; un lien suffit :

```
ln -sf /usr/lib/x86_64-linux-gnu/libmpfr.so.6 /usr/local/lib/libmpfr.so.4 && ldconfig
/root/gnuarm/install/bin/arm-elf-gcc --version      # gcc 4.5.2
```

L'arbre `osmocom-bb-transceiver` (branche jolly/testing, appli `trx`) se compile aussi avec le `arm-none-eabi-gcc` du système, avec les adaptations de `patches/osmocom-bb-transceiver-qemu-bench.patch` (préfixe, édition de liens par gcc, `-fcommon`, `solve_envs.py` en python 3) :

```
cd /opt/GSM/osmocom-bb-transceiver/src/target/firmware
make BOARDS=compal_e88 APPLICATIONS=trx ENVIRONMENTS=highram CROSS_COMPILE=arm-none-eabi-
cp board/compal_e88/trx.highram.{elf,bin} /opt/GSM/firmware/board/compal_e88/
```

Tous les patchs appliqués aux arbres osmocom-bb (mobile, transceiver, burst-ind) sont dans `patches/` d'osmo-operator et appliqués par le Dockerfile et `install_modules/40-patches.sh`.

2.27 /opt/GSM/osmo-operator/doc/magnetite.md

8187 octets, 119 lignes

2.27.1 FreeCalypso Magnetite dans qosmo (le vrai firmware TI, cible Motorola C139)

[2026-10-21] Le firmware ARM d'origine (TI TCS211, recompile par le projet FreeCalypso Magnetite de Mychaela Falconia) tourne dans l'emulateur Calypso du depot (qosmo) a la place de la layer1 osmocom-bb : Nucleus, la pile GSM L23 de TI, l'interface BMI, les traces RVTMUX. Le telephone emule devient un **C139** (ecran couleur 96x64 sur uWire, meme clavier Compal que le C123), parce que c'est la seule cible Mot C1xx pour laquelle Magnetite a une configuration avec interface (`c139 hybrid-ui-vo`) ; le C11x/C123 n'a qu'un pseudo-modem AT.

2.27.1.1 1. Sources et chaine de compilation

Magnetite ne se compile **pas** avec gcc (ni /root/gnuarm) : tout l'arbre (sources, `.s` en syntaxe TI, blobs `.lib` COFF, Nucleus) exige le compilateur TI TMS470 1.22e, un binaire Windows qu'on fait tourner sous Wine.

```
/root/.env/bin/hg clone https://hg.themyscira.org/fc-magnetite /opt/GSM/fc-magnetite
/root/.env/bin/hg clone https://hg.themyscira.org/freecalypso-tools /opt/GSM/fc-host-tools  # rvinterf, fc-shell, fc
dpkg --add-architecture i386 && apt-get update && apt-get install -y wine64 wine32:i386
# environnement TI (compilateur, cygwin, perl, java) : ftp.freecalypso.org/pub/GSM/TI_src/wine/installed-env.tar.xz
mkdir -p ~/.wine/drive_c && tar xJf installed-env.tar.xz -C ~/.wine/drive_c && wineboot -u
gcc -O2 -o /usr/local/bin/nowhine nowhine.c    # wrapper qui tait les plaintes de wine (meme repertoire FTP)
cd /opt/GSM/fc-magnetite && (cd helpers && make) && ./configure.sh c139 hybrid-ui-vo
cd build-c139-hybrid-ui-vo && make -j4          # ~1 min, 4 cl470 sous Wine en parallele
```

Resultat : `fwimage.bin` (2 Mo, a flasher a 0x10000) et `flash-script`. Les copies sont dans `/opt/GSM/fc-wine-dl` (env Wine) et `/opt/GSM/fc-dl` (bootloader).

2.27.1.2 2. L'image flash

Sur un C1xx le boot ROM est hors circuit : le CPU demarre a 0 dans la flash, ou le bootloader Compal (version FreeCalypso `compal-flash-boot-for-fc.bin`, 8 Ko) saute a 0x10058. L'image de 4 MiB pour qosmo est assemblee ainsi (`/opt/GSM/firmware/magnetite/c139-flash.img`) :

```
img = 0xFF * 4 MiB ; img[0:8192] = compal-flash-boot-for-fc.bin ; img[0x10000:] = fwimage.bin
```

Le FFS (systeme de fichiers du firmware) vit dans cette flash a 0x3C0000 (3 secteurs de 64 Ko).

2.27.1.3 3. Ce qu'il a fallu changer dans qosmo

fichier	changement
`hw/arm/calypso/calypso_uwire.c` (nouveau)	controleur uWire 0xFFFE4000 + ecran C139 (mots de 9 bits sur CS0, fenetre 0x10-0x15, RGB565) ; publie `/dev/shm/calypso_lcd` avec le magic `C139LCD` ; propriete `c139-lcd`, posee par `CALYPSO_LCD=c139`
`calypso_mb.c`	sans `-kernel` et avec `-drive if=pflash` : pas de vecteurs factices, demarrage a 0 ; flash identifiee Intel 0x88C4 (28F320C3B, celle des C139, reconnue par `ffs/board/dev.c`)
`calypso_soc.c`	`flash_boot` : l'ecriture de CNTL_ARM_CLK ne remappe plus l'IRAM a 0 (Magnetite y ecrit 0x40 : l'IRAM pleine de zeros recouvrait la flash)
`hw/ssi/calypso_spi.c`	la vraie trame TWL3025 (bit 0 lecture, registre en 5..1, donnee en 15..6, reponse dans RX_LSB), STATUS = RE_ST et WE_ST, CTRL bit 0 ou 1 declenche, VRPCSTS = ONBSTS (bouton ON relache : TI attend sa liberation en boucle)
`calypso_trx.c`	le TDMA (IRQ trame = tick Nucleus) demarre au premier `TP_Enable` : TI n'ecrit jamais TPU_INT_CTRL
`target/arm/tcg/hflags.c`	plus de fault d'alignement en ARMv4 (le driver flash TI ecrit un demi-mot a l'adresse impaire 0x5555, comme sur ARM7TDMI)

Le CPU doit etre **ti925t** (ARMv4T). Avec arm946 (ARMv5) le `pop {pc}` des trampolines Thumb de TI interworke vers ARM et plante au premier appel.

2.27.1.4 4. Lancer

Seul, avec la couche 1 gr-gsm de qosmo (pas de DSP) :

```
scripts/magnetite-c139.sh start     # QEMU + rvinterf ; RUNDIR /dev/shm/magnetite ; flash de travail persistante
scripts/magnetite-c139.sh gtk          # la fenetre : photo du C139, ecran couleur, touches cliquables
scripts/magnetite-c139.sh traces       # rvinterf.log (traces L1, GPF, Riviera decodees)
scripts/magnetite-c139.sh fsio ls /    # fc-fsio sur le socket de rvinterf (voir points ouverts)
scripts/magnetite-c139.sh stop         # FLASH_RESET=1 au start suivant pour repartir d'un FFS vierge
```

Sur le banc DSP (c54x_exe, la ROM du DSP emulee) : `c54x_exe/run.sh` avec `FIRMWARE_FLASH=/opt/GSM/firmware/magnetite/c1` (CPU ti925t, `CALYPSO_LCD=c139`, rvinterf a l'etape 3 a la place d'osmocon, pas de mobile a l'etape 4 ; air ou pont inchanges). `start-native.sh --ms-magnetite` le pose : comme `--ms`, mais le mobile est Magnetite (`--bts --ms-magnetite` : la BTS en instance 1, le C139 Magnetite en instance 2 par l'air).

La fenetre : `tools/osmo-c123-gtk.py --modele c139 [--instance N]` (auto : le magic de l'ecran partage). Photo `data/c139.jpg` (Raimond Spekking, Wikimedia Commons, CC BY-SA 4.0, credit dans `data/c139.txt`), zones `data/c139-zones.json` (`--calibrate` pour les reprendre).

2.27.1.5 5. Etat au 21/10/2026 (fin de journee)

Corrections d'emulation ajoutees dans l'apres-midi (toutes dans qosmo, non commitees) :

- `target/arm/tcg/translate.c` gen_bx : en ARMv4T un BX vers l'etat ARM ignore le bit 1 (TI int.s fait « bx pc » en Thumb a une adresse 2 mod 4 : cible (pc+4) & ~3). Sans ca, prefetch abort dans INT_DisableIRQ depuis le gestionnaire de veille, et le HISR timer de Nucleus ne tournait plus jamais (sleeps longs, UART RX, SIM tous geles). C'etait LA cause.
- `hw/char/calypso_uart.c` : IT de reveil RX (SCR[4] / SSR[1]) du driver TI, plus de stub romload en boot flash ; `CALYPSO_UART_DEBUG=1` trace IER/SCR/rx.

- `hw/arm/calypso/calypso_sim.c` : ATR livre en mode manuel (CONF1 SCLKEN/SRSTLEV), IT TX apres chaque octet, octet de procedure T=0 pour les commandes de lecture, reponses de la carte differees de 1,5 ms, reponse SELECT conforme GSM 11.11 (carte 3 V, CHV1 desactive) ; nouvel ATR a chaque warm reset. Resultat : le firmware lit SST, AD, IMSI, LOCI, Kc… puis le MMI affiche « SIM Invalid » (a elucider : sans reseau, ou IMEI absent du FFS).
- `hw/ssi/calypso_spi.c` : VRPCSIM renvoie RSIMRSU (regulateur SIM pret).
- `hw/arm/calypso/calypso_mb.c` : flash 16 bits en boot flash, identifiants places pour que drv.c lise 0x0089 / 0x88C4 : le FFS reconnait la flash (« ffs not formatted »). `fc-fsio format /` efface et programme les 3 blocs (trace pflash OK) puis le firmware finit dans os_SystemError : a reprendre (relecture apres programmation ? erreur FFS ?).
- fc-fsio / fc-shell fonctionnent (ETM par rvinterf) : `scripts/magnetite-c139.sh fsio ls /`.

Outils : `QEMU_GDB=1 [QEMU_GDB_WAIT=1]` (stub gdb tcp::1240, gdb-multiarch, adresses dans fwimage.map ; les points d'arret perturbent le firmware, preferer in_asm ou le vidage des taches Nucleus par le moniteur : TCD_Created_Tasks_List 0x80cd60), `QEMU_TRACE='pflash_*'`, `QEMU_DEBUG_FLAGS=int,in_asm`. Un build DISABLE_SLEEP=1 existe (build-c139-hybrid-ui-vo-nosleep, image c139-flash-sleep.img = build normal) mais n'est pas necessaire depuis le correctif gen_bx.

–ms-magnetite : corrige a 11:10, la flash Magnetite n'est passee qu'au mobile (instance 2), le TRX de la BTS garde le firmware trx (run.sh refuse FIRMWARE_FLASH avec OSMO_TRX=1). La fenetre GTK bascule seule en C139 quand l'ecran partage change de magic, se relance avec /usr/bin/python3 si gi manque.

[2026-10-24] « SIM Invalid » elucide (qosmo `calypso_sim.c`) : l'IT TX etait levee aussitot l'octet ecrit, l'en-tete partait en quelques us et l'ACK arrivait 1,5 ms apres le premier octet. `SIM_Command_Base` ecrit le premier octet puis seulement passe `moderx` a 6 : une tache SIM preemptee entre les deux rangeait l'ACK comme SW1, finissait en SIM_ERR_WAIT et renvoyait un en-tete que l'emulateur prenait pour les donnees (`SELECT 0xa0a4`, puis `CLA=00 INS=00` -> 6D00 a perpetuite). Correctifs : IT TX apres un temps caractere (`CALYPSO_SIM_US_OCTET`), et un silence de plus de 500 ms au milieu d'un APDU le resynchronise.

Points ouverts : os_SystemError au format du FFS ; accrochage reseau sur le banc (–bts –ms-magnetite) jamais observe jusqu'au LU ; DSP reel « occupe boot ».

2.28 /opt/GSM/osmo-operator/environment/README.md

3784 octets, 85 lignes

2.28.1 `environment/` — la configuration, par domaine

`calypso.env` mélangeait 312 variables : chemins d'installation, cadences du modèle, sondes de mesure et contournements, sans distinction. Ce répertoire les sépare **par ce qu'elles font**, pour qu'on sache ce qu'on a le droit de toucher.

Fichier	Domaine	Touchez-y ?
`paths.env`	où sont les dépendances sur **votre** machine	**oui**, c'est le premier à régler
`modes.env`	les profils prêts à l'emploi (le mode qui campe, le mode natif…)	**oui**, choisissez-en un
`bsp.env`	BSP / DMA / DARAM / feed I-Q / horloge TDMA	si vous savez ce que vous faites
`dsp.env`	cœur c54x : interruptions, IMR, cadence d'exécution	idem
`fbsb.env`	FB / FBSB / corrélateur / démodulateur	idem
`shunt.env`	canaux shunt DL/UL, injections	idem
`rf.env`	TPU / TSP / RF / AFC	idem
`debug.env`	sondes et traces — **sans effet** sur l'émulation	**oui**, librement
`fixes.env`	le sas `CALYPSO_FIXES` : correctifs en attente de validation	temporairement

`crutches.env`	**les béquilles** — contournements de branches non implémentées	**non**, sauf pour diagnostiquer

2.28.1.1 Trois choses à savoir avant de modifier quoi que ce soit

1. La vérité est le manifeste, pas la ligne de commande.

```
grep "calypso-manifest" /root/qemu.log
```

Certaines variables en reposent d'autres silencieusement : `CALYPSO_NATIVE_HELPED=1` allume aussi `FB_CORR_ENTRY`, `FB_ENERGY` et `FB_IQ_*`. Retirer l'une d'elles de la ligne de commande ne la supprime pas — elle revient à son défaut.

2. `=0` ne coupe pas tout. Quatre idiomes de gate coexistent :

Idiome	Actif quand	Comment couper
`getenv("X") ? 1 : 0`	la variable **existe**, même à `0`	**`unset X`**
`atoi(...) > 0`	valeur > 0	`X=0`
`*e == '1'`	exactement `"1"`	toute autre valeur
défaut ON + `X_OFF`	par défaut	poser `X_OFF`

3. `:=` contre `=`. Dans ces fichiers, `: "${VAR:=valeur}"` laisse la ligne de commande gagner ; `VAR=valeur` la **verrouille**. N'utilisez `=` que pour ce qui ne doit jamais être surchargé.

2.28.1.2 Référence complète

Les 312 variables, avec défaut, effet mesuré, mode, idiome, catégorie et dépendances : `../hw/arm/calypso/doc/VARIABLES_ENVIRO`

2.28.1.3 Comment ça se charge

Tout passe par `load.env`, sourcé par `start-clean.sh` :

```
set -a ; . ./environment/load.env ; set +a
```

L'ordre compte. Tous les fichiers utilisent `:=` (« si pas déjà défini »), donc **le premier qui pose une valeur gagne** :

1. la **ligne de commande** — `VAR=x ./run.sh` gagne toujours ;
2. `modes.env` — le profil choisi ;
3. les fichiers par domaine — les défauts ;
4. `calypso.env` — l'ancien monolithe, conservé le temps de la migration. Il n'utilise que `:=`, donc il ne peut rien écraser : il ne fournit que ce qui n'a pas encore été réparti. À vider progressivement.

2.28.1.4 Les fichiers hérités

`calypso.env`, `calypso_hack.env`, `calypso_native*.env`, `calypso_shunt_*.env`, `calypso_wire.env` datent d'avant ce découpage. Ils sont conservés — on ne détruit pas ce qui marche — mais la configuration nouvelle vit dans les fichiers par domaine.

Fichier	Variables	dont béquilles

2.28.1.5 Ce que contient chaque domaine

Fichier	Variables	dont béquilles
`bsp.env`	52	10
`dsp.env`	52	15
`fbsb.env`	52	25
`shunt.env`	58	34
`armdsp.env`	47	16
`opcodes.env`	50	16

116 béquilles sur 311 variables — plus d'un tiers du projet est du contournement. Chacune est annotée à son site de lecture dans le code (`grep -rn "@BEQUILLE" hw/arm/calypso/`), avec ce qu'elle masque et la condition qui la rendrait inutile.

2.29 /opt/GSM/osmo-operator/navigation/QUICKSTART.md

7054 octets, 158 lignes

2.29.1 Console SS7 - demarrage rapide

Tout vit dans `navigation/`. Aucune dependance : python3 de la distribution suffit. Rien n'est ecrit en dur : point codes, hub, ports et abonnes sont lus dans les conteneurs et dans les configurations du depot.

2.29.1.1 1. En trente secondes

```
cd /home/nirvana/osmo-operator
sudo ./launch/start-docker.sh --wan --operators 2 --hub-ip 192.168.1.49   # le lab
./ss7-console.py                                                 # le schema
```

Le schema s'ouvre. On circule **aux fleches**, on ouvre avec **Entree**, on revient avec **Echap**. `q` quitte.

2.29.1.2 2. Les touches

Touche	Effet
fleches	se deplacer de boite en boite, dans l'espace du schema
Entree	ouvrir le menu de l'element selectionne
Echap	revenir en arriere ; **dans un VTY : fermer CE VTY** et revenir au schema
Tab	element suivant
`v`	ouvrir directement le VTY de l'element
`s`	scripts SS7 (operations MAP)
`a`	audit SS7 (DAUD) d'un point code
`c`	connecter / deconnecter le lien M3UA de la console
`j`	journal du lien M3UA
`r`	relire la topologie
`?`	aide

Dans un VTY : `Entree` envoie, haut/bas rejouent l'historique, PagePrec/PageSuiv font defiler, `Echap` ferme la session. Les autres VTY restent ouverts : une session par port, independantes.

2.29.1.3 3. Ce que montre le schema

```
CONSOLE SS7 ---M3UA--- INTER-STP (hub)
                            |
              +--------------+--------------+
          OP1 - STP                      OP2 - STP
          1.11.2                         1.21.2
             |                              |
    STP  MSC/VLR  HLR  BSC  BTS  MGW  SGSN  GGSN  ...
```

Chaque boite de service porte son port VTY. Les couleurs suivent la famille : SS7, coeur, abonnes, radio, media, data.

2.29.1.4 4. Sans interface : la ligne de commande

```
./ss7-console.py list                inventaire (PC, hub, SCCP, VTY)
./ss7-console.py ops                 catalogue des operations MAP
./ss7-console.py vty 1 4258 "show subscribers all"
./ss7-console.py audit 1.11.1        le "ping" SS7 (DAUD -> DAVA/DUNA)
./ss7-console.py map sri-sm --pc 1.11.6 msisdn=600101 sc_addr=600999
./ss7-console.py serve               repondeur MAP (HLR simule)
```

`vty <n>` accepte le numero d'operateur ou le nom du conteneur.

2.29.1.5 5. Les operations MAP

Cle	Operation	Opcode	SSN vise
`sri-sm`	sendRoutingInfoForSM	45	6 (HLR)
`sri`	sendRoutingInfo	22	6
`ati`	anyTimeInterrogation	71	6
`sai`	sendAuthenticationInfo	56	6
`send-imsi`	sendIMSI	58	6
`psi`	provideSubscriberInfo	70	7 (VLR)
`ul`	updateLocation	2	6
`raw`	invoke brut (opcode + parametre hexa)	-	-

La console est un **vrai** point SS7 : elle s'attache a l'inter-STP en M3UA sur SCTP, enregistre dynamiquement une routing key pour son point code (RKM), passe ASP_ACTIVE, puis emet du TCAP/MAP qui traverse le hub comme le trafic des operateurs. Chaque envoi peut etre exporte en script Python autonome (`navigation/scripts/<operation>-<horodatage>.py`).

2.29.1.6 6. Le repondeur : un aller-retour MAP complet

Osmocom fait dialoguer MSC et HLR en **GSUP**, pas en MAP : un SRI-SM envoye au vrai HLR ne trouve personne. Le repondeur comble ce trou en repondant avec les donnees du VRAI HLR (lues par VTY), a travers le vrai hub.

```
# terminal A
./ss7-console.py serve
# terminal B
./ss7-console.py map sri-sm --pc 1.11.6 msisdn=600101 sc_addr=600999

resultat (opcode 45) :
   IMSI                         001010001000001
   Noeud de service             +600101
```

2.29.1.7 7. Diagnostic et tests

```
./navigation/ss7-diag.py             diagnostic SS7 complet
./navigation/ss7-diag.py --rapide    sans M3UA ni MAP
./navigation/test-console.sh         teste CHAQUE commande, un log par test
```

`ss7-diag.py` verifie, dans l'ordre ou une panne se lit : environnement (SCTP), topologie heritee des configs, STP/MSC/BSC/HLR de chaque operateur, association SCTP vers le hub, attachement de la console comme ASP, audit de tous les point codes connus, puis un aller-retour MAP reel. Le code de sortie est le nombre d'echecs.

`test-console.sh` ecrit un fichier par commande dans `navigation/logs/<horodatage>/`, plus un `rapport.txt`.

2.29.1.8 8. A savoir sur ce lab

- **Le lien M3UA depuis l'hote ne vit que quelques secondes.** L'inter-STP cesse de servir une association venue de l'hote au bout de ~3 s, et l'association SCTP expire vers 35 s. La console la renouvelle toute seule avant chaque echange (~0,3 s) : c'est invisible, mais cela explique un eventuel "pas de reponse" suivi d'un second essai qui marche.
- **Le hub refuse le parametre "Service Indicators"** dans une routing key (registration status 9). Il est donc omis.
- **Il faut proposer son propre routing context** : sans cela le hub rend le meme contexte a deux ASP differents, et en `traffic-mode override` le dernier attache vole le trafic du premier (la destination du premier passe DUNA).
- **Le DATA part sur le flux SCTP 1** : le flux 0 est reserve a la gestion, et un DATA dessus est refuse ("Invalid Stream Identifier").
- **Les point codes sont au format ITU 3-8-3** : `a.b.c` avec a et c de 0 a 7, b de 0 a 255. `1.11.60` n'existe pas - il serait recu comme `1.15.4`.
- Les point codes de la console (`1.11.7`, `1.11.6`…) laissent des routes dynamiques `PROHIB` dans les STP operateur quand elle se ferme : c'est normal, elles repassent `avail` au retour.

2.29.1.9 9. Les modules

Fichier	Role
`topo.py`	inventaire : conteneurs, env, `/etc/osmocom/*.cfg`, ports VTY
`vty.py`	sessions VTY persistantes par `docker exec`
`diagram.py`	le schema et la navigation spatiale aux fleches
`tui.py`	l'interface curses (schema, menus, formulaires, VTY)
`ber.py`	BER/DER minimal
`sccp.py`	SCCP UDT/UDTS, adresses, global titles, TBCD
`tcap.py`	TCAP begin/end, invoke, returnResult, returnError, AARQ/AARE
`mapops.py`	catalogue MAP : encodeurs, decodeurs, erreurs
`m3ua.py`	ASP M3UA sur SCTP : ASPUP, RKM, ASPAC, DATA, battement
`ss7.py`	la couche qui relie tout, et le compte rendu lisible
`responder.py`	repondeur MAP adosse au vrai HLR
`quickcmd.py`	commandes VTY utiles, par demon
`ss7-console.py`	point d'entree (schema + sous-commandes)
`ss7-diag.py`	diagnostic complet
`test-console.sh`	test de chaque commande, avec journaux

2.29.1.10 10. Depannage

Symptome	Cause probable
`aucun inter-STP connu`	lab arrete, ou `--hub-ip` jamais donne a `start-docker.sh`
`le conteneur ... ne tourne pas`	`docker ps` ne le voit plus
`VTY ... injoignable`	le demon n'est pas lance dans ce conteneur
`Aucune reponse` a un MAP	personne n'ecoute ce SSN : lancez `serve`
fleches sans effet	terminal en mode curseur normal - deja gere, sinon `TERM=xterm`
`DUNA` sur un point code	l'ASP de cette destination n'est pas attache au hub

2.30 /opt/GSM/osmo-operator/navigation/STATUS.md

14581 octets, 250 lignes

2.30.1 Etat des lieux et reste a faire

Derniere mise a jour : 2026-09-23 (HEAD `226203d`, runs du banc DSP de 20:22 et 20:32). Les sections 1 a 3 (SS7, numerotation) sont inchangees depuis le 2026-08-26 (lab alors a l'arret, conteneurs `Exited (127)`), sauf le point 4 des bloquants. Nouvelle section 0 pour le banc radio.

2.30.1.1 0. Banc radio — etat au 2026-09-23

montage	etat
grgsm (defaut)	montage de reference (L1 gr-gsm dans QEMU) ; etat du TCH : `pont/README.md`
`--dsp` (c54x_exe)	FB/SB/BCCH decodes par la mask-ROM, camp SI1-4 lai=001-01-1, LU ACCEPT (MSC `timer geran X1 30`, 23/09) ; A5 descendant par le coprocesseur XIO modelise ; appels TCH aboutis (echo 600, mobile a mobile) et SMS MO/MT le 23/09 20:22, voir ci-dessous ; B_BFI sur toutes les trames de parole

`--dsp` a change de nature entre le 22 et le 23/09 : il ne choisit plus de fork. Le fork monte toute la pile en profil hybride avec `--skip qemu,pty,osmocon,l2 --no-attach`, puis `exec c54x_exe/run.sh` lance les cinq etapes du banc DSP : `c54x_exe --arm`, QEMU `/opt/GSM/qosmo`, osmocon, le mobile (`mobile_pont.cfg`, VTY 4347) et `pont/pont_dsp.py`.

Le pont a deux points d'entree : `pont.py` (grgsm) et `pont_dsp.py` (`pont.dsp.main`). `pont_uncipher.py` a ete ajoute puis retire le meme jour. En DSP, le pont ne dechiffre le DL que si `PONT_DSP_DECHIFFRE=1` ; le TCH n'est plus qu'une annonce, le BSP suit la tache du firmware ; l'horloge est asservie en boucle fermee (`PONT_MARGE_DL` 16, `PONT_AVANCE_MIN` 10). Detail : `pont/README.md` § 0 et § 8.

Appels TCH en `--dsp` : bascule suivie par la tache firmware ; crash SP (`RPT *(lk)`, `c54x_exec.c`) et pointeur SACCH `0x3d89` (MVKD/MVDK, garde `[garde-3d89]` dans `c54x_mem.c`) corriges dans le coeur C54x (`qosmo/hw/arm/calypso/l1-dsp`, compile dans c54x_exe) : **confirme au run de 20:22** (deux appels complets sans LOS, aucune ligne `[garde-3d89]` dans `dsp.log`), **mais un LOS sur le premier appel du run de 20:32** (ci-dessous) ; UA perdu / SABM repetes corriges dans `pont/dsp/clock.py` apres le releve de 19:06 : **confirme au run de 20:22** (aucun « SABM frame with information not allowed » au BSC, marge DL reelle moyenne +22.3 a +38.9).

Audio : PulseAudio allege a chaque start ; le passage a speex a ete annule, webrtc reste le defaut, `AUDIO_PENDANT_APPEL` vaut 0.

Incoherences du code, hors doc, non corrigees :

- l'en-tete de `globals.conf` dit encore qu'il ecrase l'environnement ;
- le commentaire d'`audio_start` dans `start-native.sh` et celui de `lib/audio.sh:239` (« SPEEX PAR DEFAUT ») annoncent speex, alors que `lib/audio.sh:131` pose `AUDIO_AEC_METHOD:=webrtc` ;

- `INSNS` vaut 120000 dans `start-native.sh`, son bloc de commentaires dit 200000, le defaut de `c54x_exe/run.sh` est 80000 (son en-tete dit 200000).

A faire :

- ☒ rejouer un appel en `--dsp` : fait le 23/09 20:22 et 20:32, marge DL reelle moyenne >= 22 sur les deux runs, aucune ligne `[garde-3d89]` ;
- ☐ trancher B_BFI : comparer bit a bit les 33 octets livres par la ROM aux trames emises par la BTS (RTP du MGW ou decodage du pont) ;
- ☐ LOS du 20:32:45 : SACCH du TCH perdu en entier pendant 15 s ;
- ☐ SACCH/8 : `fn % 102` des blocs jetes contre ceux acceptes ;
- ☐ aligner `INSNS` entre `start-native.sh` (120000) et `c54x_exe/run.sh` (80000).

2.30.1.1.1 Runs du banc DSP du 2026-09-23, 20:22 et 20:32

Mobile du pont (DSP) = MS 1, IMSI 001010001000001, MSISDN 100101 ; l'autre mobile = 100102. Journaux : `osmo-*.log` (run 20:22 seulement, 20:22:09 a 20:24:56), archives du pont `20260923-202448` et `20260923-203413`.

Ce qui marche (constate) :

- LU du mobile DSP 20:22:41 -> 20:22:45, liberation propre 20:22:47 ;
- appel MO 100101 -> 600 (echo Asterisk) : ASSIGNMENT COMPLETE 20:22:54 sur TCH/F TN=2, ACTIVE 20:22:55, DISCONNECT 20:23:27 (osmo-sip-connector : `NORM_CALL_CLEAR`). Pont : TCH dl=1607 ul=1601 trames, perdus=11. Parole audible dans les deux sens (GAPK FR sur l'hote, decodage canal TCH/F descendant par la mask-ROM) ;
- appel mobile a mobile 100102 -> 100101 via osmo-sip-connector : paging 20:24:20, ASSIGNMENT COMPLETE 20:24:25, ACTIVE 20:24:28, release normal 20:24:32 ;
- SMS MO 100102 -> MT 100101 (20:23:49-55) et MO 100101 -> MT 100102 (20:24:05-09), CP-ACK et RP-ACK recus par le MSC ;
- A5/1 sur les **cinq** etablissements du mobile DSP (`pont.log` « chiffrement descendant confirme » a 20:22:45, 20:22:53, 20:23:53, 20:24:04, 20:24:24 : LU, appel 600, SMS MT, SMS MO, appel MT) ;
- BSP dedie : `stockes=1751 joues=1752 manques=0 perdues=0` (20:22) ; `stockes=6298 joues=6293 manques=0 perdues=0` (20:32) ;
- temps reel : 29513 trames sur la session 20:22 ;
- run 20:32 : second appel ACTIVE 20:33:39 -> DISCONNECT 20:34:06, propre.

Anomalies ouvertes, par ordre d'importance :

1. **B_BFI sur toutes les trames de parole** (le point qui borne les claims). 20:22 : les 40 etats `a_dd` echantillonnes (21 x c214, 18 x c204, 1 x 8084) ont le bit 2 = B_BFI. 20:32, sonde etendue : `vues=2200 bfi=2200` ; sur les 42 releves, `err` (a_dd_0[2], erreurs rapportees par la ROM) vaut 0 sur les 19 x c214 du debut d'appel, 15 a 93 sur les 23 autres. Les trames sont donc reellement degradees et le FR reste intelligible (le firmware ne remonte pas le BFI). Signal (BSP, IQ, egalisation) ou coeur C54x (Viterbi, recodage) : non tranche. Qualite de la parole **non mesuree**.
2. **LOS a 20:32:45 sur le premier appel du run 20:32** : ACTIVE 20:32:30, puis 408 blocs jetes de 20:32:30 a 20:32:45 (54 a 111 bit errors), dont 32 SACCH : `LOSS counter for ACCH` 31 -> 0 jusqu'au « LOS during dedicated mode ». Le SACCH du TCH est perdu en entier ; l'appel suivant (20:33:39) n'en perd presque aucun. Cause non trouvee.
3. **SACCH descendant du SDCCH/8** : 30 blocs jetes par le mobile sur le run 20:22, dont 18 SACCH (« LOSS counter for ACCH », 15 hors TCH) ; LU : 5 SACCH jetes sur ~10 en 5 s, ~la moitie. Suspect : la table 45.002 du BSP pour le SACCH/8 (alternance sur deux multitrames de 51).
4. **UI SAPI 0 sur le TCH** : BSC 20:22:54, lchan(0-0-2-TCH_F-0) {WAIT_RLL_RTP_ESTABLISH} « SAPI=0 UNIT DATA INDICATION: unimplemented Abis RLL message ». UI avec charge (une UI de longueur 0 serait jetee par libosmocore). Non localisee, non bloquante.
5. **Horloge du pont** : avance visee au plafond de 40 a 20:22:37, 20:22:56, 20:32:06, 20:32:40, 20:33:45 (avance BTS mesuree -25 a -29) ; marge DL reelle min +0 a la premiere mesure de chaque run, +6 puis +4 juste apres le LOS de 20:32:45. Moyenne 22 a 39 ailleurs, pas de perte associee.
6. Mineures : `MM_EVENT_NO_CELL_FOUND` x2 a 20:24:08 (resynchronisation apres le SMS sortant) ; MSC 20:24:26 « Duplicate DTAP » sur la reponse au paging.

Pas des anomalies :

- les echecs CRC du **moniteur TCH descendant du pont** (appel 600 : 30 sur les 32 premieres trames, 40 -> 45 en conversation, 103 a la fin) : la BTS n'a rien a mettre sur le TCH tant que le RTP ne coule pas ; decodage du pont, independant du DSP ;
- le `ko=376` de la sonde `[a_dd]` (B_FIRE1) : sans sens defini sur la parole ;
- la fin du run 20:22 a 20:24:52 : arret volontaire (« SIGINT received » de la BTS, rapporte au BSC).

2.30.1.2 1. Ce qui est fait et verifie

2.30.1.2.1 Correctifs du depot

Fichier	Correctif	Verifie
`start-docker.sh`	`asound.conf` : Docker avait cree un REPERTOIRE a la place du fichier, le mount echouait (`not a directory`). Le repertoire parasite est supprime avant la copie, la copie est verifiee, sinon repli propre sur `ALSA_*=default`.	oui, `bash -n` + repertoire nettoye
`start-docker.sh`	`--hub-ip` et `--node-per-op` posent `WAN_MESH=1` : sans cela le plan de point codes retombait sur le plan LOCAL (1.1.2 / 1.2.2, rctx 150 / 250), identique sur chaque machine du WAN.	syntaxe ; **a rejouer sur le lab**
`start-docker.sh`	le rang d'operateur passe a `start-native.sh` (`--op N`) au lieu de `--op 1` fige : `set-node-id.sh` reecrivait les configs de l'operateur 2 avec l'identite de l'operateur 1.	syntaxe ; **a rejouer**
`start-docker.sh`	garde-fou : `--node-per-op` refuse un numero de noeud hors de 1..9 avec un message qui dit quoi faire.	syntaxe ; **a rejouer**
`checks/_mode.sh`	decouverte du hub distant (env des conteneurs puis `osmo-stp.cfg`), point codes libres au format ITU 3-8-3, sonde d'association SCTP.	oui, sur le lab
`checks/ss7_check.sh`	mode WAN : « Container absent » remplace par « hub distant + lien M3UA/SCTP », matrice de connectivite reactivee, route par defaut enfin verifiee.	oui : 19 pass, 0 warn
`checks/wan_ss7_check.sh`	le hub parle SCTP : la sonde `nc -z 2908` (TCP) rendait « injoignable » sur un lien parfait. Remplacee par l'association SCTP reelle. `HUB_IP` lu dans les conteneurs.	oui : 25 ok, 0 echec
`checks/operator_summary.sh`	section « hub distant » quand il ne figure pas dans le dump.	oui

2.30.1.2.2 Console SS7 (`navigation/`)

14 modules, ~4300 lignes de Python, sans dependance. Schema navigable aux fleches, VTY par noeud (Echap ferme la session), ASP M3UA reel sur le hub, operations MAP, repondeur, diagnostic, tests.

Verifie sur le lab en marche :

- schema rendu et navigation spatiale (pilotage sous pty) ;
- VTY OsmoBSC ouvert, commande envoyee, sortie reelle, Echap ferme ;
- ASP M3UA : ASPUP / RKM / ASPAC -> `ASP_ACTIVE` sur `192.168.1.49:2908` ;

- DAUD : `DAVA` sur les 6 point codes du lab, `DUNA` sur un PC inexistant ;
- aller-retour MAP complet : `sendRoutingInfoForSM 600101 -> IMSI  001010001000001`, a travers le hub ;
- `ss7-diag.py` : 25 ok ;
- `test-console.sh` : 10 tests passes avec le lab a l'arret (3 ignores), un journal par commande.

2.30.1.2.3 Ce que le lab nous a appris (garde dans le code, avec le pourquoi)

- le hub REFUSE le parametre « Service Indicators » d'une routing key ;
- il faut PROPOSER son routing context, sinon deux ASP recoivent le meme et, en `traffic-mode override`, le dernier vole le trafic du premier ;
- le DATA doit partir sur le flux SCTP 1 (le 0 est reserve a la gestion) ;
- une association venue de l'HOTE n'est servie que ~3 s, puis expire vers 35 s : la console renouvelle son lien avant chaque echange.

2.30.1.3 2. Ce que l'audit de la numerotation a trouve

Quatre cartographies (point codes, abonnes, indicatifs WAN, identifiants reseau) confrontees aux sources. **12 problemes bloquants**, dont 3 corriges ci-dessus. Les 9 autres, par ordre d'urgence :

2.30.1.3.1 Bloquants, non corriges

1. **Le MSISDN n'encode aucun noeud** — `600000 + op*100 + rang`, et le Ki n'en depend pas non plus. Deux noeuds portent le meme `600101` avec le meme Ki ; seul l'IMSI (via le MCC) les separe. Des qu'un chemin oublie de prefixer l'indicatif, il tombe sur l'homonyme local. `start-docker.sh:1436-1437`.
2. **ISO : meme IMSI, meme MSISDN, meme Ki sur tous les noeuds** — `build-iso.sh` fige `MCC=001 MNC=01 op_id=1`, ce qui contredit `op_mcc() = noeud`. `build-iso.sh:289-290,369-373`.
3. **`--node-per-op` : l'operateur 2 d'un noeud distant est injoignable** — ses MSISDN sont `600201` mais le maillage ne genere que `_<ind>6001XX`. `network/setup-wan-mesh.sh:602,763` vs `start-docker.sh:1678`.
4. ~~**`globals.conf` ecrase l'environnement**~~ — **corrige cote `start-native.sh` le 2026-09-22** : les variables posees non vides par l'appelant sont reposees apres la lecture de `globals.conf`, qui n'a pas ete modifie. Reste : l'en-tete de `globals.conf` dit toujours « ecrasent leur homonyme », et `generate_configs.sh` n'est pas concerne. `start-native.sh:457-490`, `globals.conf:8-11`.
5. **ARFCN / BSIC / LAC / CI ne dependent que de l'operateur** — deux noeuds diffusent `ARFCN 514 / BSIC 7 / LAC 0x0001 / CI 6001`. `generate_configs.sh:287-306`.
6. **Le garde-fou `_gc_warn_perop` ne tourne jamais** sur le chemin reel (mode source) et teste `N_OPERATORS`, que `start-docker.sh` ne lit nulle part.
7. **La plage RTP du MGW (4002-16001) recouvre tous les ports de service** du conteneur, VTY compris. `configs/osmo-mgw.cfg:70-71`.
8. **`node_nops` : repli mort** — `WAN_NOPS[id]=1` est toujours pose, donc un pair est toujours vu a 1 operateur. `network/wan-nodes.sh:109-120`.
9. **La grille d'AS du hub est indexee par RANG, pas par numero de noeud** — une table `1: 3: 5:` produit `as-n1 as-n2 as-n3`. `helpers/create_interop.sh:118-130`.

2.30.1.3.2 Serieux, notables

- trois replis divergents pour MCC/MNC (`start-native.sh`, `run_modules/21`, `scripts/sms-routing-setup.sh`) ;
- seules 2 routes SMS locales generees quel que soit `OP_MS` ;
- `checks/diag-stp-operator.sh` et `wan_ss7_check.sh` figent l'operateur a 1 ;
- deux identites de noeud pour la meme machine en `--node-per-op` ;
- aucun routage par global title (tout est en PC/SSN 254).

2.30.1.4 3. Reste a faire

2.30.1.4.1 Priorite 1 — rejouer ce qui est corrige

- ☐ relancer le lab et verifier les 3 correctifs `start-docker.sh` : `sudo ./launch/start-docker.sh --wan --operators 2 --hub-ip 192.168.1.49` puis `./navigation/ss7-diag.py` (attendu : 2 PC distincts par operateur, rctx distincts, aucun `DUNA` sur les PC du lab) ;
- ☐ `./navigation/test-console.sh` lab en marche (les 3 sections ignorees doivent passer).

2.30.1.4.2 Priorite 2 — numerotation

- ☐ trancher le plan MSISDN : y encoder le noeud, ou assumer que l'indicatif est le seul discriminant et verifier CHAQUE chemin qui le retire ;
- ☐ `build-iso.sh` : deriver MCC/MNC/op_id du `--node`, comme `start-docker.sh` ;
- ☐ `generate_configs.sh` : indexer ARFCN / BSIC / LAC / CI par noeud ;
- ☒ `globals.conf` : l'environnement non vide gagne, cote `start-native.sh` (2026-09-22) ;
- ☐ corriger l'en-tete de `globals.conf`, qui dit encore l'inverse ;
- ☐ `network/wan-nodes.sh` : ne poser `WAN_NOPS` que si le champ existe.

2.30.1.4.3 Priorite 3 — outillage

- ☐ `checks/diag-stp-operator.sh` : ne plus figer l'operateur 1 ;
- ☐ console : bornes du repondeur (plusieurs operateurs, choix du SSN) ;
- ☐ console : rejouer un script exporte (`navigation/scripts/`) depuis le schema.

2.30.1.5 4. Points d'attention

- `navigation/` appartient a `nirvana:nirvana` ; `start-docker.sh` reste `root:root` comme avant.
- Sauvegardes : `start-docker.sh.bak.20260826-221924` (asound) et `start-docker.sh.bak.*-numerotation` (les 3 correctifs), `checks/*.bak`.
- L'audit complet (phase 2 et 3 du workflow) n'a pas ete rejoue : seules les quatre cartographies ont abouti. Leurs conclusions sont resumees ici.

2.31 /opt/GSM/osmo-operator/pont/README.md

30234 octets, 619 lignes

2.31.1 Chiffrement A5, Kc et SACCH — ce qui a été mesuré, et pourquoi c'est écrit ainsi

Ce document est un **compte rendu de mesure**, pas une présentation. Il existe parce que chacune des pannes ci-dessous s'est présentée comme « le pont a des CRC » — un compteur global qui ne distingue rien — et qu'il a fallu à chaque fois remonter d'un symptôme identique à des causes sans rapport entre elles.

Tout ce qui est chiffré ici vient des journaux du banc. Les hypothèses écartées sont conservées : elles étaient plausibles, et les réécarter coûte moins cher que les redécouvrir.

2.31.1.1 0. Deux montages, deux points d'entrée

Depuis le 2026-09-01, le pont est un **paquet** (`trx`, `downlink`, `uplink`, `cipher`, `state`, `gsm`, `record`, `stats`, `config`, `airmesh`). Depuis le 2026-09-23, il a deux points d'entrée sur ce paquet :

point d'entrée	montage	défauts posés avant l'import
`pont.py`	grgsm (L1 gr-gsm dans QEMU, GSMTAP 4730/4731)	`PONT_UL_RETARD_MAX=0` ; refuse `--dsp-port`
`pont_dsp.py` → `pont.dsp.main`	DSP (`c54x_exe --arm`) ; `--dsp-port 6702` (ou `PONT_DSP_PORT`) obligatoire	`PONT_UL_RETARD_MAX=26`, `PONT_KC_RETENTION=1`

Les numéros de ligne cités aux §1 à 3ter (`pont.py:941-942…`) et les noms `ul_facch_from_sideband`, `_tn_epoque`, `tch_desarme` renvoient à l'**ancien** `pont.py` monolithique. Le montage DSP est décrit au § 8.

2.31.1.2 1. Le Kc — trois écrivains, aucun arbitrage

2.31.1.2.1 Le symptôme

`/dev/shm/calypso_kc` à 32 zéros. `A5=non` sur la totalité du canal dédié. `SOUS-VOIE ACTIVE : clair 50/277 chiffre 0/96` — **zéro** bloc chiffré décodé sur 96. Tout ce qui suit le CIPHERING MODE COMMAND rate le CRC, SDCCH puis TCH (qui conserve le chiffrement après l'ASSIGNMENT COMMAND). D'où des CRC qui n'apparaissent qu'au deuxième appel : le premier passe en clair, le second trouve un réseau qui chiffre.

2.31.1.2.2 La cause

Trois producteurs écrivaient le même fichier, sans le moindre arbitrage :

#	écrivain	état
1	`osmocon`	hors dépôt, **efface** le fichier sur `DM_EST_REQ`
2	`l1ctl_sock.c` (QEMU)	chemin mort — le socket l1ctl est orphelin, le mobile parle à osmocon
3	`calypso_dsp_shunt.c`	**n'existait pas**

Le troisième est le point clé. `pont.py` documentait `shunt_publish_kc` depuis le 27/08 et avait mis `PONT_KC_RETENTION=0` *sur la foi de son existence*. Or `grep shunt_publish_kc` ne rendait rien dans tout `qosmo-grgsm`. On avait donc coupé le dernier filet en se fiant à un producteur imaginaire.

Le dernier qui écrit gagne — et c'est l'effaceur.

2.31.1.2.3 Pourquoi osmocon effaçait

osmocon est le multiplexeur série entre le mobile et le firmware. Il **espionnait** `L1CTL_CRYPTO_REQ` au passage et en profitait pour publier la clé. C'était la seule façon de la sortir à l'époque.

Mais un espion voit passer la clé, pas sa fin de vie : il doit la **deviner**. Il la devinait sur `DM_EST_REQ` et `DM_REL_REQ`. L'intention est juste pour le RELEASE, **fausse pour l'ESTABLISH** : le mobile émet aussi un `DM_EST_REQ` pour ouvrir un lien *supplémentaire* sur un canal **déjà chiffré** (le SAPI 3 du SMS, un ré-établissement LAPDm). La clé disparaissait en pleine session.

2.31.1.2.4 Le correctif

Un format commun ne règle pas une course — c'était l'erreur de la première passe. Il faut séparer les **autorités** :

- **La source autoritaire a son propre fichier.** `calypso_dsp_shunt.c` publie `/dev/shm/calypso_kc_l1` depuis `d_a5mode` et `a_kc[4]` du NDB, c'est-à-dire ce que le **firmware** a réellement chargé dans le DSP (`calypso/dsp.c:dsp_load_ciph_param`). Aucun autre écrivain ne connaît ce chemin. La course disparaît au lieu d'être arbitrée.
- **Le patch osmocon est retiré** du `Dockerfile`, désappliqué, et osmocon reconstruit. Il n'écrit plus rien. Le patch reste dans `patches/` à titre documentaire.
- `pont.py` lit `calypso_kc_l1` en priorité et ne retombe sur l'historique que s'il est absent — **jamais un mélange des deux**, sous peine de réintroduire la course.

2.31.1.2.5 Les offsets NDB ne se devinent pas

`d_a5mode` et `a_kc` sont résolus **du DWARF du firmware vivant**, via `tools/ndb-offsets.py`, comme les six autres champs. Valeurs au 2026-08-30 : `d_a5mode=0x1ce`, `a_kc=0x2ce`.

C'est ici que le repli par `#define` est le plus dangereux : un offset faux ne produit pas d'erreur, il produit une **clé fausse**. `osmo_a5` accepte n'importe quels 8 octets et rend un keystream. Le trafic se « déchiffre » alors en bruit, et le CRC accuse la radio.

Même piège sur l'ordre des octets : le firmware range la clé **à l'envers, et par mots** — `a_kc[0] = key[7] | key[6]<<8` … `a_kc[3] = key[1] | key[0]<<8`. Il faut défaire les **deux** inversions.

2.31.1.2.6 Preuve de fonctionnement

```
KC : couche 1 EN CLAIR (d_a5mode=0, seq=1)
KC : A5/1 actif, Kc=29365ddd5623c400 (NDB d_a5mode@0x1ce a_kc@0x2ce, seq=2)
KC : couche 1 EN CLAIR (d_a5mode=0, seq=3)
KC : A5/1 actif, Kc=29365ddd5623c400 (seq=4)   …
```

Côté pont : `clair 21/21`, `chiffre 62/69`.

2.31.1.3 2. La rétention du Kc — trois marqueurs faux, puis le bon critère

Quatre tentatives, dont **trois miennes ou héritées, et fausses**. Elles sont listées parce que chacune paraît raisonnable jusqu'à la mesure qui la tue.

marqueur de relâchement	pourquoi c'est faux
`seq` de `calypso_dcch_cfg`	ne bouge pas du LU à l'appel (`chan_nr` reste 0x41) → clé retenue **indéfiniment**, A5 appliqué à un canal qui démarre en clair → UA brouillée, T200 ×6, appel mort
SABM montante	une SABM ne veut **pas** dire « on repart en clair » : un ré-établissement LAPDm sur canal chiffré en est une, le SAPI 3 du SMS aussi. Mesuré : 11 lâchers en pleine session
`algo=0` d'un producteur autoritaire	`d_a5mode` décrit **notre** L1, pas le **réseau**. Le firmware appelle `dsp_load_ciph_param(0, NULL)` à chaque resynchronisation (`sync.c:419`), donc **pendant** la bascule SDCCH→TCH. Lâcher là, c'est cesser de déchiffrer l'ASSIGNMENT COMMAND elle-même

Le troisième est instructif : il a produit, en une ligne de journal chacun, la cause **et** l'effet.

```
Kc LACHE (la couche 1 declare le clair (seq=7))
TCH arme TN=2 depuis 5 s SANS ASSIGNMENT COMPLETE : le mobile n'a pas bascule
```

Le DSP étant shunté, **c'est le pont qui déchiffre pour le firmware**. Le mobile ne recevait donc jamais l'ordre de basculer.

2.31.1.3.1 La règle retenue

> La fin de vie d'une clé est un **événement réseau**, pas un état local.

Les trois marqueurs conservés le sont tous : `IMMEDIATE ASSIGNMENT` (seul octroi d'un canal **neuf**, qui démarre toujours en clair), `CHANNEL RELEASE` sur SDCCH, `CHANNEL RELEASE` sur FACCH. La rétention (`PONT_KC_RETENTION`, défaut `1`) ne compense plus l'effaceur d'osmocon — il n'existe plus — mais la fenêtre pendant laquelle notre L1 est en clair alors que le réseau chiffre encore.

2.31.1.4 3. `unsupported SAPI` — deux diagnostics faux, un vrai défaut trouvé en chemin

2.31.1.4.1 Le symptôme

Côté mobile, en boucle :

```
DLLAPD lapdm.c Received frame for unsupported SAPI 6!
DRR    gsm48_rr.c:4390 Radio link is released
DMNCC  mnccms.c:714    Call has been released (cause 16)
```

SAPI 6 n'existe pas en LAPDm (0 = signalisation, 3 = SMS).

2.31.1.4.2 Une fausse piste, gardée parce qu'elle était convaincante

Première explication : l'en-tête L1 du SACCH (2 octets) ne serait pas retiré, et LAPDm lirait l'octet de **puissance ordonnée** comme octet d'adresse. Le comptage collait parfaitement — 33 SAPI invalides pour exactement 33 blocs SACCH tentés (`TS1/32 : 30/33`), et les quatre valeurs observées se rangeaient dans les paliers de puissance.

C'était faux. Le tampon réellement présenté commence par `07` : `(0x07 >> 2) & 7` = **SAPI 1** — une valeur jamais observée. Une corrélation de comptage n'est pas une preuve tant que l'arithmétique du mécanisme ne suit pas.

Vérifié au passage, et négatif aussi : la table `mf_sdcch8_0` du firmware est conforme au 05.02 (`MF_F_SACCH` bien posé à `fn%102 ∈ [32,35]`), et `prim_rx_nb.c:122` en tire correctement `link_id = 0x40`. **Aucun patch firmware n'était nécessaire** — c'est précisément ce que la mesure a évité.

2.31.1.4.3 Deuxième explication : le SI du camp — réfutée elle aussi

`lapdm_rx_ph_data_ind` calcule `sapi = (l2h[0] >> 2) & 7`. Les valeurs observées correspondent à des premiers octets dans `0x08–0x1B`, soit la forme d'un **octet de pseudo-longueur** de bloc BCCH — `(L << 2) | 1`, donc `SAPI = L & 7` :

SI	L	1er octet	`L & 7`	SAPI observé
SI1 / SI2	21	`0x55`	5	5 (×2)
SI3	18	`0x49`	2	2 (×2)
SI4	12	`0x31`	4	4 (×1)
—	22	`0x59`	6	6 (×28)

Ce ne sont pas des blocs SACCH : ce sont les **SI du camp**, injectés dans `a_cd` pendant le canal dédié. Le rapport de forces, mesuré :

écrivain de `a_cd`	occurrences
`DCCH-SACCH #` (présentation légitime)	**3**
`DISPATCH ALLC → SI3 a_cd[3..14]`	**29 483**

`DCCH-GARDE` doit empêcher exactement cela. Elle s'armait bien, mais se levait sur un **TTL de fraîcheur de blocs** (`dcch_guard_tick`, ~2 s) rafraîchi uniquement par le **descendant** — donc par notre chance de décodage. Trois blocs sur tout le run : entre deux, le TTL expirait, la garde concluait « canal fini », et le camp reprenait `a_cd`.

```
DCCH-GARDE : ARMEE  -- SI du camp supprime dans a_cd
DCCH-GARDE : levee (peremption) -- SI du camp retabli     ← ×4, canal bien vivant
```

C'est **la même erreur de raisonnement** qu'aux §1 et §2 : déduire un événement de cycle de vie (« le canal est fini ») d'un symptôme local (« je n'ai rien décodé depuis 2 s »). Sur un banc où le décodage est imparfait, les deux sont confondus en permanence, et la garde se retourne contre le canal qu'elle protège.

2.31.1.4.4 Le correctif — et ce qu'il a prouvé, ce qu'il n'a pas prouvé

La garde est rafraîchie par le **montant dédié** (`d_task_u` ∈ {12 DUL, 13 TCHT, 14 TCHA}) : le mobile qui émet sur son canal dédié prouve que ce canal existe, indépendamment de notre décodage.

Mesuré, et le correctif fait ce qu'il annonce — sonde du site de présentation (`CAMP: a_cd<-SI`), et non celle du point d'injection, qui continue d'imprimer en amont de la garde :

	avant	après
`CAMP: a_cd<-SI` pendant le dédié	flot continu	**0**
`levee (peremption)`	×4	**0**

Mais les SAPI invalides ont AUGMENTÉ : 33 → 52. Le SI du camp n'était donc pas leur source. La cause reste **inconnue** — c'est le deuxième diagnostic faux sur ce même symptôme, après celui des paliers de puissance. Le correctif de la garde est conservé parce qu'il corrige un défaut réel et démontré ; il ne corrige simplement pas celui-là.

2.31.1.4.5 Entretenir n'est pas armer

La première version **armait** la garde depuis le montant. `task_u` vaut 12/13/14 bien au-delà du canal dédié (621 / 1954 / 81 sur le run) : la garde s'est armée à 6 % du journal et n'a **jamais** été levée — `CAMP: a_cd<-SI` = 0 sur les 94 % restants, et 27 lignes de sélection de cellule côté mobile. C'est trait pour trait la famine de SI que ce TTL existait pour éviter (commentaire du 2026-08-12), et qu'on réintroduisait en croyant la contourner.

Corrigé : l'**armement** reste au descendant (`set_dcch_active`), seul signal lié à un bloc dédié réel ; le montant ne fait que **repousser la péremption** d'une garde déjà armée. Le TTL reste court (~2 s) — c'est lui qui rend le camp au mobile dès la fin du canal, et l'allonger l'affame. Le montant couvre les trous *pendant* le canal ; le TTL tranche *après*.

2.31.1.5 3bis. Identité n'est pas instance — un vrai défaut, mais pas LA cause

2.31.1.5.1 La mesure

Un LU suivi de **deux appels**, tous trois sur SDCCH/8 SS0 :

```
DCCH #1 : chan_nr=0x41 -> SDCCH/8 SS=0 TN=1 (vu sur DATA_IND)     ← et c'est tout
/dev/shm/calypso_dcch_cfg : seq=1
```

Trois canaux dédiés successifs, un seul appris.

2.31.1.5.2 La cause

```
/* l1ctl_sock.c */
if (kind >= 0 && chan_nr != last_chan_nr) {
    dcch_seq++;
    ... écrit /dev/shm/calypso_dcch_cfg ...
    calypso_dsp_shunt_set_dcch(kind, ss);
}
```

La mise à jour ne se déclenche que si `chan_nr` **change**. Or le BSC réalloue volontiers la même sous-voie : un second appel qui retombe sur SDCCH/8 SS0 présente le même `chan_nr = 0x41`. Rien ne se passe — ni nouveau `dcch_seq`, ni `set_dcch()`. Le shunt reste configuré sur l'**instance précédente** : fenêtre de présentation `a_cd` périmée pour le canal courant.

`chan_nr` **est** l'identité du canal ; il ne porte aucun numéro d'instance. On ne peut donc pas comparer plus finement — il faut **oublier** l'identité quand le canal se termine.

2.31.1.5.3 Le correctif

`calypso_l1ctl_dcch_forget()` remet `last_chan_nr` à `0xFF`. Elle est appelée par la garde `DCCH` au moment où celle-ci conclut à la fin du canal — et la mesure montre que la garde, elle, cycle correctement (**4** paires `ARMEE`/`levee` sur ce run). C'est donc elle qui sait, et c'est d'elle qu'on prévient.

La prochaine établissement redéclenche alors, même à `chan_nr` égal.

2.31.1.5.4 Confirmation par le réseau, et par la mesure d'après

Le BSC voit la même chose par l'autre bout — il réutilise tout, jusqu'à l'objet :

```
lchan(0-0-2-TCH_F-0)[0x5e7e3da21f50]   ×7    (jamais TS3..TS7, sur six TCH/F)
lchan(0-0-1-SDCCH8-0)[0x5e7e3da21e20]  ×4    (jamais SS1..SS7, sur huit)
lchan(0-0-1-SDCCH8-0){ESTABLISHED}: ERROR INDICATION
    cause=SABM frame with information not allowed in this state   ×4
```

Le mobile réémet un SABM sur un lien que le BSC tient pour établi : il n'a jamais vu l'UA. Puis `EQUIPMENT FAILURE: Timeout` et `rll_ready=no`.

La réutilisation n'est pas le défaut, c'est le révélateur. Rien dans le GSM ne l'interdit, et l'allocateur reprend naturellement le premier canal libre : avec un seul mobile, c'est toujours le même. Un réseau réel ferait pareil — donc ce défaut frapperait *tous* les deuxièmes appels, partout. Ce n'est pas un artefact de banc.

Après correctif :

```
DCCH #1 : chan_nr=0x41 -> SDCCH/8 SS=0 TN=1
DCCH #2 : chan_nr=0x41 -> SDCCH/8 SS=0 TN=1     ← MEME chan_nr, et pourtant appris
dcch_cfg : seq=2      identite oubliee : x1      DCCH-GARDE : ARMEE x2, levee x1
```

2.31.1.5.5 Le motif, pour la quatrième fois

Déduire un **événement** (« canal neuf ») d'une **comparaison d'état** (« `chan_nr` a changé ») au lieu de l'événement lui-même. Exactement comme : osmocon devinant la fin de vie du Kc sur `DM_EST_REQ` (§1), les trois marqueurs de relâchement (§2), et la garde concluant « canal fini » sur un TTL de fraîcheur de décodage (§3). Quand ce dépôt se trompe, c'est presque toujours de cette façon-là.

2.31.1.6 3ter. Le DISC de l'appel précédent rejoué sur le canal suivant — LA cause

C'est le défaut qui produisait les échecs d'assignation intermittents. Trouvé par un fan-out de 29 agents avec réfutation adversariale, après que quatre hypothèses posées à la main se soient révélées fausses.

2.31.1.6.1 Le mécanisme

`/dev/shm/calypso_tch_facch_ul` est un slot **unique**, dont le `seq` est de portée-processus côté shunt : il n'a aucune notion d'époque de canal dédié.

1. Un appel se termine **normalement** → `tch_desarme()` met `_tch_tn = None`.
2. `ul_facch_from_sideband` part alors en `continue` sans jamais lire (`pont.py:941-942`) — et son curseur `last`, **local au thread**, gèle.
3. Le mobile émet ensuite son **DISC** LAPDm de libération. Le shunt le publie dans le slot avec un `seq` neuf. Le bloc y reste, avec `seq > last`.
4. À l'ASSIGNMENT COMMAND suivant, le thread se réveille, voit `seq != last`, et traite ce vieux DISC comme un montant frais : il pose `_tch_mobile_ok` **avant toute bascule réelle**, et l'ordonnanceur l'émet sur la FACCH du TCH tout neuf.
5. La LAPDm de la BTS, en `LAPD_STATE_IDLE` sur ce lchan fraîchement activé, répond **DM**.
6. Le mobile, qui vient d'envoyer son vrai SABM, reçoit ce DM en `SABM_SENT` → `ASSIGNMENT FAILURE`. Le BSC n'a jamais son ASSIGNMENT COMPLETE → `T10` → `EQUIPMENT FAILURE: Timeout`.

2.31.1.6.2 La mesure qui l'établit

Capture GSMTAP, premier bloc FACCH montant réellement **émis par le pont** après chaque ASSIGNMENT COMMAND — corrélation **6/6**, aux octets :

```
01 3f (SABM) -> 01 73 (UA)    => appel OK
01 53 (DISC) -> 01 1f (DM)    => appel ECHOUE
```

Et l'arithmétique de propagation exclut « le DM répondait au SABM » : DISC fn 26340 → DM fn 26367 (Δ=27) ; SABM fn 26353 → UA fn 26380 (Δ=27). **Même delta** — le DM répond au bloc émis 13 trames *avant* le SABM.

2.31.1.6.3 Pourquoi c'était intermittent — et pas « le 2e appel »

Rien de positionnel : ce qui compte est le **mode de sortie de l'appel précédent**.

sortie précédente	effet	appel suivant
normale → `tch_desarme` (`_tch_tn = None`)	le thread cesse de lire, le DISC empoisonne le slot	**échoue**
en échec → `tch_suspend` (**conserve** `_tch_tn`)	le thread continue de vider la bande latérale	**réussit**
premier appel	slot à `seq=0`, la garde `if seq and ...` saute	réussit

D'où des séries `ÉCHEC/OK/ÉCHEC` ou `OK/ÉCHEC/OK` selon l'amorce. Toute la formulation « le 2e appel échoue », qui a orienté des heures d'enquête, était un artefact d'échantillon.

2.31.1.6.4 Le correctif

À chaque **nouvelle époque de canal**, resynchroniser `last` sur le `seq` courant **sans consommer** le bloc : ce qui date d'avant l'armement n'appartient pas à ce canal. C'est exactement l'idiome que ce fichier applique déjà à l'anneau voix montant (`session precedente : on ignore`) et qui n'avait pas été porté sur ce sideband-ci.

L'époque est suivie par `_tn_epoque`, remis à `None` au désarmement — car le BSC réalloue toujours TS2, donc comparer les TN ne suffit pas : **identité n'est pas instance**, encore (cf. §3bis).

2.31.1.6.5 Validation

`scripts/banc-repro.sh --calls 4`, pile redémarrée :

	avant	après
appels	ÉCHEC / OK / ÉCHEC	**OK / OK / OK / OK**
`Assignment failed` (BSC)	20	**0**
`unsupported SAPI`	14	4

2.31.1.7 4. Le remplissage de C0 compté comme échec

`osmo-bts` remplit **tous** les timeslots de C0 avec le dummy burst (`scheduler_trx.c`) pour tenir la puissance de la balise. Un canal dédié inactif reçoit donc du remplissage en continu.

Le filtre `_is_dummy()` existait pour le xcch — avec sa mesure : *515 blocs de remplissage comptés en `crc_fail`, 76 % d'échec annoncés sur une radio saine.* **Le chemin TCH ne l'avait jamais eu** : `dl_dispatch` fait `return` sur la branche TCH avant d'atteindre le filtre.

Le traçage `PONT_TCH_TRACE=1` a montré la même cause sous **deux signatures**, selon que le Kc était disponible ou non :

phase	entrée du Viterbi	`ne`
pas de Kc	dummy brut, motif **figé**	`0`, constant
Kc arrivé	dummy **chiffré**, clé variant par trame	~50

`ne=0` avec `rc=-1` ne s'obtient pas sur du bruit : c'est la signature d'une entrée constante. Le filtre est posé **avant** l'`a5_apply`, pour la même raison que côté xcch — le dummy n'est jamais chiffré, et le passer au keystream le rendrait méconnaissable.

2.31.1.8 5. `CALYPSO_CANNED=1` ne cannait rien

`shunt_parse_canned()` attend une **liste de jetons** : `FBDET,TOA,PM,SNR,ANGLE,CRC | FULL | ALL | NONE`.

`start-native.sh` posait `CALYPSO_CANNED=1` depuis toujours. `1` n'est aucun de ces jetons : il tombait dans le `else`, sortait un `token inconnu '1' ignore` noyé dans le démarrage, et le masque restait **vide** :

```
CALYPSO_CANNED (dette fabriquée EXPLICITE) : FBDET=0 TOA=0 PM=0 SNR=0 ANGLE=0 CRC=0
```

Le banc tournait avec **rien de canné** pendant que sa configuration affichait l'inverse.

Corrigé des deux côtés — et les deux comptent : `start-native.sh` pose désormais `FULL`, **et** le parseur accepte `1/ON/YES/TRUE` et `0/OFF/NO/FALSE`. Une variable booléenne qui veut dire « rien » au lieu de « tout » est un piège ; changer la valeur sans changer le parseur l'aurait laissé armé.

2.31.1.9 6. Hypothèses écartées

Conservées pour ne pas être réexplorées.

hypothèse	verdict
Plan SDCCH/8 mal implémenté	**non** — `PLAN_SD8` est conforme au 05.02 : blocs à `4·ss`, SACCH/C8 à 32/36/40/44 avec report `+51` pour SS4..7, UL à `+15`
Trame idle (`m26=25`) empoisonnant la fenêtre TCH	**non** — `tch_dl_burst` écarte `m26 in (12, 25)`, et `gap=0` sur tout le run
Alias SACCH 32/83 en `fn%51`	**non** — filtré sur la 102-multitrame : `(k[1] % 102) != plan.sacch_dl102[_act]`
TCH resté armé après l'appel	**non** — compteurs figés sur cinq intervalles après `DESARME`
`_TBL_SOFT` écrasant deux encodages	**non pour ce symptôme** — le DL TRXD arrive en bits durs `0x00/0x01`. La fragilité reste réelle (une table, deux encodages : `0x01` et `0xFF` donnent tous deux `-127`) mais n'est pas la cause

---

2.31.1.10 7. Reste ouvert

- **`unsupported SAPI` côté mobile** — tombés de 64 à 4, mais pas à zéro. Source résiduelle inconnue. Ne bloque plus les appels (§3ter : 0 échec d'assignation sur 4 appels). Ni l'en-tête L1 du SACCH (§3, réfuté), ni le SI du camp (§3, réfuté par la mesure). Source toujours inconnue : chercher qui d'autre atteint `a_cd`, ou si le firmware relit un bloc périmé faute de rafraîchissement (`B_BLUD`).
- **CRC résiduels** — `chiffre 62/69`, soit 7 échecs sur le SDCCH chiffré, plus 3 sur le SACCH. Non isolés.
- **TCH** — l'histogramme rend `octets[00:928]` : les 928 octets de la fenêtre sont à **zéro**. Ce ne sont ni des dummy bursts (motif non nul, désormais filtrés) ni du bruit : le timeslot ne porte rien. Vraisemblablement en aval du mobile qui ne bascule pas, donc à revoir **après** le correctif SACCH.

2.31.1.11 8. Montage DSP (`pont_dsp.py`, sous-paquet `pont/dsp/`)

Mêmes briques qu'en grgsm, mais l'état dédié suit le DSP au lieu du décodage du pont. Ce qui change :

- **A5 descendant : fait par le DSP.** La mask-ROM programme le coprocesseur XIO du Calypso (ports `0x2800..0x2818`, modélisé dans `qosmo/hw/arm/calypso/l1-dsp/calypso_a5.c`, compilé dans c54x_exe) avec `a_kc` et `a_a5fn`. Le pont n'applique donc plus le flux au DL : le faire en plus renverrait le burst chiffré au décodeur. `PONT_DSP_DECHIFFRE=1` rétablit l'ancien comportement, et va avec `CALYPSO_A5=0` sur c54x_exe. Le **montant** reste chiffré par le pont (au fn de l'air).
- **`TchDsp` — l'ASSIGNMENT COMMAND est une annonce.** `/dev/shm/calypso_tch_cfg` est écrit comme avant, mais c54x_exe ne bascule le BSP qu'à la première tâche TCHT/TCHA/TCHD posée par le firmware. Basculer à l'heure de la BTS remplaçait les trames pas encore jouées — l'ASSIGNMENT COMMAND comprise. `abandon()` écrit `seq+1, tn=0` : retour au SDCCH **sans lâcher le Kc**.
- **`UplinkDsp` — bursts SDCCH/SACCH montants de la ROM** (2026-10-03). Les 4 bursts d'un bloc ne sont plus codés par `gsm.xcch_encode` mais pris dans `/dev/shm/calypso_xcch_ul_rom`, que c54x_exe remplit avec ce que la ROM émet (`data[0x3f8a]`, un burst par trame) et le flux A5 montant qu'elle y XORe en chiffré. Anneau : en-tête `w`(u32) `n`(u32), cases de 1024 octets `seq fn l2[23] . bits[4×116] flux[4×116]`. Le pont redécode, défait le XOR de la ROM si besoin (elle chiffre à l'heure du DSP), compare au L2 de `calypso_sdcch_ul`, puis émet et rechiffre au fn de l'air comme avant. Sans bursts ROM après `PONT_UL_ROM_ATTENTE` (0,12 s) : codage hôte. STATS : `xCCH ul rom= hote= rom_ko=`. Le hook est `Uplink._bursts_xcch` (hôte par défaut, montage grgsm inchangé).
- **`UplinkDsp`** — un bloc de `calypso_sdcch_ul` n'est **jamais** une FACCH (le TCH passe par `calypso_tch_facch_ul`) ; une ASSIGNMENT FAILURE, ou un bloc SDCCH alors que le mobile était sur le TCH, retire l'annonce. Le Kc est gardé après la libération jusqu'à la prochaine IMMEDIATE ASSIGNMENT : la libération se lit à l'heure du DSP, et la BTS chiffre encore.
- **`DedicatedDsp`** — genres 2/3 = TCH/F et TCH/H dans `calypso_dcch_cfg` ; `plan()` rend toujours le dernier SDCCH tant que la connexion vit, seul `0xFF` la termine.
- **`TrxDsp`** — le TS du TCH compte comme dédié **dès l'annonce** (osmo-bts chiffre le TCH dès l'activation), toujours sous la garde de `cipher.dl_active`.
- **`ClockDsp` — horloge en boucle fermée.** avance = `PONT_MARGE_DL` (16) − creux de l'avance BTS sur `PONT_MARGE_FENETRE` (3 s), bornée à [`PONT_AVANCE_MIN` 10, `PONT_AVANCE_MAX` 40]. Depuis le correctif qui suit le relevé de 19:06 (2026-09-23), l'avance BTS se mesure sur le point visé (DSP + avance) et non sur l'horloge réelle, qui traîne dès que le DSP attend la BTS : l'avance tombait au plancher, le SDCCH DL se trouait, l'UA se perdait et le mobile répétait son SABM. Une ligne « marge DL réelle … min … moy » toutes les 10 s dans `pont.log` : c'est elle qui doit rester > 0. `PONT_MARGE_DL=0` revient à l'avance fixe de `Clock`. Correctif **confirmé au run du 2026-09-23 20:22** : aucun « SABM frame with information not allowed » au BSC, marge DL réelle moyenne +22.3 à +38.9.

2.31.1.11.1 État au 2026-09-23 — runs du banc DSP de 20:22 et 20:32

Mobile DSP = MSISDN 100101, l'autre = 100102. Constaté :

- appel 100101 → 600 (écho Asterisk), ACTIVE 20:22:55 → DISCONNECT 20:23:27, parole audible dans les deux sens ; TCH dl=1607 ul=1601 trames, perdus=11 ;
- appel mobile à mobile 100102 → 100101 via osmo-sip-connector, ACTIVE 20:24:28, release normal 20:24:32 ;
- SMS MO/MT dans les deux sens (20:23:49-55, 20:24:05-09), acquittés ;

- A5/1 sur les **cinq** établissements (« chiffrement descendant confirmé » à 20:22:45, 20:22:53, 20:23:53, 20:24:04, 20:24:24), Kc retenu sans casse jusqu'à l'IMMEDIATE ASSIGNMENT suivante.

Ouvert, par ordre d'importance :

1. **B__BFI** *(mise à jour 2026-10-03 : `err` = 0 partout désormais, BFI=1 dès le 1er bloc incomplet ; voir c54x_exe docs/README.fr.md)* sur toutes les trames de parole : 40/40 états `a_dd` à 20:22, `vues=2200 bfi=2200` à 20:32 ; `err` = 0 sur les 19 c214 du début d'appel, 15 à 93 ensuite. Trames réellement dégradées, FR intelligible malgré tout ; signal ou cœur C54x, non tranché.
2. **LOS à 20:32:45** : premier appel du run 20:32 (ACTIVE 20:32:30), 408 blocs jetés jusqu'au LOS, dont 32 SACCH : SACCH du TCH perdu en entier, `LOSS counter for ACCH` 31 → 0. L'appel suivant (20:33:39 → 20:34:06) est propre.
3. SACCH descendant du SDCCH/8 : 30 blocs jetés par le mobile à 20:22, dont 18 SACCH (« LOSS counter for ACCH »).
4. UI SAPI 0 sur le TCH (BSC 20:22:54, « unimplemented Abis RLL message »), non bloquante.
5. Horloge : avance visée au plafond de 40 cinq fois (avance BTS −25 à −29), marge DL réelle min +0 à la première mesure de chaque run.

Pas une anomalie : les échecs CRC du moniteur TCH descendant du pont (30 sur les 32 premières trames de l'appel 600, 103 en fin d'appel) ; la BTS n'a rien à mettre sur le TCH tant que le RTP ne coule pas. Décodage du pont, indépendant du DSP. Détail : `navigation/STATUS.md` § 0.

2.31.1.12 Rejouer une experience

`scripts/banc-repro.sh` fige toute la sequence : demarrage detache (`CALYPSO_NO_ATTACH=1`), attente du camp **interrogee** au VTY (`show ms -> C3`) et non supposee par un `sleep`, N appels identiques, verdict par appel calcule sur la seule tranche de `pont.log` de cet appel, collecte horodatee dans `/root/repro-<stamp>/`.

```
scripts/banc-repro.sh --calls 4                # avec redemarrage
scripts/banc-repro.sh --no-restart --calls 3   # sur la pile en cours
PONT_KC_RETENTION=0 scripts/banc-repro.sh      # A/B d'un reglage
```

C'est ce qui a permis de decouvrir que l'echec n'etait pas positionnel : trois appels suffisent, et deux runs successifs donnent `ECHEC/OK/ECHEC` puis `OK/ECHEC/OK`. Sans sequence figee, on compare des runs qui ne se comparent pas.

2.31.1.13 Outils de diagnostic

variable	effet
`PONT_TCH_TRACE=1`	*ancien pont, absent du paquet actuel* — une ligne par bloc TCH/F en échec : `fn/m26/idx/acc` (alignement), `A5` (déchiffrement et clé), `rc/ne/nb` (décodeur), et l'**histogramme des octets** de la fenêtre
`PONT_KC_RETENTION=0`	désactive la rétention, pour comparer (défaut 1)
`PONT_A5=1..3`	*ancien pont, absent du paquet actuel* — forçait l'algorithme au lieu de suivre celui du Kc
`PONT_DSP_DECHIFFRE=1`	DSP : le pont déchiffre le DL (avec `CALYPSO_A5=0` sur c54x_exe)
`PONT_UL_ROM=0`	DSP : bursts SDCCH/SACCH montants codés par le pont au lieu de ceux de la ROM
`PONT_UL_ROM_ATTENTE`	DSP : attente maximale des bursts ROM d'un bloc avant repli hôte (0.12 s)
`PONT_UL_RETARD_MAX`	retard maximal (trames) d'un burst montant encore envoyé ; 0 = jeter (grgsm), 26 (DSP)
`PONT_WINDOW_ESSAIS`	re-attentes d'un burst montant réveillé trop tôt (12)
`PONT_HORLOGE=0`	horloge murale au lieu de l'horloge asservie au DSP
`PONT_HORLOGE_AVANCE` / `_PERIODE` / `_KP` / `_PHASE_N`	asservissement : avance fixe (12), période (0.05 s), gain (1.0), phase (100)

variable	effet
`PONT_MARGE_DL`, `PONT_AVANCE_MIN/MAX`, `PONT_MARGE_FENETRE`	`ClockDsp` (§ 8) : 16, 10/40, 3 s
`PONT_FACCH_SKIP=1`	saute aussi la FACCH en attente au passage sur le TCH (ancien comportement)
`PONT_AIRREC=0`	coupe l'enregistrement I/Q (`--no-record`), donc la FFT du panneau
`CALYPSO_CANNED=NONE`	rien de canné, tout réel

Lecture des compteurs : `ne` est le nombre d'erreurs binaires, `nb` le nombre **total** de bits (378 pour TCH/FS — constant, ce n'est pas un compteur d'erreurs). `ne=0` avec `rc=-1` signale une entrée constante, pas une radio saine.

2.32 /opt/GSM/osmo-operator/services/README.md

5542 octets, 102 lignes

2.32.1 `services/` — unités systemd

Les unités du projet, rassemblées ici plutôt que dispersées entre la racine et `scripts/`. Elles sont installées par le `Dockerfile` (image) et par `build-iso.sh` (support bootable) ; l'installation native ne les active pas.

Unité	Rôle
`osmo-bts-trx.service`	station de base, transceiver TRX
`osmo-egprs-web.service`	console web d'exploitation
`osmo-banc.service`	le banc GSM autonome (`start-native.sh`) ; pile dans tmux `calypso`
`osmo-multi.service`	le banc multi-opérateur (`start-multi.sh`) ; `Requires=osmo-banc`
`osmo-dsp-rom-max.service`	à la demande : le workflow « la ROM fait le maximum » (`scripts/dsp-rom-max.sh`) : profil ROM max dans `/etc/default/osmo-banc` (persistant), banc DSP, échelle `tests/banc-max.sh --dsp`, bilan ROM/hôte
`osmo-egprs-dsp.service`	le workflow EGPRS (`scripts/egprs-dsp.sh`) : BTS 2 virtuelle EGPRS + `osmo-pcu` + modem GPRS osmocom-bb, par-dessus le banc DSP ; `Requires=osmo-banc`

Ni l'une ni l'autre n'est activée au boot (depuis le 2026-09-05 : une machine fraîchement installée démarre sur son bureau, pas sur une pile radio qu'on n'a pas encore configurée). Elles sont *posées* par l'ISO et par `addition.sh` ; c'est l'opérateur qui démarre son banc — par l'icône du bureau, par `launch.sh`, ou :

```
systemctl start osmo-banc         # une session
systemctl enable --now osmo-banc  # et à chaque démarrage
```

Pour graver une image qui les active quand même : `./build-iso.sh --banc --multi` (ou `OSMO_ISO_BANC=1 OSMO_ISO_MULTI=1`), et `OSMO_MULTI_ENABLE=1 ./addition.sh` sur une machine déjà installée.

Reprendre la main sur le banc lancé en service :

```
tmux attach -t calypso            # Ctrl-b d pour détacher, le banc continue
journalctl -u osmo-banc -f        # sortie de start-native.sh / run.sh
systemctl restart osmo-banc       # un banc neuf ; launch.sh (icône) fait pareil puis s'attache
systemctl start osmo-multi        # ajoute les conteneurs + inter-STP (docker requis)
```

Le DSP est le défaut (start-native.sh, 2026-10-01). Options durables : `OSMO_BANC_ARGS="--grgsm"` dans `/etc/default/osmo-banc` (`OSMO_MULTI_ARGS` dans `/etc/default/osmo-multi`). `launch.sh --dsp` les pose pour la session via `systemctl set-environment`. Avec `--dsp`, osmo-banc lance aussi `c54x_exe/run.sh` après le plan du fork (hand-off par `exec`) ; `systemctl stop` / `./start-native.sh --stop` arrêtent le banc DSP même sans `--dsp`. Le mobile a alors sa VTY sur le port **4347**, et non 4247.

Les unités sous `contrib/systemd/` appartiennent à QEMU en amont et ne relèvent pas de ce dossier.

Installer à la main :

```
sudo cp services/osmo-bts-trx.service /etc/systemd/system/
sudo systemctl daemon-reload && sudo systemctl enable --now osmo-bts-trx
```

2.32.1.1 `osmo-egprs-dsp` — le workflow EGPRS

`scripts/egprs-dsp.sh` monte, étape par étape, la donnée paquet sur le banc : preflight (ROM TI 7/7, `c54x_exe`, `modem`), configs GPRS dans `/etc/osmocom` (`configs/gprs/`, la BTS 2 re-insérée dans `osmo-bsc.cfg` après chaque `--regen`), `osmo-banc` (chaîne DSP), l'abonné du modem au HLR, la cellule EGPRS virtuelle (`osmo-bts-virtual` + `osmo-pcu` sur `/tmp/pcu_bts3`), puis le modem (`virtphy` + `modem`) : GPRS attach, contexte PDP, ping vers `apn0` depuis le netns `msgprs`. Les démons sont des unités transitoires (`systemd-run`) : `osmo-bts-virtual-gprs`, `osmo-pcu-gprs`, `osmo-virtphy-gprs`, `osmo-modem-gprs`.

La porte DSP. Le firmware `layer1` Calypso ne traite pas `L1CTL_GPRS_*` (aucun PDCH sur l'ARM émulé, quoi que sache la ROM) : le script le détecte et met le modem sur `virtphy`, dans la même PLMN/LAC/RA que le mobile DSP qui campe sur la BTS 0. `EGPRS_L1=dsp` (ou `--l1 dsp`) prépare l'autre chemin — `modem` à la place de `mobile` dans `c54x_exe/run.sh` via `MOBILE=/MOBILE_CFG=` dans `/etc/default/osmo-banc` — pour le jour où le firmware l'ouvrira. La ROM Calypso ne module qu'en GMSK : GPRS CS-1..4 au mieux sur le C54x, le 8-PSK reste au modem virtuel.

```
systemctl enable --now osmo-egprs-dsp     # au boot, après osmo-banc (--dsp dans /etc/default/osmo-banc)
scripts/egprs-dsp.sh check                # preflight + configs, ne lance rien
scripts/egprs-dsp.sh status               # unités, mm-context et pdp-context du SGSN, TBF du PCU
scripts/egprs-dsp.sh stop [--all]         # la cellule et le modem (--all : le banc aussi)
journalctl -u osmo-egprs-dsp -u osmo-modem-gprs -u osmo-pcu-gprs -f
```

2.32.1.2 `osmo-dsp-rom-max` — la ROM fait le maximum

`scripts/dsp-rom-max.sh run` pose le profil ROM max dans `/etc/default/osmo-banc` (RACH, SDCCH/SACCH montants et bursts finaux par la ROM, A5 descendant par la ROM, rien « en conserve »), redémarre `osmo-banc` en `--dsp`, joue l'échelle de `/opt/GSM/tests/banc-max.sh --dsp --continue` (LU, SMS, appel, couverture) et relit `couverture.txt` : ce que la ROM a fait, ce qui est resté à l'hôte malgré le profil, et ce qui n'a pas encore de chemin ROM (FACCH et SACCH-en-TCH montants, parole montante, A5 montant au fn de l'air, vocodage). `--fw-gprs` prend le firmware `layer1.highram.gprs.elf` (PDCH, `L1CTL_GPRS_*`). L'unité est posée mais pas activée : le profil persiste, la campagne se lance.

```
systemctl start osmo-dsp-rom-max          # ou : scripts/dsp-rom-max.sh run
scripts/dsp-rom-max.sh report             # relire le dernier bilan
scripts/dsp-rom-max.sh profile            # poser le profil seul
```

2.33 /opt/GSM/osmo-operator/wiki/Build.md

8880 octets, 185 lignes

2.33.1 Build — Docker image, ISO, `.deb` packages, native install

Four paths, one build engine. Everything that gets compiled (libosmocore, osmo-*, gapk, QEMU and its tree, osmocom-bb, the gr-gsm venv, the firmware) comes out as a `.deb` package and stays **on the host**, in `/var/cache/osmo-debs`. The first build compiles everything; later ones only compile what is missing.

```
build.sh ──► image osmocom-nitb ──► start-docker.sh / compose.sh      (Docker)
     │              │
     │              └──► build-iso.sh ──► ISO amd64 / .img arm64   (bootable)
     └──► /var/cache/osmo-debs ──► packaging/build-debs.sh ──► dpkg -i   (no git)
install.sh ──────────────────────────────────────────────────►  native Ubuntu 24.04
```

2.33.1.1 1. `build.sh` — the Docker image

```
sudo ./build.sh [--no-cache] [--lite] [--stp] [--arch=arm64|--arm]
```

Option	Effect
(none)	`osmocom-nitb` image (Dockerfile): core + BTS + QEMU Calypso + gr-gsm + Asterisk
`--lite`	`osmocom-nitb:lite` image (Dockerfile.lite): without the workshops
`--stp`	`osmocom-stp` image (Dockerfile.stp): the inter-STP alone, arm64 possible
`--no-cache`	docker cache ignored **and** `OSMO_DEB_REFRESH=1`: everything is recompiled, the `.deb` cache rewritten
`--arch=arm64`, `--arm`	arm64 cross-build (Raspberry Pi 4)

What the script does, in order:

1. `cd` into the script's directory (the `.desktop` goes through pkexec, which starts in `/root` — without this `cd`, `docker build .` looks for `/root/Dockerfile`).
2. `/usr/local/sbin` in the `PATH`: that is where `packaging/apt-fast-install.sh` puts `apt-fast` (aria2). Same installer on the host, in the images and in the ISO rootfs.
3. Installs Docker + **compose v2 + buildx** — a dependency, not an option. `compose.yaml` describes the `nitb`, `run`, `lite`, `stp` services. `docker build` remains the fallback.
4. Copies `/var/cache/osmo-debs` into `.deb-cache/` (the docker context). The image installs these packages with `dpkg` instead of recompiling.
5. Builds. Every directory compiled in the image comes out as a `.deb` via `packaging/osmo-deb.sh`.
6. Brings what the image produced back into the host cache.

Variables:

Variable	Default	Role
`OSMO_DEB_CACHE`	`/var/cache/osmo-debs`	where the cache lives
`OSMO_DEB_REFRESH`	`0`	`1` = recompile and rewrite (what `--no-cache` sets)

```
./compose.sh debs                    # cache contents
sudo ./compose.sh build [--no-cache] [--lite] [--stp]   # = build.sh
sudo ./compose.sh up [--ms 2] [--phy faketrx] [--no-run]  # hub + osmo-operator-1, HLR populated
sudo ./compose.sh down | ps | logs | shell
```

First build time: 15-20 min (osmocom-bb + ARM toolchain + QEMU).

2.33.1.2 2. `build-iso.sh` — bootable ISO / SD image

```
sudo ./build-iso.sh [options]
```

No option: the four amd64 images — `interstp`, `operator` (normal), `lite`, `desktop`. A **single** docker build (`osmocom-nitb`), then each rootfs derives from it: interstp; normal, from scratch; lite = copy of normal with the workshops removed; desktop = normal + the desktop. Only exception: `--role=interstp` alone builds only `osmocom-stp`.

2.33.1.2.1 What to build

Option	Effect
`--role=operator|interstp`	a single image
`--node=N`	ISO pre-assigned to node N (otherwise the node is chosen at launch, `start-native.sh --node`)
`--lite`	image without workshops
`--desktop`	GNOME, bench status Conky, icons « Lancer le banc GSM », « multi-operator », « osmo-lte », « Pilotes graphiques »
`--all`	everything
`--arm`	arm64 SD image for Raspberry Pi 4, Armbian 24.04 base (`osmo-operator-<…>-rpi4.img`); `--lite` accepted, `--desktop` and `--all` refused
`--version=24.04|22.04`	Ubuntu base
`--kb=fr`	keyboard
`--output=fichier`	ISO file name

2.33.1.2.2 Where the Docker image comes from

By default (amd64) **the image is pulled, not built**, in this order: `ghcr.io/<dépôt>/osmocom-nitb:base-<empreinte du dépôt>` → `:latest` from the same repo → Docker Hub. `build.sh` only runs if none of them answers. The fingerprint is the workflows' one: when it matches, the published image was built from exactly this tree; the other two references don't guarantee that, and the script says so. On arm64 nothing is published: it compiles by default.

Option	Variable	Effect
`--skip-build[=image:tag]`	`OSMO_ISO_SKIP_BUILD`	pull mandatory, no fallback to `build.sh`; `=image` forces the reference. `OSMO_ISO_SKIP_BUILD=0` = never pull
`--build-docker`	`OSMO_ISO_BUILD_DOCKER=1`	the opposite, **and it wins**: builds even if a `--skip-build` is lying around
`--no-cache`		passed to `build.sh`

2.33.1.2.3 What travels in the image

Option	Variable	Effect
(default)		the cached `.deb`s are installed into the rootfs with `dpkg`, then the rest of the image is copied; the packages stay in the ISO's `/var/cache/osmo-debs`
`--without-debs`	`OSMO_ISO_WITHOUT_DEBS=1`	the `.deb`s are used to install the stack but **do not stay**

Option	Variable	Effect
	`ISO_EMBED_DEBS=1`	also embeds the `.deb`s from `packaging/build-debs.sh`

2.33.1.2.4 What starts at boot

The `osmo-banc.service`, `osmo-multi.service`, `osmo-lte.service` units are **installed but not enabled**: a freshly installed machine boots to its desktop. The operator starts things (icon, `launch.sh`, `systemctl start osmo-banc`).

Option	Variable	Effect
`--banc`	`OSMO_ISO_BANC=1`	enables `osmo-banc` at boot
`--multi`	`OSMO_ISO_MULTI=1`	enables `osmo-multi` (`Requires=osmo-banc`, docker)
	`OSMO_ISO_LTE=1`	enables `osmo-lte`

On an already installed machine: `OSMO_MULTI_ENABLE=1 ./addition.sh`. Persistent service options: `OSMO_BANC_ARGS="--grgsm"` in `/etc/default/osmo-banc` (DSP has been the default since 2026-10-01).

2.33.1.2.5 WAN in the ISO

```
sudo ./build-iso.sh --wan --wan-nodes="1:IP:IND 2:IP:IND …" --wan-id=N --wan-ops=N --hub-ip=IP
```

Sets `WAN_AUTO=1` in `/etc/osmo-wan.conf`: the bench brings up the WAN without asking. Without `--node`, **a single ISO serves all nine nodes** — the number is deduced from the local IPs or passed with `start-native.sh --node N`.

2.33.1.2.6 Miscellaneous

Variable	Role
`OSMO_ISO_WORK`	working directory, **on disk** (not `/tmp`, often a tmpfs → « No space left on device »)
`OSMO_ISO_VERSION`	= `--version`

The disk installed by Calamares: accounts from the « Utilisateurs » (Users) screen + root unlocked (no more `osmocom` account from the live system); LUKS offered (LVM in manual mode, `lvm2` included); the « Pilotes graphiques » (Graphics drivers) page = `apt install nvidia-driver-610` if `lspci` sees a card (network required), and the `tools/osmo-drivers.sh` icon afterwards. `/etc/osmocom/coeur.env` (written by the ISO) carries `N_MS`.

2.33.1.3 3. `packaging/build-debs.sh` — installing without git

```
./packaging/build-debs.sh          # osmo-operator, pont, qemu-calypso, calypso-firmware → packaging/dist/
sudo dpkg -i packaging/dist/*.deb  # refuses to install over a git clone at the same path
```

The `pont` package ships `pont.py` and `pont_dsp.py` (executables) and the `pont/dsp/` subpackage since 2026-09-23.

`packaging/snapshot-lte-debs.sh` freezes Open5GS / srsRAN; `build-pmos-kernel-deb.sh` the phone's postmarketOS kernel (PPP to the `ue1` netns).

2.33.1.4 4. `install.sh` — native, Ubuntu 24.04 (noble)

```
sudo ./install.sh                 # all steps
./install.sh --list               # the steps, without doing anything
./install.sh --check              # what is already in place
./install.sh --dry-run
sudo ./install.sh --reinstall     # replays everything: sources updated, configs and shortcuts rewritten
sudo ./install.sh --only bureau   # a single step (deps, sources, build, binaires, configs, bureau…)
sudo ./install.sh --skip build    # all but one
```

The steps live in `install_modules/`. `bureau` installs the same files as the ISO (`data/desktop/*.desktop`, `data/*.svg`) in the menu and on the desktop of root and of the `sudo` user. The native install **does not enable** the systemd units. Launching is still `./start-native.sh`, which delegates to `/opt/GSM/qosmo/run.sh`.

The emulated phone (steps `firmware`, `qosmo`, `c54x`, `grgsm-exe` — group `calypso`, `install_modules/45-calypso.sh`) goes through each repository's installer (`qosmo/install.sh`, `c54x_exe/install.sh`, `grgsm_exe/install.sh`, the same ones the Dockerfile and the ISO call); `sudo ./install.sh --telephone` installs it alone on your own Osmocom stack — see Telephone-emule.

2.34 /opt/GSM/osmo-operator/wiki/Environnement.md

9769 octets, 171 lignes

2.34.1 Environment — who wins, and what reaches QEMU

The test bench is driven by environment variables, at three levels: `start-native.sh` (the preparer), `run.sh` + `run_modules/` (the orchestrator of the fork), and QEMU itself (the `calypso` machine, which reads its `CALYPSO_*` at boot). A variable set in the wrong place, or with the wrong idiom, is silently reset to its default before reaching the last level. This document says where it is read, by whom, and how to check what QEMU **actually** saw.

2.34.1.1 1. Load order

```
1. command line           VAR=x ./start-native.sh          always wins (except § 2)
2. /etc/osmocom/coeur.env N_MS                             `:=`, cannot override anything
3. options                --profile --dsp --node --wan …
4. globals.conf           26 network variables             `=`, but a non-empty caller wins (§ 2)
5. environment/load.env
     modes.env            the profile                      `:=`
     paths.env            where things are                 `:=`
     coeur/radio/reseau/debug.env, then bsp/dsp/fbsb/shunt/rf/armdsp/opcodes.env
     calypso.env          the old monolith                 `:=`, supplies only the rest
6. start-native.sh defaults     RUN_DIR, LOG_DIR, ENCRYPTION, CALYPSO_BRIDGE, MS_COUNT, HOST_IP
7. exec run.sh            the whole environment, intact
8. run_modules/40-qemu.sh → C launcher → QEMU   the manifest is written here
```

All environment files use `: "${VAR:=value}"` — "if not already set". So **the first one to set a value wins**, and that first one is the command line.

2.34.1.2 2. The exception: `globals.conf`

`globals.conf` uses `VAR=value` (not `:=`). On the 26 variables it declares — identity (MCC, MNC, OP_NAME), security (ENCRYPTION, SIM_ALGO, KI, IMSI, IMEI), radio (ARFCN, BAND, BSIC, LAC, CELL_ID, IPA_UNIT_ID, MS_MAX_POWER, RXLEV_ACCESS_MIN, CELL_RESEL_HYST), GPRS (BVCI, NSEI, NSVCI, APN), SMS_SC, MS_COUNT, N_OPERATORS, HOST_IP — it would overwrite the environment if it were sourced on its own. Since 2026-09-22, `start-native.sh` sets aside those of these variables that the caller has set **non-empty**, sources the file (`set -a`), then puts them back, and prints "globals.conf surchargé par l'environnement : …" (globals.conf overridden by the environment). The command line therefore also wins on these 26 variables; an empty value always means "computed" and overrides nothing. Only `start-native.sh` applies this rule, and the header of `globals.conf` still says the opposite ("écrasent leur homonyme dans l'environnement" — they overwrite their namesake in the environment).

Everything it does not declare (`CALYPSO_*`, `MODE`, `LOG_DIR`, `QEMU_*`…) passes through intact. To change one of its 26 values:

```
vim globals.conf
./generate_configs.sh ARFCN=520      # rewrites the line
./generate_configs.sh --force        # regenerates the whole file
```

An empty field = computed (often from the operator number). Only pin the `[auto/operator]` fields if `N_OPERATORS=1`, otherwise the operators step on each other.

Concrete case: `ENCRYPTION`. `ENCRYPTION="a5 0" ./start-native.sh --dsp` gives a5 0; before 2026-09-22 it gave a5 1, silently. With nothing on the command line, the value comes from `globals.conf`; the script's `: "${ENCRYPTION:=a5 1}"` is only a fallback for the bare ISO or a container without `globals.conf`.

2.34.1.3 3. `:=` versus `=` in your own files

Syntax	Effect
`: "${VAR:=value}"`	default; the command line wins
`VAR=value`	**locks it**; no one can override

Use `=` only for what must never be overridden. A `=` slipped into a per-domain file is the classic cause of "my variable has no effect".

2.34.1.4 4. The four gate idioms — `=0` doesn't turn everything off

The QEMU code (`hw/arm/calypso/`) reads its variables in four ways. How to **turn off** a variable depends on the idiom, not on the value:

Idiom in the code	Active when	To turn off
`getenv("X") ? 1 : 0`	the variable **exists**, even at `0`	**`unset X`**
`atoi(getenv("X")) > 0`	value > 0	`X=0`
`*e == '1'`	exactly `"1"`	any other value
default ON + `X_OFF`	by default	set `X_OFF`

So `FB_ENERGY=0 ./start-native.sh` can leave `FB_ENERGY` **active**. Check the idiom at the read site (`grep -rn '"X"' hw/arm/calypso/`) or in the reference of the 312 variables: `hw/arm/calypso/doc/VARIABLES_ENVIRONNEMENT.md` (default, measured effect, mode, idiom, category, dependencies).

2.34.1.5 5. Variables that set other variables

Some variables are **profiles**: `CALYPSO_NATIVE_HELPED=1` also turns on `FB_CORR_ENTRY`, `FB_ENERGY` and `FB_IQ_*`. Removing one of them from the command line does not remove it — it reverts to its default, set by the profile.

2.34.1.6 6. The truth is the manifest

Neither the command line, nor `env`, nor the files say what QEMU saw. At boot, the `calypso` machine writes a manifest to its log:

```
grep "calypso-manifest" /root/qemu.log          # or /var/log/osmocom/qemu.log
```

This is the only reliable source. If a variable is not there at the expected value, it was reset somewhere between 1 and 8 — walk back up the chain in § 1.

2.34.1.7 7. What reaches QEMU

Variable	Default	Read by	Role
`CALYPSO_PROFILE` / `MODE`	`faketrx-qemu` (script), `core` (modes.env)	modes.env, run.sh	the profile; `MODE` wins if set
`CALYPSO_FORK`	`qosmo-grgsm`	paths.env, start-native	`--dsp` no longer changes it; qosmo-dsp has not been built since 17/09, still reachable by hand (`CALYPSO_FORK=qosmo-dsp`)
`OQC_ROOT`	`../<fork>` else `$GSM_ROOT/<fork>`	start-native	the fork's tree; `OQC_ROOT=/path ./start-native.sh`
`RUN_SH`	`$OQC_ROOT/run.sh`	start-native	the `run.sh` that is run (overridable since 2026-08-08)
`QOSMO_LAUNCHER`	`/usr/local/bin/<fork>`	40-qemu.sh	the C launcher; absent → historical `qemu-system-arm`
`CALYPSO_BRIDGE`	`pont` (`none` with `--dsp`)	run.sh	`none` = QEMU + BTS only; with `--dsp`, it is `c54x_exe/run.sh` that launches `pont_dsp.py`; **never `ipc`** on qosmo-dsp (means MS#1 via osmo-trx-ms-ipc, firmware not launched)
`CALYPSO_NO_ATTACH`		start-native	`1` = do not attach to the tmux (`osmo-banc.service`)
`CALYPSO_LANG`		run.sh	language of the messages
`CALYPSO_FIXES`		QEMU (fixes.env)	the airlock for fixes awaiting validation
`QEMU_BIN`	`$GSM_ROOT/qemu/build/qemu-system-arm`	40-qemu.sh	binary (without launcher)
`QEMU_FW`	`$GSM_ROOT/firmware/board/compal_e88/layer1.highram.elf`	launcher	ARM firmware
`QEMU_DSP_ROM`	`$GSM_ROOT/calypso_dsp.txt`	launcher (`-dsp`)	C54x mask-ROM
`QEMU_BRIDGE`	`$OQC_ROOT/bridge.py`	run.sh	the bridge (historical)
`QEMU_L1CTL_SOCK`	`/tmp/osmocom_l2`	launcher, mobile	L1CTL socket
`QEMU_MON_SOCK`	`/tmp/qemu-calypso-mon.sock`	launcher	HMP monitor
`RUN_DIR` / `LOG_DIR`	`/run/osmo-direct` / `$RUN_DIR/logs`	everything	tmpfs — the tmux scrollback depends on it
`GSM_ROOT`	`/opt/GSM`	paths.env	root of the installation
`N_MS`	`1` (coeur.env)	21-abonnes-hlr.sh	subscribers provisioned in the HLR
`MS_COUNT`	`2`	start-native, globals	mobiles launched
`PHY_MODE`	`faketrx`	run.sh (Docker)	`faketrx` / `virtphy` / `qemu`; `qemu` forces `N_MS=1`
`OSMO_STP`, `OSMO_MSC`, …	bare name (PATH)	paths.env	Osmocom binaries; empty = packages

DSP bench (`--dsp`): these variables are read by **c54x_exe** and its `run.sh`, not by QEMU — so they do not appear in the `calypso` manifest.

Variable	Default	Read by	Role
`BANC_DSP`	`$GSM_ROOT/c54x_exe/run.sh`	start-native	the bench launched after the fork's plan; `none` to turn it off
`INSNS`	`120000` (start-native), `80000` (c54x_exe/run.sh alone)	c54x_exe (`--insns`)	DSP instruction ceiling per frame
`LOCKSTEP`	`1`	c54x_exe/run.sh	QEMU only advances the frame once the DSP has finished the previous one
`IQ`	`none`	c54x_exe/run.sh	no synthetic cell: the bursts come from the BTS
`CALYPSO_BSP_STREAM`	`1` (start-native; not set by `c54x_exe/run.sh` alone)	c54x_exe (`calypso_bsp.c` from `qosmo/hw/arm/calypso/l1-dsp`, compiled in; `src/pont.c` only displays it)	one TS0 burst per frame, in FN order
`CALYPSO_RHEA_DMA_XFER`	`1` (start-native)	**no one anymore**	gate removed from the core on 2026-09-18 (qosmo `cf34ab6`): the DMA2 transfer is unconditional; `src/pont.c` only prints the variable in its report. The real gate is `CALYPSO_RHEA_DMA` (default 1, `0` = old silent stub)

The fine-grained domains (`bsp.env`, `dsp.env`, `fbsb.env`, `shunt.env`, `rf.env`, `armdsp.env`, `opcodes.env`) carry 311 variables, of which **116 are crutches**: workarounds for unimplemented branches, each annotated `@BEQUILLE` at its read site with what it hides and the condition that would make it unnecessary (`grep -rn "@BEQUILLE" hw/arm/calypso/`). `crutches.env` is touched only to diagnose; `debug.env` (probes, traces) has no effect on the emulation and can be modified freely.

2.34.1.8 8. Verification recipe

```
./start-native.sh --list                        # the plan: profile, fork, run.sh, launcher
./start-native.sh --dry-run --verbose           # unrolls it, without launching
./start-native.sh --check-paths                 # the declared dependencies
X=1 ./start-native.sh && grep calypso-manifest /var/log/osmocom/qemu.log   # what QEMU saw
grep -rn '"X"' /opt/GSM/qosmo-grgsm/hw/arm/calypso/   # the read idiom
```

If `X` is not in the manifest: either `globals.conf` declares it (§ 2), or a file sets it with `=` (§ 3), or a profile resets it (§ 5), or it is only read by the other fork.

2.35 /opt/GSM/osmo-operator/wiki/Home.md

11699 octets, 212 lignes

2.35.1 osmo-operator — Quick start & reference

A complete **GSM + LTE** operator in a single image, with the radio emulated end to end down to the baseband (Calypso under QEMU), and a postmarketOS phone that does data over 4G and voice/SMS over 2G on the same identity — a real handset's CSFB, at the modem level. No SDR, no physical phone.

Page	Contents
Home (here)	recommended quick start (the ISO), launching, reference
Start-native	`start-native.sh` step by step, options manual
Resultats	reference `check_all.sh` runs, 1 and 3 operators, and how to read the deviations
Build	`build.sh`, `build-iso.sh`, `.deb`, `install.sh`
Environnement	which variable wins, what makes it all the way to QEMU
Reference	the old full README (in English): multi-PLMN architecture, SS7, SMS, voice, diagnostics
Telephone-emule	plug **only** the emulated phone (qosmo, c54x_exe, grgsm_exe) into **your own** Osmocom stack
Lancer-a-la-main · lancer-pas-a-pas.ipynb	the phone launched process by process, no script, with proof at every step
SIM	the bench's card in an Omnikey HID reader (sysmoISIM programmed by pySim) or softSIM, served to the mobile over SAP: `start-native.sh --sap`, `tools/osmo-sim.sh`
WAN	linking two nodes (LAN or Internet via WireGuard): ports, SS7 hub, `--wan-nodes`, layer-by-layer checks, failures
Payload-ISM	SMS **text** between two nodes over an ISM-band radio link, everything else (SS7, SIP, addresses) over IP: `start-native.sh --wan --payload-ism`, `network/setup-payload-ism.sh`
Injecteur-bruit	degrade the downlink between the bridge and layer 1 (bit flips, bursts, AWGN, I/Q): `BRUIT_MODE=…`, off by default
README-notebook · installer-osmo-jusquau-dsp.ipynb	the install notebook: fresh Ubuntu 24.04 to a call on the DSP, step by step, and how to run it

---

2.35.1.1 Recommended method: the desktop ISO (release)

Everything the ISO contains, how it is built and installed, and each of its capabilities: see ISO.

This is the path that works without compiling or configuring anything: the latest release ships `osmo-operator-desktop.iso`, a GNOME live system with Wireshark, Linphone, the I/Q spectra and the three launchers in the dock.

2.35.1.1.1 1. Download and reassemble

GitHub caps a release file at 2 GiB, so the image ships in pieces. **Download all four `part-*`**, then:

```
cat osmo-operator-desktop.iso.part-* > osmo-operator-desktop.iso
sha256sum -c SHA256SUMS
```

2.35.1.1.2 2. Requirements

x86_64, **8 GB of RAM and 4 cores minimum**. The live root is a tmpfs (~6 GB with `toram`): logs and captures there are capped and purged at every boot. The bench runs **two QEMUs** (the Calypso and the pmOS VM): inside a virtual machine, enable **nested virtualization**, otherwise the pmOS VM falls back to software emulation and becomes very slow. On real hardware KVM is available directly: that is the most comfortable setup.

2.35.1.1.3 3. Launch

QEMU

```
qemu-system-x86_64 -cdrom osmo-operator-desktop.iso -m 8G -enable-kvm \
  -cpu host -smp 4 -nic user,hostfwd=tcp::8080-:8080
```

VirtualBox — new Linux / Ubuntu 64-bit VM, 8 GB, 4 CPUs, 3D disabled. System → Processor: tick *Enable Nested VT-x/AMD-V* (if greyed out: `VBoxManage modifyvm "<nom>" --nested-hw-virt on`). Attach the ISO to the optical drive, no disk. NAT network: port forward host 8080 → guest 8080.

VMware (Workstation / Player / Fusion) — VM from the ISO, Linux / Ubuntu 64-bit, 8 GB, 4 cores, tick *Virtualize Intel VT-x/EPT or AMD-V/RVI*, skip easy install.

USB stick — the ISO is hybrid:

```
sudo dd if=osmo-operator-desktop.iso of=/dev/sdX bs=4M status=progress conv=fsync
```

On Windows, Rufus in *DD Image* mode. UEFI or BIOS boot, either works.

2.35.1.1.4 4. Start the bench

Nothing starts at boot: the machine lands on its desktop. Three launchers in the dock, 2G and 4G independent and in any order, **the smartphone last**:

Launcher	Starts
2G — the red handset	osmo-stp / bsc / msc / hlr / sgsn / ggsn / pcu, the Calypso under QEMU, both BTSs, both osmocom-bb mobiles, Asterisk, the SMSC
4G — the signal bars	open5gs, srsENB + srsUE over ZMQ, the SGs interface to the MSC (CSFB)
Smartphone — phone icon	the postmarketOS VM: oFono on the virtual modem, data over 4G, voice and SMS over 2G

Passwords: live **`osmo`**, pmOS phone **`147147`**.

Check that everything is up: `checks/diag-stp-operator.sh`, or the full check `bash ./checks/check_all.sh` (reference result for a standalone native node in Resultats: everything passes, only the SS7 interconnect is *down* for lack of an inter-STP — that is expected).

2.35.1.1.5 5. Make a call

The two provisioned subscribers are **100101** and **100102**, one per cell; `100`/`200` are the Linphones, `600` the echo test. An incoming call to the smartphone is paged by the MME, the UE falls back to the BTS (CSFB), voice goes over 2G; the speedtest, on the other hand, goes over 4G. Everything is visible in Wireshark (GSMTAP, cleartext NAS, SGs, M3UA).

2.35.1.2 Other paths

Path	When	See
Docker — `docker pull bastienbaranoff/norf_gsm`, `sudo ./launch/start-docker.sh`	N operators interconnected through an inter-STP, on an existing host	README-docker
Native — `sudo ./install.sh` then `./start-native.sh`	a dedicated Ubuntu 24.04 machine, no container	Build § install.sh
Build the ISO yourself — `sudo ./build-iso.sh`	modify the image, pre-wired WAN, arm64 for the hub	Build § build-iso.sh
Install to disk — Calamares from the live system	keep the bench, LUKS, NVIDIA drivers	Build

---

2.35.1.3 Reference

2.35.1.3.1 What is in the image (v0.1-9)

`/etc/os-release`: `IMAGE_VERSION="OSMO_EGPRS_V2"` — `/etc/osmo-role`: `OSMO_ROLE=operator`, `OSMO_LITE=0`.

2G / GSM — Osmocom

- Core: osmo-stp (M3UA, SCTP :2905), osmo-bsc, osmo-msc, osmo-hlr, osmo-sgsn, osmo-ggsn (GTPv1-C/U), osmo-pcu, osmo-mgw (MGCP :2427).
- Two DCS1800 cells, LAC 1: BTS 0 (Cell ID 6001, BSIC 7) and BTS 1 (Cell ID 6012, BSIC 8), each on its own osmo-bts-trx.
- Layer 1 (default, `./start.sh --bts-classique --ms`): `osmo-bts-trx` talks to `pont/pont_dsp.py`; the emulated phone (`qemu-system-arm -M calypso`, qosmo) runs osmocom-bb `layer1` + `mobile` on the C54x DSP mask ROM. Other modes: `--ms-magnetite` (FreeCalypso Magnetite on a C139), `--bts` (the Calypso as the BTS radio, `--ts 1 | 2 | 2/1`), `banc-rssi`.
- GPRS/EDGE: BSSGP/NS to the PCU, PDP contexts through osmo-ggsn.
- Voice: external MNCC → Asterisk (SIP :5060, FR/HR) via osmo-sip-connector.
- SMS: SMS-over-GSUP → proto-smsc-daemon, SMPP :2775 with a test ESME, inter-operator relay :7890.
- A3A8 authentication with deterministic RAND (`*-force-rand-toy` patches): reproducible from one boot to the next.
- Live GSMTAP in Wireshark.

4G / LTE — open5gs + srsRAN

- open5gs core (MME, SGW-C/U, SMF, UPF, PCRF, HSS…) on dedicated loopbacks; Prometheus :9090, WebUI :9999, MongoDB behind the HSS.
- PLMN 001/01, TAC 7. NAS security with null ciphering (EIA2/EIA1 + EEA0), on purpose: the NAS reads in cleartext.
- srsENB 10 MHz, EARFCN 3350, band 7; srsUE connected over ZMQ. Milenage subscriber IMSI 001010001000001. The UE has an IP in a netns and goes out through NAT.

CSFB

- SGs interface osmo-msc ↔ MME (:29118), TAI table (001-01, TAC 7) → LAI (001-01, LAC 1): the VLR files the subscriber under LAC 1 while it is on LTE.
- Combined EPS+IMSI attach; incoming call paged by the MME, the UE falls back to the BTS.

postmarketOS

- x86_64 VM, patched kernel (`CONFIG_PPP`, missing from upstream `linux-postmarketos-stable`) shipped prebuilt (Git LFS) in `/opt/user_interface/kernel/pmos/`.
- 4G data: NetworkManager → pppd (`ATD*99***1#`) → virtio-console `osmo.data` → `osmo-phonesim-banc.py` → UE netns → srsUE → srsENB → UPF → Internet.
- 2G voice/SMS: second virtio-console `osmo.modem`, `osmo-phonesim-banc.py` plays the 27.007 AT modem (state read from the bsc, msc and mobile VTYs), oFono drives it.

- Audio: two full-duplex PulseAudio chains (intel-hda) — a GAPK bridge to the osmocom-bb mobile, and a chain with echo cancellation. Slimmed down at every start (speech-dispatcher stopped, `osmo_rec` at 8 kHz mono, stream-restore without device restore). webrtc canceller by default, its `aec_args` passed as a single argument — without that the AGC muted the mic; `AUDIO_AEC_METHOD=speex` is still available.

2.35.1.3.2 Ports and access

Port	Service	Port	Service
4239	osmo-stp VTY	8080	bench dashboard
4242	osmo-bsc VTY	8081	I/Q spectra (fft-web)
4243	osmo-mgw VTY	9090	open5gs Prometheus
4254	osmo-msc VTY	9999	open5gs WebUI
4258	osmo-hlr VTY	5060	Asterisk SIP
4729	GSMTAP (Wireshark)	2775	SMPP
2905	local M3UA	7890	inter-op SMS relay
2908	inter-STP M3UA	29118	SGs (MSC ↔ MME)

```
tmux attach -t calypso          # the 2G stack; Ctrl-b d detaches
journalctl -u osmo-banc -f      # if started as a service
./start-native.sh --status | --stop
./ss7-console.py                # browsable SS7 diagram, built-in VTYs
```

2.35.1.3.3 Numbering plan

Number	Use
`100101`, `100102`	the bench's two subscribers (one per cell)
`N0001…N9999`	subscribers of operator N (multi-op)
`100` / `200`	Linphone A / B
`600`	echo test
`9XXXXX`	inter-op breakout from a softphone

2.35.1.3.4 Most useful options

Command	Effect
`./start-native.sh --list`	the plan, without launching
`./start-native.sh --menu`	asks the questions instead of guessing
`./start-native.sh --regen`	regenerates the configs from the templates
`./start-native.sh`	C54x DSP bench (c54x_exe, TI mask ROM), **the default** since 2026-10-01: the mobile camps and does its LU
`./start-native.sh --grgsm`	gr-gsm layer 1 inside QEMU (the old default): the stack goes all the way to a voice call
`./start-native.sh --node 2 --hub-ip IP`	this node = node 2 of a WAN, ASP towards the inter-STP
`./start-native.sh --wan`	N-node WAN, interactive questions
`./generate_configs.sh ARFCN=520`	changes a value in `globals.conf`
`sudo ./build-iso.sh --desktop`	rebuild the desktop ISO
`bash ./checks/check_all.sh`	full check, log in `/tmp`

2.35.1.3.5 Files worth knowing

File	Role
`globals.conf`	the network configuration; a non-empty variable set on the command line overrides it (since 2026-09-22)
`environment/load.env`	the loading order of profiles and defaults
`/etc/osmocom/coeur.env`	`N_MS`, written by the ISO
`/etc/osmo-wan.conf`, `/etc/osmo-role`	WAN table, node role
`/etc/default/osmo-banc`	`OSMO_BANC_ARGS="--grgsm"` to make the service start in gr-gsm (DSP otherwise)
`/var/log/osmocom/{qemu,bridge,run.sh}.log`	QEMU, bridge, orchestration
`/tmp/c54x-pont/`, `/dev/shm/pont.log`	DSP bench: links to its logs (real files at the dashboard paths), bridge log
`pont/README.md`	read it **before** hunting for a radio fault in a CRC counter

2.36 /opt/GSM/osmo-operator/wiki/ISO.md

49607 octets, 839 lignes

2.36.1 The osmo-operator ISO — full description, build, installation, capabilities

This page describes **everything the ISO is**: what it contains, how it is built (`build-iso.sh` and its modules), how it boots, how it installs to a disk, and everything it can do once booted. It points to the repository files for each item; the Build, Start-native and Environnement pages detail the scripts themselves.

Reference state: **osmo-operator-desktop.iso v0.1-9** (tag `v0.1-9`, commit `eeda9a2`). It includes the 2026-10-01 changes described below (DSP by default, LUKS question on the console, GTK launchers); what is new in 0.1-9 is listed in `.github/release-notes.md`. The sizes in § 1 were measured on v0.1-6.



Figure 5: The ISO desktop: Telecom Operator, no RF, full stack

---

2.36.1.1 What's inside — a mobile operator on one USB stick

Boot it, click one icon, and you are running a **whole GSM network with its phones**: no radio, no SDR, no SIM card required. Everything below is in the image. Nothing starts by itself; you start it.

A real 2G core, end to end — osmo-bsc, osmo-msc, osmo-hlr, osmo-stp, osmo-sgsn/ggsn, osmo-pcu (GPRS/EDGE), osmo-mgw, Asterisk, an SMSC. Two cells, two subscribers, A5/1, location update, SMS, **audible voice calls**, GSMTAP straight into Wireshark. → § 5.1

☎ **Real phone firmware on an emulated phone** — a QEMU fork (`qosmo`) runs the Calypso baseband of a Motorola C123 with the genuine OsmocomBB firmware, and the **TI TMS320C54x DSP runs on its own mask ROM**: the emulated phone syncs to the cell (FCCH/SCH), reads SI1-4 and camps. → § 5.2, § 5.3

The phone can even *be* the base station — `./start-native.sh --bts` turns the emulated Calypso into the transceiver of the BTS; `--bts --ms` adds a second Calypso that camps on it **over an emulated air interface**. A fourth one runs the RSSI app. One command each: `banc-trx`, `banc-mobile`, `banc-rssi`. → § 5.13

Hold the phone in your hand — `c123` opens a photo of the Motorola C123 with its live LCD inlaid and clickable keys (or your PC keyboard). → § 5.13

4G next to it, with fallback to 2G — Open5GS core, srsENB and srsUE over ZeroMQ, data out to the Internet, and **CSFB** over SGs back to the 2G MSC. → § 5.6

A smartphone on the network — a postmarketOS/Phosh VM whose modem is the bench: SMS and voice through 2G, data through 4G. → § 5.7

Several operators, roaming between them — SS7 with an inter-STP hub, 1 to 9 nodes across machines (`--node`, `--wan`), operators in Docker next to the native one. → § 5.4, § 5.5

A desktop made for the bench — GNOME with a city wallpaper that **raises its antenna when a network starts**, a live I/Q spectrum panel, a timeslot banner, a Conky status board, a **GLava spectrum circle dancing on the speakers and the microphone**, Wireshark as root with ready-made filters, Linphone on extension 100, a web dashboard and a tutorial. Every launcher opens a progress window, never a bare terminal. → § 5.8, § 5.13

Tools to look inside — gdb on the emulated baseband, VTYs for every daemon, `ss7-console.py`, `checks/check_all.sh`, a ring-buffer tcpdump. → § 5.10, § 5.11

Live, in RAM, or installed — hybrid ISO (UEFI/BIOS, Secure Boot), `toram` entry, persistence, and a Calamares installer with an optional LUKS2-encrypted `/home`. → § 3, § 6

One click for more — the *Supplements* icon adds the multi-operator Docker bench, OpenCL + deka (A5/1 tables), Claude Code, games and media, and extra phone firmwares (FreeCalypso Magnetite on a C139, MediaTek Fernvale, osmocom-bb-trx). → § 5.5

On ARM boards too — the same stack as SD images built by armbian/build for the **Raspberry Pi 4** and **every Orange Pi 5** (5, 5B, 5 Plus, 5 Pro, 5 Max, 5 Ultra), with or without the bench desktop. → § 2

2.36.1.2 1. At a glance

Item	Value
Base	Ubuntu 24.04 (noble), kernel `linux-image-generic` 6.8.0-146
Size (desktop)	ISO 6.5 GB · squashfs 6.3 GB (zstd -19, 41 % of 15.9 GB) · 314,178 files
Boot	hybrid ISO (dd to a USB stick, UEFI or BIOS), Secure Boot shim → grub → kernel, `toram` entry (9 GB RAM)
Live accounts	`root` autologin (pw `osmo`) · `osmocom` sudoer (pw `osmo`) · pmOS phone `147147`
Build	`sudo ./build-iso.sh --desktop`, ~2 h 15 on 16 cores; Docker image pulled from ghcr.io, no recompilation; minimal debootstrap + the build's `.debs` (~41); ~1,650 packages in the chroot; dpkg integrity check before and after squashfs
2G	Full bench (`launch.sh`, `osmo-banc`), 2 MS, A5/1, inter-operator SMS, MSC with SGs (CSFB)
Emulated baseband	QEMU `qosmo` (ARM7 + C54x DSP), 7 DSP ROMs, layer1 firmware for 10 boards, gdb/telnet, `c54x_exe`, `grgsm_exe`
4G	portable srsENB/srsUE + srsGUI, Open5GS, freeDiameter TLS, MME SGsAP, HSS console `:9999`, subscribers in Mongo
Phone	postmarketOS VM 1.7 GB (PPP kernel), patched pmbootstrap, pmaports `9479a409e3`
Desktop	minimal GNOME, Firefox (Mozilla deb, fr, uBlock), Wireshark, Linphone (extension 100), Kodi, dino, oFono, Conky, Plymouth `osmo-bts`, web dashboard + tutorial
Audio	system-wide PulseAudio, PipeWire for screencast only, PCM `gsm_out`/`gsm_in`
Removed	avahi, Chromium, sssd masked, pipewire-pulse
Installation	Calamares (`osmo-install`), optional LUKS2-encrypted `/home`, `osmo-update` at boot
Services installed, not enabled	`osmo-banc`, `osmo-multi`, `osmo-lte` (`--banc`, `--multi`, `OSMO_ISO_LTE=1` at build time to enable them)

2.36.1.3 2. Images produced

A single chain (`build-iso.sh` + `iso_modules/`) produces the four amd64 images, built back to back with no option (`--all`); `--arm` hands over to the Armbian framework for the SD images.

Image	File	Role	Contents
SS7 hub	`interstp.iso`	`--role=interstp`: the inter-STP alone (PC 0.0.0), no operator; rejects `--lite` and `--desktop`	`osmocom-stp` image (Dockerfile.stp), libosmocore/netif/sigtran only, `osmo-interstp.service` at boot
Operator	`osmo-operator.iso`	`--role=operator` (default): the full bench, no desktop	2G stack, emulated baseband, 4G, dashboard, tools; text console
Lite	`osmo-operator-lite.iso`	`--lite`: the same minus the build workshops	`/opt/GSM` stripped of source trees (`iso_modules/87-lite.sh`, same rules as `Dockerfile.lite`)
Desktop	`osmo-operator-desktop.iso`	`--desktop`: + GNOME, Wireshark, Linphone, VLC, Calamares, Conky, Firefox (~2.5 GB more)	this is the image published in releases
ARM boards	`Armbian-…_<Board>_noble_….img` (CI: `osmo-operator-<board>[-desktop].img`)	`--arm`: one arm64 SD image **per board**, built by Armbian/build (`armbian/build-armbian.sh`): `rpi4b` and every Orange Pi 5 — `orangepi5`, `orangepi5b`, `orangepi5-plus`, `orangepi5pro`, `orangepi5-max`, `orangepi5-ultra` (`--board=rpi4|opi5|all|<board>`, default `all`)	Armbian builds kernel, BSP, u-boot/firmware, rootfs and image; the stack comes from the arm64 Docker image through `armbian/userpatches/customize-image.sh`. `--desktop`: GNOME (Armbian configng, tier minimal) + the bench desktop, without GLava (OpenGL 4.2 missing on these GPUs). No 4G, no `--lite`. The first Docker build is emulated: about ten hours

With `--node=N` (1 to 9), the name becomes `osmo-operator-N[-lite][-desktop].iso` and the image carries the SS7 identity of node N. `--output=FILE` forces the name.

2.36.1.4 3. Getting it, booting, finding your way

2.36.1.4.1 3.1 Download

The latest release ships the desktop ISO **in 1,900 MB parts** (GitHub caps a release file at 2 GiB):

```
cat osmo-operator-desktop.iso.part-* > osmo-operator-desktop.iso
sha256sum -c SHA256SUMS
```

2.36.1.4.2 3.2 Machine requirements

x86_64, **8 GB of RAM and 4 cores minimum**. The bench runs two QEMUs (the Calypso and the postmarketOS VM): in a virtual machine, enable nested virtualization. On real hardware, KVM is available directly.

2.36.1.4.3 3.3 Booting

- **QEMU**: `qemu-system-x86_64 -cdrom osmo-operator-desktop.iso -m 8G -enable-kvm -cpu host -smp 4 -nic user,hostfwd=tcp::8080-:8080`
- **VirtualBox**: Linux/Ubuntu 64-bit VM, 8 GB, 4 CPUs, 3D disabled, *nested VT-x/AMD-V* checked, NAT with port forwarding 8080 → 8080.
- **VMware**: VM from the ISO, *Virtualize Intel VT-x/EPT* checked, skip Easy Install.
- **USB stick**: `sudo dd if=osmo-operator-desktop.iso of=/dev/sdX bs=4M status=progress conv=fsync` (Rufus in *DD Image* mode on Windows). UEFI or BIOS, either works.

2.36.1.4.4 3.4 The GRUB menu (`iso_modules/90-iso.sh`)

Entry	Kernel line	Use
Run from medium (default)	`boot=live quiet`	the stick stays plugged in, root on a tmpfs overlay
In RAM	`boot=live toram=filesystem.squashfs`	copies only the squashfs to RAM (stated: 9 GB), the stick can be removed
Persistence	`boot=live persistence persistence-encryption=none quiet`	`persistence` partition on the stick to keep changes

The live stick boots **without** `splash`: Plymouth does not show there. The `osmo-bts` theme (mast, handset and waves) appears on an installed disk, where GRUB boots with `quiet splash`.

2.36.1.4.5 3.5 Secure Boot (`iso_modules/91-secure-boot.sh`)

The ISO is assembled by xorriso with the **Canonical-signed shim** and the signed GRUB (`shim-signed`, `grub-efi-amd64-signed`), plus a `/.disk/info` so the signed GRUB can find the medium. The shim → grub → Ubuntu kernel chain is complete: the image boots with Secure Boot on. Without these packages on the build host, the ISO comes out unsigned and the build says so.

2.36.1.4.6 3.6 The live system: space, logs, network (`83-espace-ram.sh`, `81-cloture-systeme.sh`)

- `/dev/shm` and `/tmp` are capped as a percentage of RAM; I/Q cfiles, logs and captures live there and are purged at every boot. A `tcpdump` wrapper writes a pcap ring buffer.
- No network wait at boot: the `*-wait-online` units are masked, shutdown is capped at 30 s. DNS through the `systemd-resolved` stub with a fallback `resolv.conf`. NetworkManager ignores `apn*`, `tun*`, `veth*`, `docker*`.
- `osmo-ip-plan.service` derives the node's private addresses; `osmo-qemu-link` links `QEMU_BIN` and rebuilds the `qosmo` launcher if needed.
- `osmo-update.service` does a fast-forward `git fetch` of the embedded repositories at every boot (never a reclone, never apt), log in `/var/log/osmo-update.log`.
- When an interactive shell opens, an "SMS delivered" animation runs once (`/usr/local/sbin/osmo-sms.sh`, formerly `update.sh`) and reinstalls the icons.

2.36.1.4.7 3.7 Accounts and sessions (`84-comptes.sh`)

Account	Password	Role
`root`	`osmo`	the bench's working account: autologin on tty1, ttyS0 (ttyAMA0 on Pi) and in GDM
`osmocom`	`osmo`	unprivileged, sudoer (sudo, adm, dialout, audio, video, pulse-access, docker); session selectable after logout

Account	Password	Role
`linphone_A`	—	pre-provisioned SIP account: extension **100**, UDP to 127.0.0.1:5060
pmOS `user`	`147147`	Phosh screen, account and ssh of the postmarketOS VM

The bench runs as root. Icons elevate via `pkexec`, the command line via `sudo`.

2.36.1.5 4. What the image contains

2.36.1.5.1 4.1 Useful tree

Path	Contents
`/opt/GSM/osmo-operator`	**the full repository, with its `.git`** (clone of the `main` branch at build time, `51-depot.sh`): scripts, configs, services, tools, wiki
`/opt/GSM/qosmo`	Calypso QEMU fork (`run.sh`, `run_modules/`), C launcher `tools/qosmo-launch`
`/opt/GSM/c54x_exe`	the TMS320C54x DSP outside QEMU, TI mask ROM in `rom/`, its `run.sh` (5 steps)
`/opt/GSM/grgsm_exe`	the gr-gsm layer 1 outside QEMU
`/opt/GSM/firmware/board/compal_e88/`	OsmocomBB firmware `layer1.highram.elf` and the ten boards
`/opt/GSM/osmocom-bb`	osmocon, mobile, trxcon
`/opt/user_interface/pmos/`	patched pmbootstrap, PPP kernel, VM image `image/qemu-amd64.img.zst` (if embedded), `bin/` launchers
`/etc/osmocom/`	generated Osmocom configs (PLMN 001-01, A5/1, 2 MS), `coeur.env` (`N_MS`, `OPERATOR_ID`), `osmo-multi.conf` after the add-ons
`/etc/asterisk/`, `/etc/asound.conf`	voice PBX, PCM `gsm_out`/`gsm_in`
`/var/cache/osmo-debs/`	the bench `.debs` (osmo-operator, bridge, qemu-calypso, calypso-firmware) for a git-less install
`/usr/share/osmo-operator/`	`info.html` (Info page), SVG icons
`/var/log/osmocom/`	`qemu.log`, `bridge.log`, `run.sh.log`

2.36.1.5.2 4.2 Packages and software chain (`80-chroot.sh`, `50-injection-image.sh`, `81-cloture-systeme.sh`)

- Base: `debootstrap --variant=minbase` noble, systemd, dbus, kmod, zstd, curl, iproute2, `live-boot`, generic kernel, Osmocom libs, tshark, openssh, cryptsetup.
- Bench: asterisk, pulseaudio, ffmpeg, MongoDB 8.0, NodeSource (dashboard), srsRAN and Open5GS dependencies, `build-dep gnuradio`, venv `/root/.env`.
- **The compiled binaries come from the `osmocom-nitb` Docker image** as `.debs` (`osmo-build-*~noble_amd64.deb`) installed with `dpkg -i`, then `/usr/local/*`, `/opt/GSM`, `/root/.env`, `/opt/node`, `/etc/osmocom`, `/etc/asterisk` and the `osmo-*.service` units are copied from the container.
- **`ldd` closure**: every `.so` used in the Docker image is copied into the rootfs; `/usr/local/lib` goes first in `ldconfig`. The environment that works is the image's, not a different apt version.
- Desktop: `ubuntu-desktop-minimal`, wireshark, linphone-desktop, vlc, calamares, conky, `python3-gi`, `python3-gi-cairo`, `python3-cairo`, `gir1.2-gtk-3.0`, `gir1.2-vte-2.91`, GLava (compiled, `85-glava.sh`), Firefox from packages.mozilla.org, GDM with root autologin, no Wayland.
- Integrity check: the rootfs is verified against dpkg **before** compression, then the squashfs is read back and re-verified (`90-iso.sh`).

2.36.1.6 5. Capabilities, one by one

General rule: **nothing 2G/4G starts by itself at boot**. Starting is the operator's call, via the icon, the bottom bar, the command line or `systemctl`. The units are installed and ready.

2.36.1.6.1 5.1 The standalone 2G bench

Full native Osmocom stack, one operator (PLMN 001/01), two DCS1800 cells (ARFCN 514, LAC 1; BTS0 Cell ID 6001 BSIC 7, BTS1 Cell ID 6012 BSIC 8), subscribers `100101` and `100102`.

- Components: osmo-stp, osmo-bsc, osmo-msc, osmo-hlr, osmo-sgsn, osmo-ggsn (GTPv1), osmo-pcu (GPRS/EDGE), osmo-mgw, Asterisk (MNCC voice via osmo-sip-connector, SIP :5060), SMSC (SMS-over-GSUP, SMPP :2775, inter-operator relay :7890). A3A8 auth with deterministic RAND. Live GSMTAP :4729 for Wireshark.
- Start: red icon **"Lancer le banc GSM"** (Start the GSM bench; `launch.sh`, which starts `osmo-banc.service` and reports it by notification); bottom bar, **Banc** group, **run standalone**; `sudo ./start-native.sh` in the foreground; `systemctl start osmo-banc`. The stack lives in tmux: `tmux attach -t calypso`.
- Stop: right-click **Arrêter le banc** (Stop the bench), `./start-native.sh --stop` (also tears down the DSP bench, no need to repeat `--dsp`), `systemctl stop osmo-banc`.
- Sources: `start-native.sh` (delegates to `/opt/GSM/qosmo/run.sh`), `launch.sh`, `services/osmo-banc.service`, `configs/*.cfg`, `generate_configs.sh`.

2.36.1.6.2 5.2 Two layer-1 chains: DSP (default) and gr-gsm

	DSP `--dsp` (default since 2026-10-01)	**gr-gsm** `--grgsm`
Who demodulates	the real TMS320C54x on the TI mask ROM, **outside** QEMU (`/opt/GSM/c54x_exe`)	a gr-gsm decoder in QEMU + `grgsm_exe` bridge
QEMU	ARM7 only (`CALYPSO_DSP_EXTERN=1`)	ARM7 + gr-gsm layer 1
TRX bridge	`pont/pont_dsp.py` started by `c54x_exe/run.sh` (TRXD 5700-5702, DL bursts UDP 6702)	`pont/pont.py`
Result	the mobile acquires FB/SB, decodes SI1-4, camps, does its location update; mobile VTY :4347	end to end: camp, LU, auth, A5/1, SMS, voice call; VTY :4247

The default lives at the top of `start-native.sh` (`DSP_MODE=1`) and applies to everything that starts the bench (icon, service, bar, multi's native operator). If `c54x_exe` is missing, the bench says so and falls back to gr-gsm with its bridge. Icon right-click: **"Lancer en mode gr-gsm"** (Start in gr-gsm mode). Persistent: `OSMO_BANC_ARGS="--grgsm"` in `/etc/default/osmo-banc`.

2.36.1.6.3 5.3 The emulated Calypso baseband

- QEMU fork **qosmo** (`-M calypso`, ARM7TDMI/arm946) runs the real OsmocomBB firmware `layer1.highram.elf`, loaded by osmocon as on a C123. Seven DSP ROMs, firmware for ten compal boards.
- C launcher `qosmo` (`/usr/local/bin/qosmo`): `-gdb tcp::1234`, `-serial pty` published as `<RUN_DIR>/modem.pty`, unix monitor, L1CTL `/tmp/osmocom_l2`.
- Debug: gdb on :1234 (`target remote :1234` against the layer1 ELF), monitor `socat - unix-connect:/tmp/qemu-calypso-mo`, `--assembly-logs` for the ARM asm trace.
- Manual: `qosmo -k …/layer1.highram.elf`, `MODE=dsp PONT=1 /opt/GSM/c54x_exe/run.sh`.

2.36.1.6.4 5.4 `start-native.sh` profiles and options

- Profiles: `faketrx-qemu` (default, alias `hybrid`: core + BTS0 QEMU + BTS1 fake_trx + MS#2), `faketrx`, `qemu` (Calypso only), `noproc/core` (core only).
- `--menu` asks the questions (layer 1, standalone/hybrid topology, profile, node, WAN, regeneration) instead of guessing.
- `--node N` (1-9): the node's SS7 identity (point codes `1.<node><op>.<role>`, routing contexts), ASP towards the inter-STP; `--op N`, `--hub-ip ADDR`.
- `--wan`, `--wan FILE`, `--wan-nodes "1:IP:IND …"`, `--wan-id`, `--gen-conf`, `--air-mesh[=PORT]` (shared inter-node radio medium): mesh of 1 to 9 nodes. `--virtualbox[=N]` sets up the WAN between the host and N-1 VirtualBox VMs (`tools/vbox-wan-lab.sh`).
- `--regen`, `--list`, `--dry-run`, `--status`, `--check-paths`, `--force`, `--verbose`, `--launcher <bin>`, exclusive L2 clients `--mobile`, `--ccch_scan`, `--bcch_scan`, `--cell_log`.
- Variables: `ENCRYPTION="a5 1"`, `MS_COUNT=2`, `CALYPSO_BRIDGE`, `BANC_DSP`, and any `globals.conf` variable set non-empty on the command line (see Environnement).

2.36.1.6.5 5.5 SS7 multi-operator (Docker) and the add-ons

Not in the ISO: the **"Supplements"** icon (`addition.sh`, checkbox window, inside a bench GTK window) installs what is missing.

- **`--multi`**: docker.io, the operator image pulled from `ghcr.io/bbaranoff/osmo-operator/osmocom-nitb:latest` then derived into `osmocom-run` (Dockerfile.run), the topology `/etc/osmocom/osmo-multi.conf` (op1 **native** PC 1.1.2 + op2/op3 **docker** 172.20.0.12/.13 PC 1.2.2/1.3.2 + hub `osmo-inter-stp` 172.20.0.10, M3UA :2908), `osmo-multi.service` and the antenna icon **"multi-operator"**. It ends by starting a multi bench session via `launch.sh --multi`, the same path as the icon (`OSMO_MULTI_START=0` to install without starting). `addition.sh --status` shows docker, the image, the topology, the unit and the icon.
- Then start: antenna icon, bar **run multi**, `systemctl start osmo-multi`, `sudo ./launch/start-multi.sh` (`--status`, `--stop`, `--dry-run`). The native operator is held by `osmo-banc`: `start-multi.sh` **keeps** it if already running and **starts** it if stopped (it only restarts it if its point code was just realigned, or with `--restart-natif`).
- **`--opencl`**: OpenCL stack (ocl-icd, clinfo, pocl, detected Intel/Mesa/NVIDIA driver) and the tools cloned into `/root`: **deka**, a51_tools, dst80_reversing, tea1-cracker. **deka** icon, bar **run deka**.
- **`--claude`**: Claude Code (native installer, standalone binary). The **Claude** icon self-installs on first click.
- **`--extras`**: Doom (freedoom), Quake III (OpenArena), Kodi, YouTube (Firefox + uBlock), Wireshark as root, Linphone, filed in the bar's Jeux, Media, Telephone, Outils folders.

2.36.1.6.6 5.6 4G (srsRAN + Open5GS) and CSFB

- Full **Open5GS** core on loopbacks (MME, SGW-C/U, SMF, UPF, PCRF, HSS, NRF…), MongoDB behind the HSS, **freeDiameter** with TLS certificates installed, Prometheus :9090, **WebUI :9999** (`open5gs-webui.service`, Next.js) for subscribers.
- **srsENB** radio (10 MHz, EARFCN 3350, band 7) and **srsUE** linked over **ZeroMQ** (:2000/:2001), portable binaries (no build-CPU AVX), **srsGUI** for real-time traces (`osmo-lte start-gui`). Milenage subscriber IMSI 001010001000001, UE in a `ue1` netns, data via `ogstun` (10.45.0.0/16) with NAT to the Internet.
- **CSFB**: SGs interface between osmo-msc and the MME (:29118), mapping TAI (001-01, TAC 7) → LAI (001-01, LAC 1).
- Start: **"4G du banc"** icon (Bench 4G; `osmo-lte toggle` → `osmo-lte.service`); `osmo-lte start|stop|status|up|down|log` core only `osmo-epc start|stop|status`. Log and status in bench GTK windows.
- Sources: `tools/osmo-lte.sh`, `tools/osmo-epc.sh`, `tools/osmo-lte-install.sh`, `configs/srsran/`, `configs/open5gs/`, `services/osmo-lte.service`, `iso_modules/88-lte-pmos.sh`.

2.36.1.6.7 5.7 The postmarketOS phone

- x86_64 postmarketOS/Phosh VM (1.7 GB compressed), kernel patched with PPP, patched pmbootstrap (modem serial port, sound cards, screen size).
- **4G data**: NetworkManager → pppd (`ATD*99***1#`) → virtio-console `osmo.data` → `osmo-phonesim-banc.py` → UE netns → srsUE → srsENB → UPF → Internet. **2G voice and SMS**: virtio-console `osmo.modem`, the 27.007 AT modem is played by `osmo-phonesim-banc.py` based on the bsc/msc/mobile VTYs; oFono drives it on the VM side.

- Start: **Téléphone** icon (`osmo-pmos up`), **smartphone** (720x1440) and **tablet** (1280x800) formats (`osmo-pmos-qemu`), status, stop and reconnect-modem actions (`osmo-pmos-setup`), VM alone without modem.
- **The image**: `osmo-pmos-build` decompresses `/opt/user_interface/pmos/image/qemu-amd64.img.zst` on first launch. If it is missing or corrupt (`zstd -t`), it is fetched from the GitHub release `pmos-image` (resumable, archive verified); otherwise `pmbootstrap install` builds it. `OSMO_PMOS_IMAGE_URL` changes the URL.
- Sources: `tools/osmo-pmos*.sh`, `tools/osmo-phonesim-banc.py`, `tools/osmo-ofono-bridge.py`, `iso_modules/88-lte-pmos.sh`

2.36.1.6.8 5.8 The desktop

- Minimal **GNOME**, root session, dock with the favorites: Firefox, Files, Claude, Lancer le banc, Installer, Tutoriel, Info, Supplements, multi-operator, Update, Paint, deka, Linphone, Wireshark, Téléphone, 4G. The **multi-operator** icon is installed only by the add-ons (`addition.sh --multi`): before that, it does not exist and the dock does not show it.
- **No favorite opens a terminal or a shell.** Everything goes through `tools/osmo-gtk-run.py`. Since 2026-10-12, launches (2G, multi, 4G, add-ons, Update, deka, pmOS phone, drivers) open in `--progress` mode: a small window with a **progress bar** (driven by the `[n/m]` markers the scripts print: `run.sh` [n/10], `osmo-lte` [n/4], `osmo-pmos` [n/3]; otherwise the bar pulses), the current step, the last line, and a **"console"** button that expands the full output (VTE terminal, the command runs in a pseudo-terminal). A failure expands the console automatically. Only Claude and oFono, being interactive, keep a terminal in the window. The systemd unit's journal scrolls in the window for 2G and multi (`launch.sh --service-suivi`, `--multi-suivi`) and 4G (`osmo-lte up|toggle`).
- **Mobile VTY port**: 4247, or **4347 with `--dsp`**. `start-native.sh` publishes the actual port in `/run/osmo-banc/ms1-vty` and in the systemd environment (`OSMO_BB1_VTY_PORT`, read by the `osmo-egprs-web` dashboard on restart); `launch.sh --vty`, the panel and `vty-menu.sh` read it.
- **Bottom bar** (`tools/osmo-launcher.py`): **removed from the desktop** on 2026-10-12 (bench decision). The file remains; `OSMO_LAUNCHER=1` in the session environment brings it back (Banc, Jeux, Media, Telephone, Outils groups, "run" logs).
- **Panel** (`tools/osmo-panel.py`, `osmo-fft-snap.py`): the illustration becomes the I/Q spectrum and `mobile.log` while the bench runs; top banner (`osmo-topzone.py`, `configs/conky/labgsm.html`) fed by `osmo-ts-probe.py`: 8 timeslots, daemons, subscribers, calls, uptime.
- Status **Conky** at the top right (machine, selected operator, `Ctrl+AltGr+←/→` or `osmo-op --next`). Redone on 2026-10-12: dark translucent plate with rounded corners (`configs/conky/osmo-conky.lua`, Cairo), Ubuntu font throughout, colors by kind of information, dots and a smiley per bench state (green if running, grey otherwise). It is **on screen only while a bench runs**: `osmo-panel.py` fades it in and out (X11 window opacity, xdotool) together with the panel.
- **Central panel** (`tools/osmo-panel.py`, spectrum + log, Dashboard / tmux / VTY / ZMQ buttons): **appears only while a bench runs** - a 2G (native, container, hub) or a 4G (Open5GS, srsENB, srsUE). With the bench stopped, the desktop shows only the background. `OSMO_PANEL_TOUJOURS=1` keeps it on screen.
- **Web dashboard** :8080 (`osmo-egprs-web.service`, self-signed HTTPS on :80, GSMTAP/SCTP capture), **tutorial** (`osmo-tutorial`), **Info page** (`osmo-info`, `configs/info.html`), I/Q spectra `fft-web` :8081.
- Applications: Firefox (only browser, microphone and certificate policy set at boot), **Wireshark as root** (`osmo-wireshark-root`, 2G/4G/call/SMS/data filters), **Linphone** extension 100, Kodi, dino, oFono, Paint (overlay), graphics drivers (`tools/osmo-drivers.sh`, NVIDIA via `ubuntu-drivers`).
- **Plymouth** `osmo-bts`: mast, handset and waves (rings), nothing else (preview: `wiki/plymouth-osmo-bts.gif`). Generated at build time by `tools/plymouth-render.py` (`86-finitions.sh`).
- **Two-state** wallpaper (`tools/osmo-fond.py`, `osmo-fond ville|antenne`): at rest `/usr/share/backgrounds/gsm-lab-ville.j` (the waterfront city, no antenna — the schema default); as soon as a 2G or 4G starts, a fade (~2.4 s, Pillow intermediate frames + GNOME Shell cross-fade) to `gsm-lab-antenne.jpg` (the same city, mast raised), and back on stop. Triggered by `osmo-panel.py` when it sees the bench come up or go down, by `osmo-lte start` and, if the bottom bar is active, on clicking *run standalone*. Likewise, the top zone carries only the **Telecom Operator** title while nothing runs, and fades to the LAB GSM banner on launch (`$XDG_RUNTIME_DIR/osmo-banc.etat`, read by `osmo-topzone.py`). The low-poly landscape `gsm-lab-fond.jpg` and `gsm-lab-wallpaper.jpg` (`tools/paysage-render.py`) remain installed, but are not the default.

2.36.1.6.9 5.9 Audio

- **System-wide PulseAudio** (`osmo-pulse.service`, socket `/run/pulse/native`, sink `gsm_audio`): osmo-gapk → ALSA `gsm_out` → null sink → monitor → loopback; PCM `gsm_out`/`gsm_in` in `/etc/asound.conf`, which

`mobile` opens. Two duplex chains for the pmOS phone (GAPK bridge, echo cancellation).
- **PipeWire** kept **for GNOME screencast only**; `pipewire-pulse` purged and masked.
- Sources: `iso_modules/82-services.sh`, `scripts/pulse-gsm-setup.sh`, `scripts/audio-chain.sh` (slimmed-down PulseAudio on every start).

2.36.1.6.10 5.10 Diagnostics

`bash ./checks/check_all.sh` runs everything and gives a verdict (log in `/tmp`): `global_check.sh` (STP/HLR/MSC/BSC per operator), `ss7_check.sh` (inter-STP matrix), `diag-stp-operator.sh`, `diag-interstp.sh`, `wan_ss7_check.sh`, `annuaire.sh` (subscribers), `operator_summary.sh`, `vty-debug-dump.sh`. `./ss7-console.py`: navigable SS7 diagram with built-in VTYs. `./start-native.sh --status` tells what is running.

2.36.1.6.11 5.11 Ports, VTY, network

Port	Service	Port	Service
4239	VTY osmo-stp	8080	bench dashboard
4242	VTY osmo-bsc	8081	I/Q spectra (fft-web)
4243	VTY osmo-mgw	9090	Prometheus Open5GS
4245	VTY osmo-sgsn	9999	WebUI Open5GS (HSS)
4254	VTY osmo-msc	5060	Asterisk SIP
4258	VTY osmo-hlr	2775	SMPP
4260	VTY osmo-ggsn	7890	inter-op SMS relay
4247 / 4248	VTY mobile 1 / 2 (gr-gsm)	29118	SGs (MSC ↔ MME)
4347	VTY mobile (DSP bench)	2905 / 2908	M3UA local / inter-STP
4729	GSMTAP (Wireshark)	5700-5702	TRXD (bridge ↔ osmo-bts-trx)
4730 / 4731	decoded SI / SCH	6702	DL bursts to the DSP
1234	gdb QEMU	2000 / 2001	ZeroMQ srsENB ↔ srsUE

Network: docker interconnect `gsm-inter` 172.20.0.0/24 and `gsm-net-opN` 172.20.N.0/24; node private IP `192.168.<N+1>.10`; LTE `ogstun` 10.45.0.0/16. Numbering plan: `100101`/`100102` (the two subscribers), `N0001`…`N9999` (operator N in multi), `100`/`200` (Linphone), `600` (echo), `9XXXXX` (inter-op outgoing).

2.36.1.6.12 5.12 What was removed, and why

Removed	Reason	Source
avahi (daemon, utils, autoipd, libnss-mdns)	no mDNS on a bench; purged and masked	`80-chroot.sh`
Chromium	Firefox is the only browser, starts as root with no wrapper	`86-finitions.sh`
sssd	masked, not uninstalled (no domain): `systemctl unmask sssd` restores it	`86-finitions.sh`
pipewire-pulse	system-wide PulseAudio instead; pipewire kept for screencast	`80-chroot.sh`
`osmo-dsp` icon	DSP is the default, right-click offers gr-gsm	`85-installeur-bureau.sh`
`osmo-trust-desktop` and the desktop icons	launchers are in the dock and the menu	`85-installeur-bureau.sh`

2.36.1.6.13 5.13 The Calypso bench, the C123 window and GLava

The Calypso chains (`/opt/GSM/c54x_exe/run.sh`, `MODE=dsp`): each chain is an *instance* of four processes — `c54x_exe` (the DSP), `qosmo` (the ARM), osmocon, and the host application. The network itself (core, BSC, BTS) stays with `./start-native.sh`; these commands only add phones to it.

Command	Instance	What runs	Logs
`banc-trx`	1	firmware `trx.highram.elf` + host `transceiver`: the Calypso **is the BTS#0 radio** (TRXD 5700-5702)	`/tmp/c54x-pont`
`banc-mobile`	2	firmware layer1 + `mobile`, camped on instance 1 **over the air** (`pont/air.py`)	`/tmp/c54x-pont2`
`banc-mobile-seul`	1	the classic set-up: one mobile + `pont/pont_dsp.py` towards osmo-bts-trx	`/tmp/c54x-pont`
`banc-rssi`	4	firmware `rssi.highram.elf`: the RSSI app alone, screen and keys emulated; kept apart (`banc-stop` and `start-native.sh --stop` leave it running)	`/tmp/c54x-pont4`
`banc-stop` / `banc-status` / `banc-logs`		`run.sh --stop` / `--status` / `--logs`	
`banc-mobile-stop`/`-status`/`-logs`, `banc-rssi-stop`	2, 4	the same for one instance	

In classic-BTS mode (`pont_dsp.py` running) `banc-rssi` takes its downlink from the bridge's copy (UDP 6762); with `--bts`, from the air. Realistic air levels are on by default (`CALYPSO_AIR_NIVEAUX=1`): the cell is heard only on its ARFCN (`PLAN_ARFCN` in `/etc/osmocom/radio-plan.env`), noise elsewhere.

The same set-ups in one go, network included:

```
./start-native.sh --bts           # the Calypso is the BTS radio, alone
./start-native.sh --bts --ms      # + a mobile camped on it over the air
./start-native.sh --bts-classic   # osmo-bts-trx + bridge + emulated mobile (the --dsp default)
```

These names are defined once, in `scripts/osmo-banc-aliases.sh`, installed as `/etc/profile.d/98-osmo-banc-aliases.sh` for login shells. They are **also real commands** in `/usr/local/bin`, links to `tools/osmo-banc-cmd.sh`, which sources that file and replays its own name: they work from Alt+F2, `sudo`, a script or `bash -c` (`tools/osmo-banc-cmd.sh --install` lays the links; `70-scripts.sh` does it in the ISO).

The C123 window (`tools/osmo-c123-gtk.py`, GTK3, system `python3` with `python3-gi`, `python3-gi-cairo`, `gir1.2-gtk-3.0`): a photo of the Motorola C123 (`data/c123.jpg`) with the ST7558 LCD that `qosmo` publishes in `/dev/shm/calypso[N]/calypso_lcd` redrawn into it, and keys you can click — or type: `0-9`, `*`/`s`, `#`/`h`, arrows, Enter = menu, `o` = ok, F1/F2 = soft keys. Power (red key, Esc, `p`) needs a **long press** and is never sent to a TRX instance: it would switch the BTS radio off.

Command	Window on
`c123`	the first live instance that is **not** a TRX (the mobile); in a login shell, the alias `c123` opens instance 1
`c123-1`, `c123-2`, `c123-3`	instances 2 (mobile), 3 (second TRX phone), 4 (rssi)
`c123-tous`	one window per live instance, plus those that appear later

Command	Window on
`c123 --modele c139`	the colour C139 screen, for the Magnetite firmware (add-on)
`c123 --calibrate`	redefine the key zones (`data/c123-zones.json`) with the mouse

GLava (`tools/glava-build.sh`, `iso_modules/85-glava.sh`, desktop ISO only): GLava with the *NCS Spectrum* module, compiled at build time (no noble package). A spectrum circle sits above the city of the wallpaper and a bar spectrum runs along the bottom edge, both sized from the screen at login (`tools/osmo-glava.sh`). They listen to **the speakers and the microphone together**: a null sink `osmo_glava` fed by two loopbacks, the monitor of the hardware output (everything that plays, bench voice included) and the default source (`osmo_mic_ec`, the echo-cancelled mic). The mic goes only to that sink, never to the speakers: no feedback. A systemd user service: `systemctl --user stop|start|disable glava`. It needs OpenGL 4.2.

2.36.1.7 6. Installing to a disk

2.36.1.7.1 6.1 Starting the installer

The dock's **"Installer"** icon, or `osmo-install` (`/usr/local/bin`, written by `85-installeur-bureau.sh`). It re-elevates to root via `pkexec`, finds the squashfs whatever the boot mode (medium, `toram`, loop) and mounts it at a stable path (`/run/osmo-install-src/live/filesystem.squashfs`), unmounts a target left mounted by a previous attempt, regenerates the apt mirror page ranked by speed (`packaging/apt-mirror.sh --list`), then opens Calamares with the repository's configuration (`installer/calamares/`, copied to `/etc/calamares` at build time).

2.36.1.7.2 6.2 The screens (`installer/calamares/settings.conf`)

Welcome (20 GB, 2 GB RAM, root required) → Location (Europe/Paris) → Keyboard (pc105) → Partitions → Users → Graphics drivers → Ubuntu mirror → Summary → Install → Finish. The slideshow during the copy reproduces the image's spec sheet (`branding/osmo/show.qml`).

2.36.1.7.3 6.3 Partitioning (`modules/partition.conf`)

- **GPT** table, ext4, 300 MiB ESP on `/boot/efi`, no swap by default (`swapfile` optional).
- Automatic layout: **"osmo-egprs" on `/`** (50 %, 25 GiB minimum, **never encrypted**) and **"crypthome" on `/home`** (the rest, 5 GiB minimum).
- **Ticking "encrypt" encrypts `/home` only**, with LUKS2: the root stays readable, secrets (profiles, keyrings, cookies) live on the encrypted volume. Manual partitioning allowed.

2.36.1.7.4 6.4 Accounts and drivers

- Users: the created account is a sudoer (sudo, adm, dialout, audio, video, pulse-access, docker…), with GDM autologin; the root password is asked for (`setRootPassword`), no complexity requirement; the name `root` is forbidden.
- Graphics drivers: nouveau (free) by default; option **NVIDIA 610 proprietary**, installed in the target only if `lspci` sees a 10de card and the network responds (`apt-get install nvidia-driver-610`, then `update-initramfs`).
- Mirror: the live stick's one or a measured mirror.

2.36.1.7.5 6.5 What the installer does in the target (`modules/shellprocess-osmo.conf`)

1. Unlocks root (`passwd -u root`).
2. Deletes the live stick's `osmocom` account (unless it is the chosen name).
3. `crypthome-postinstall <user>`: wires up the encrypted `/home` (see 6.7).
4. Removes the root autologins on tty1 and ttyS0.
5. GDM: `AutomaticLogin=<user>`; the PAM rule refusing root at the login screen is restored.
6. Copies the launchers from `/etc/skel` to the user's desktop.
7. Installs and enables `osmo-egprs-web.service`.
8. Removes the `osmo-install` and `calamares` icons from the menu.
9. Generates `~/.config/pmbootstrap_v3.cfg` for the user.
10. Purges `live-boot`, rebuilds the initramfs.

GRUB: `/EFI/ubuntu` (prefix compiled into the signed GRUB), `GRUB_TIMEOUT 5`, os-prober enabled, `GRUB_ENABLE_CRYPTODISK=y`, kernel line inherited from the stick (`quiet splash`). fstab: `noatime`, `discard` on SSD.

2.36.1.7.6 6.6 After installation: accounts and services

	Live stick	Installed disk
GDM session	root, automatic	the created user, automatic
root	pw `osmo`	pw chosen at install, unlocked
`osmocom`	sudoer, pw `osmo`	deleted
`osmo-egprs-web`	active	active
`osmo-banc`, `osmo-multi`, `osmo-lte`	installed, not enabled	installed, not enabled

Enable a bench at every boot: `systemctl enable --now osmo-banc` (or `osmo-multi`, `osmo-lte`). At build time: `--banc`, `--multi`, `OSMO_ISO_LTE=1`. On disk for multi: `OSMO_MULTI_ENABLE=1 ./addition.sh`.

2.36.1.7.7 6.7 Living with the encrypted `/home` (`configs/crypthome/`)

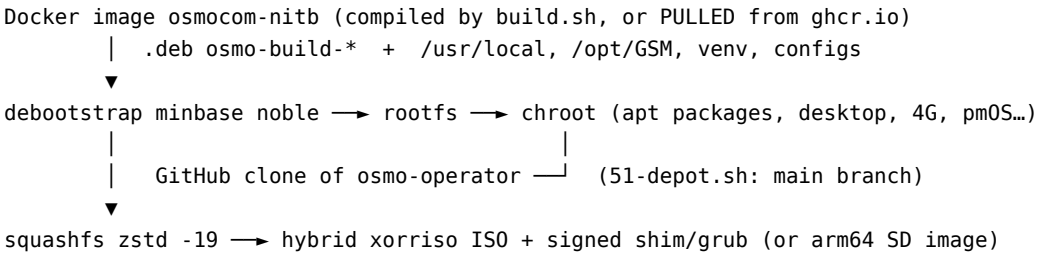
- **At boot**, `unlock-home.service` asks a single question, on the text console, after leaving Plymouth: `decrypt /home -passphrase- or Enter`. The passphrase (echoed as *) opens the volume and the user's session. **Empty Enter, 60 s timeout, three rejected passphrases, missing disk or failed mount: the machine boots anyway**, into an **ephemeral root** session (GDM switches to root for this boot only; `/root` is covered by an overlay whose writes go to a tmpfs). Boot is never blocked.
- `CRYPTHOME_PROMPT=plymouth` in `/etc/default/crypthome` puts the question back in the Plymouth theme (only for machines where its keyboard input works; on an Intel + NVIDIA laptop it took no keystrokes).
- Afterwards: `unlock-home` / `lock-home` in a terminal; apps holding secrets (Firefox, Thunderbird, KeePassXC…) run as the volume owner via `run-as-owner` when it is open, as root otherwise (`refresh-owner-apps` redirects the launchers).
- Second disk added later: `/root/init-crypthome.sh` (LUKS2, `rsync` copy, crypttab/fstab with `noauto`, service enabled, the old `/home` set aside in `/var/backups/home-avant-chiffrement-*` at the next reboot).

2.36.1.7.8 6.8 Updating an installed machine

- `osmo-update` (at boot, or by hand): fast-forward `git fetch` of `/opt/GSM/osmo-operator` and the embedded repositories. No packages.
- What the build installs **outside** the repository (units in `/etc/systemd/system`, `unlock-home` in `/usr/local/sbin`, `osmo-*-anim` launchers) is reinstalled by hand from `services/` and `configs/crypthome/`, or by reinstalling.
- `addition.sh` and `update.sh` rewrite the `.desktop` files they own.

2.36.1.8 7. Building the ISO

2.36.1.8.1 7.1 Where the pieces come from



- **The Docker image is not rebuilt by default** (amd64). Pull order: `ghcr.io/<repo>/osmocom-nitb:base-<fingerprint>` (the fingerprint is the sha256 of the tracked files in Dockerfile, packaging, configs, scripts, services, helpers, opt-gsm, tools, patches: when it matches, the published image was built from exactly this tree), then the repo's `:latest`, then the Docker Hub image. `build.sh` runs only if none responds, or with `--build-docker`.
- **The repository burned into the ISO is a clone of the `main` branch on GitHub**, not the local tree: push before burning. To burn the local tree, `OSMO_EGPRS_REPO=/chemin/du/depot OSMO_EGPRS_BRANCH=<branche>`.
- **The `iso_modules/` modules and `build-iso.sh`, however, are read from the local tree** at build time.

2.36.1.8.2 7.2 Host requirements (`iso_modules/20-hote.sh`)

Ubuntu 24.04 as root, docker (installed if missing), and the packages the module installs: debootstrap, git, rsync, zstd, xz, cpio, gcc, make, python3-pil, python3-yaml, kpartx, dosfstools, mtools; on amd64 squashfs-tools, xorriso, grub-pc-bin, grub-efi-amd64-bin and -signed, shim-signed, isolinux, syslinux-utils; on arm64 qemu-user-static, binfmt-support, docker-buildx. `OSMO_ISO_HOST_READY=1` skips this step. No disk space check is done: leave plenty of room in `/var/tmp` (the work directory is `/var/tmp/iso-build-<pid>`, never `/tmp`, often a tmpfs: rootfs, squashfs and ISO must fit there together) and for the docker store (base image ~11 GB).

2.36.1.8.3 7.3 The command

```
sudo ./build-iso.sh --desktop                # the published image
sudo ./build-iso.sh                          # the four amd64 images
sudo ./build-iso.sh --role=operator --node=2  # the operator for node 2
sudo ./build-iso.sh --arm --lite             # Raspberry Pi 4 SD image
```

Option	Variable	Effect
`--role=operator|interstp`	`ISO_ROLE`	bench or SS7 hub
`--node=N`	`ISO_NODE`, `ISO_WAN_ID`	node identity 1-9, name `osmo-operator-N`
`--lite`, `--desktop`, `--all`	`ISO_LITE`, `ISO_DESKTOP`, `ISO_ALL`	see §2
`--arm` / `--arch=arm64`	`ISO_ARCH`	ARM SD images through `armbian/build-armbian.sh` (§2); `OSMO_ISO_ARM_DRY=1` prints the `compile.sh` commands only
`--board=LIST`	`ISO_ARM_BOARD`, `OSMO_ISO_BOARD`	`--arm` boards: `rpi4`, `opi5`, `all` (default) or Armbian names
`--banc`, `--multi`	`OSMO_ISO_BANC`, `OSMO_ISO_MULTI`	enable the unit at the image's boot
`--version=24.04|22.04`	`ISO_UBUNTU`, `ISO_SUITE`	noble (default) or jammy
`--kb=fr`	`OSMO_ISO_KB`	keyboard layout (asked in the terminal otherwise)
`--output=FILE`	`OUTPUT`	output name
`--no-cache`	`NO_CACHE`	full recompilation, `.deb`s rebuilt
`--skip-build[=image:tag]`	`ISO_SKIP_BUILD`, `ISO_PULL_IMAGE`	pull required, no fallback to build.sh
`--build-docker`	`ISO_FORCE_BUILD`	build the image locally
`--without-debs`	`ISO_WITH_DEBS`	no bench `.deb`s in the image
`--wan`, `--wan-nodes=`, `--wan-id=`, `--wan-ops=`, `--hub-ip=`	`ISO_WAN*`, `ISO_HUB_IP`	WAN table burned into `/etc/osmo-wan.conf`

Other variables: `OSMO_ISO_WORK` (work directory), `OSMO_DEB_CACHE` (`/var/cache/osmo-debs`), `OSMO_ISO_LTE=1`, `OSMO_ISO_PMOS_IMAGE` (`/chemin/qemu-amd64.img.zst` or `none`), `OSMO_PMOS_IMAGE_URL`, `OSMO_ISO_LTE_BUILD=0` (no 4G compilation if the `.deb`s are missing), `OSMO_EGPRS_REPO`/`OSMO_EGPRS_BRANCH`, `OSMO_WEB_REPO` (dashboard).

2.36.1.8.4 7.4 The module chain (`iso_modules/`)

All are sourced in the same shell, in numeric order; a module with nothing to do exits via `return` (that is how `--arm` skips a step).

Module	What it does
`00-options.sh`	options, help, suite, mirror, root check
`10-clavier.sh`	keyboard layout (menu if terminal, otherwise `fr`)
`20-hote.sh`	host packages, docker, tools, arm64 binfmt
`21-docker-build.sh`	pull or build of the Docker image, child passes of `--all`, output names
`22-wan-table.sh`	WAN table, `trap cleanup`, tool checks
`30-image-configs.sh`	Osmocom configs generated from templates (`start-docker.sh`), native patches
`31-image-source.sh`	source image, suite check, injection of `/etc/osmocom` and `/etc/asterisk`
`40-rootfs.sh`	`debootstrap --variant=minbase` (cache `~/.cache/osmo-iso-debs`), or inherited rootfs
`50-injection-image.sh`	build `.deb`s via `dpkg -i`, copy of `/usr/local`, `/opt/GSM`, venv, node, units
`51-depot.sh`	clone of osmo-operator and the tests, `coeur.env`
`52-qemu.sh`	qosmo, c54x_exe, grgsm_exe, DSP ROMs, firmware, qemu-system-arm, `qosmo` launcher, toast/untoast
`60-dashboard.sh`	osmo-egprs-web (native, no docker), tutorial, unit enabled
`70-scripts.sh`	`/usr/local/bin` links (`c123`, the `banc-*` commands), `98-osmo-banc-aliases.sh`, `osmo-wan.conf`, NetworkManager
`71-debs-banc.sh`	`packaging/build-debs.sh` → rootfs `/var/cache/osmo-debs`
`80-chroot.sh`	apt repos (NodeSource, MongoDB, Armbian), kernel, packages, venv, desktop, Firefox, GDM, gsettings, `update-initramfs`
`81-cloture-systeme.sh`	`ldd` closure from the image, no-wait network, DNS, `osmo-ip-plan`, `osmo-qemu-link`, `osmo-update`, SMS animation
`82-arm-natif.sh`	former `--arm` path (native compilation in a debootstrap arm64 chroot); no longer reached: `00-options.sh` hands `--arm` over to `armbian/build-armbian.sh`
`82-services.sh`	role marker, os-release, Asterisk, audio (system-wide PulseAudio, `asound.conf`), units installed not enabled, Open5GS WebUI
`83-espace-ram.sh`	fstab, `/tmp` and `/dev/shm` as % of RAM, journal, logrotate, ring-buffer tcpdump, ssh, purge
`84-comptes.sh`	root and osmocom, console and GDM autologin, Linphone account
`85-installeur-bureau.sh`	Calamares (`/etc/calamares`), `osmo-install`, icons, GTK launchers (`osmo-*-anim`), Info page
`86-finitions.sh`	fixed live-boot `toram`, sssd masked, Chromium removed, Firefox microphone, screenfetch banner, Plymouth `osmo-bts`
`85-glava.sh`	`--desktop`: GLava + NCS Spectrum compiled into the rootfs, `glava.service` (user)
`87-lite.sh`	`--lite` pruning of `/opt/GSM`

Module	What it does
88-lte-pmos.sh	srsRAN + Open5GS (from the .debs, otherwise compiled), freeDiameter, pmbootstrap, PPP kernel, VM image, final 2G + 4G + phone check
89-home-chiffre.sh	crypthome tools, unlock-home.service, root keyring on the encrypted volume, launchers under run-as-owner
90-iso.sh	dpkg check, squashfs zstd -19, kernel, GRUB menu, /.disk/info, squashfs read-back
91-secure-boot.sh	xorriso ISO with signed shim and GRUB (or unsigned grub-mkrescue)
92-rpi-image.sh	former --arm SD image assembly (Pi 4 only); no longer reached
99-fin.sh	rootfs handed to the next pass (--all), summary

2.36.1.8.5 7.5 CI (.github/workflows/build-iso.yml)

- Triggers: push of a v* or iso-* tag, or manual dispatch with the inputs release, no_cache, build_docker, mode (all, stp, base, lite, desktop) and pmos_image (embeds the postmarketOS VM, +1.7 GB per ISO).
- A single job on ubuntu-24.04 (6 h max): runner cleanup, docker moved to /mnt, .deb cache keyed by fingerprint (osmo-debs-noble-g2-<empreinte>). If ghcr.io/…/osmocom-nitb:base-<empreinte> exists, it is pulled (--skip-build=REF); otherwise --build-docker.
- ARM: .github/workflows/build-arm.yml — the arm64 Docker image from docker-arm.yml, then **one job per board** (matrix, native ubuntu-24.04-arm runners) running build-iso.sh --arm --board=<board> --skip-build=<ghcr image> [--desktop], and a release job gathering every board. Inputs boards (default all) and desktop (default on).
- Outputs: SHA256SUMS, MD5SUMS, files over 2 GiB split into *.iso.part-NN (1,900 MB) with PARTS.sha256, 14-day artifact, GitHub release (softprops/action-gh-release) with the notes from .github/release-notes.md, pre-release except on a v* tag.

2.36.1.8.6 7.6 Durations and sizes

Desktop build, Docker image pulled	~2 h 15 on 16 cores
Full Docker compilation (--build-docker)	1 h 23 in CI (wiki/Build.md states 15-20 min on a reference machine)
Desktop ISO	6.5 GB (4 parts in the release)
Desktop vs operator	+2.5 GB
Base Docker image	~11 GB
--arm first time on x86	about ten hours (arm64 Docker image under emulation), then one Armbian build per board

---

2.36.1.9 8. Files to know

File	Role
build-iso.sh, iso_modules/	the build chain
installer/calamares/	the installer (modules, branding, slideshow)
configs/crypthome/	encrypted /home: unlock-home, crypthome-postinstall, init-crypthome.sh, run-as-owner
configs/plymouth/osmo-bts/, tools/plymouth-render.py	the boot theme, generated
services/*.service	osmo-banc, osmo-multi, osmo-lte, osmo-egprs-web, open5gs-webui
data/desktop/*.desktop	the dock icons; tools/osmo-gtk-run.py the windows
/etc/default/osmo-banc, /etc/default/osmo-multi, /etc/default/osmo-lte	persistent service options

File	Role
`/etc/default/crypthome`	UUID, owner, timeout, `CRYPTHOME_PROMPT`
`/etc/osmocom/coeur.env`, `/etc/osmo-wan.conf`, `/etc/osmo-role`	node, WAN, image role
`/var/log/osmocom/`, `journalctl -u osmo-banc`	bench logs
`addition.sh`	the add-ons (docker/multi, OpenCL/deka, Claude, extras)
`checks/check_all.sh`, `ss7-console.py`	diagnostics

2.37 /opt/GSM/osmo-operator/wiki/Installer-c54x_exe.md

10823 octets, 194 lignes

2.37.1 Installing c54x_exe (the Calypso DSP, TI mask ROM) and the DSP bridge

> Version of 2026-10-03. Back: Telephone-emule. Prerequisite: qosmo. The repository ships its own installer, `c54x_exe/install.sh`: it is also what the `Dockerfile` (stage `l1`), `Dockerfile.run`, `start-docker.sh`, the ISO and `install.sh --telephone` call. The manual commands below are **what the script does**, step by step. `Dockerfile:N` numbers refer to the Dockerfile at commit `4266c8b`, back when it still carried these commands.

`c54x_exe` (`https://github.com/bbaranoff/c54x_exe`) runs the **original mask ROM of the TMS320C54x** found in Calypso basebands, as a Linux process. In the DSP setup, the osmocom-bb firmware under QEMU talks to it through the real shared API RAM, frame by frame (`c54x_exe/README.md:1-9`). It does not contain the C54x core: it **compiles qosmo's sources** (`Makefile:1-7`, `README.md:57-61`).

2.37.1.1 With the installer

```
git clone https://github.com/bbaranoff/c54x_exe /opt/GSM/c54x_exe
cd /opt/GSM/c54x_exe
./install.sh --check      # qosmo cloned? libosmocoding seen by pkg-config? ROM? — changes NOTHING
sudo ./install.sh         # make, ROM into /opt/GSM and rom/, then c54x_exe --trames 200
```

step	does	Dockerfile line	already done if
`build`	`make QOSMO=…` (with `--portable`: `CFLAGS` without `-march=native`); `ldd` check	733-735	— (the Makefile recompiles on every call, `Makefile:46-52`)
`rom`	copies an **already verified** ROM found on the machine, otherwise `rom/fetch-rom.sh --dest …` (download, conversion, checksum)	736	all 6 ROMs + `Registers` present and matching `rom/SHA256SUMS.3606` in every destination
`verify`	`c54x_exe --rom-dir … --trames 200`: « N sections chargees » (“N sections loaded”) then « bilan sur 200 trames » (“summary over 200 frames”)	—	—

Options (and variable): `--qosmo DIR` (`QOSMO`, default `$GSM_ROOT/qosmo`), `--rom-dir DIR` repeatable (`ROM_DIR`, default `$GSM_ROOT` **and** the repo's `rom/`, like `Dockerfile:736`), `--portable` (`PORTABLE=1`, the `CFLAGS` from `Dockerfile:735`), `--dest DIR` (copies the repo without `.git` or binaries into `DIR` and builds there: **the binary in service is not overwritten**), `--only`, `--skip`, `--reinstall`, `--with-deps`, `-v`. Logs: `/tmp/c54x_exe-install/<step>.log`.

Without root: `ROM_DIR=$HOME/calypso-rom ./install.sh`, then run `c54x_exe --rom-dir $HOME/calypso-rom` (the binary's default is `/opt/GSM`, `src/main.c:79`).

The `--trames 200` check runs without `--arm`: **private** API RAM, no `/dev/shm`, no socket, no `/tmp/c54x-pont` (verified in a private mount namespace) — it does not disturb a running bench.

Typical failure, spelled out:

```
[FAIL] Compilation de c54x_exe (sources de qosmo) (manque : libosmocoding + libosmocore vus de pkg-config)
     → export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig  (libosmocore est en /usr/local, Dockerfile:92-94)
```

("manque" = "missing": libosmocore lives in `/usr/local`.)

2.37.1.2 What the script does, by hand

2.37.1.2.1 Prerequisites

- qosmo cloned (default `/opt/GSM/qosmo`) — Installer-qosmo.
- gcc, make, pkg-config, **libosmocore + libosmocoding** visible to pkg-config (`Makefile:14`). Check before compiling (the Makefile hides the error, `2>/dev/null`):

  ```
  pkg-config --cflags --libs libosmocoding libosmocore
  ```

  If empty: `export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig` (the Dockerfile's value, `Dockerfile:92-94`).
- python3 + curl or wget for the ROM (`rom/fetch-rom.sh:47-52`).
- For the bridge: python3, `python3-numpy` (`pont/record.py:6`), `libosmocoding.so` and `libosmogsm.so` loaded through ctypes (`pont/gsm.py:50,59`) — so `ldconfig` must be up to date if libosmocore is in `/usr/local`.

2.37.1.2.2 Build (`build` step)

```
git clone https://github.com/bbaranoff/c54x_exe /opt/GSM/c54x_exe
cd /opt/GSM/c54x_exe
make                                  # or: make QOSMO=/path/to/qosmo   (README.md:63-68)
```

For a binary that must run on **a different CPU** than the build machine, drop `-march=native` as the Dockerfile does (`Dockerfile:726-735`):

```
make QOSMO=/opt/GSM/qosmo \
  CFLAGS="-O3 -g -Wall -Werror=format -Werror=format-extra-args -Wno-unused-function -Wno-unused-variable
↪  -Wno-unused-but-set-variable -Wno-sign-compare"
```

`make` recompiles **on every call** (a single cc command, no `.o`): this is intentional, the sources come from another repository (`Makefile:42-52`).

2.37.1.2.3 The DSP ROM (`rom` step)

Not in the repository. `rom/fetch-rom.sh` downloads the FreeCalypso 3606 dump, converts it (`tools/dsp_txt2bin.py`) and checks `rom/SHA256SUMS.3606` (`README.md:113-119`). The binary reads it from `--rom-dir`, default **`/opt/GSM`** (`src/main.c:79,105`). Dockerfile command (`:736`):

```
bash rom/fetch-rom.sh --dest /opt/GSM --dest /opt/GSM/c54x_exe/rom
```

Expected (`fetch-rom.sh:46,57,65`):

```
telechargement : ftp://ftp.freecalypso.org/pub/GSM/Calypso/dsp-rom-3606-dump.txt
somme de controle verifiee (SHA256SUMS.3606)
ROM DSP 3606 ecrite dans /opt/GSM
ROM DSP 3606 ecrite dans /opt/GSM/c54x_exe/rom
```

(download / checksum verified / ROM written to.) Run again, it says `ROM DSP 3606 deja en place et verifiee` ("already in place and verified"). Options: `--version 3311` (D-Sample), `--force`, mirror via `FREECALYPSO_URL=` (`fetch-rom.sh:2-15,30`). The installer first copies an already verified ROM found on the machine (what `cp c54x_exe/rom/*.bin /opt/GSM/` used to do in `Dockerfile.run`): no network needed if it is there.

2.37.1.2.4 Checking the DSP alone (`verify` step)

```
./c54x_exe --trames 200    # the ROM alone, no ARM: does it boot? (README.md:66)
./c54x_exe --help
```

Optional ISA test (`Makefile:54-61`): `make isa_test && ./isa_test tools/isa_tests.txt 2>/dev/null`.

2.37.1.3 The DSP bridge

The bridge connecting your osmo-bts-trx to the DSP lives in **osmo-operator**, not here (`README.md:82-83`, `run.sh:37-41`). The tree is enough, nothing to compile:

```
git clone https://github.com/bbaranoff/osmo-operator /opt/GSM/osmo-operator
cd /opt/GSM/osmo-operator && python3 pont/pont_dsp.py --help
```

`pont_dsp.py` requires `--dsp-port 6702` (`pont/README.md` §0). It reads the timeslot table from `PONT_BSC_CFG` (default `/etc/osmocom/osmo-bsc.cfg`): point it at **your** BSC config.

2.37.1.4 Running

Against your already started stack (osmo-bts-trx included):

```
cd /opt/GSM/c54x_exe
PONT=1 INSNS=120000 CALYPSO_BSP_STREAM=1 CALYPSO_RHEA_DMA_XFER=1 \
  PONT_BSC_CFG=/etc/osmocom/osmo-bsc.cfg ./run.sh
./run.sh --status | --logs | --stop        # README.md:76-79
```

- `PONT=1`: without it, no bridge, no burst arrives (`run.sh:197-198`).
- `INSNS=120000`, `CALYPSO_BSP_STREAM=1`, `CALYPSO_RHEA_DMA_XFER=1`: what `start-native.sh` sets before handing over to this `run.sh` (block « LE BANC DSP : RÉGLAGES MESURÉS », "DSP bench: measured settings"); the default in `run.sh:49` is 16000, "TCH measured up to 87000 insn/frame". `CALYPSO_RHEA_DMA_XFER`: "without it the API page is never filled: no SB, no BCCH, no SI".
- Other variables: `MODE`, `LOCKSTEP` (1), `VERB`, `IQ`, `AMP`, `QOSMO`, `FIRMWARE_ELF`, `FIRMWARE_BIN`, `OSMOCON`, `MOBILE`, `MOBILE_CFG`, `PONT_PY`, `RUNDIR`, `L2_SOCK`, `C54X_BIN` (`run.sh:19-21`, `README.md:144`).
- Logs: `/tmp/c54x-pont/{dsp,qemu,osmocon,mobile,pont}.log`, sessions archived in `/tmp/c54x-pont/archives/<date>/` (`run.sh:249-270`).

Process-by-process details: Lancer-a-la-main.

All-in-one variant for a **systemd** stack: `./run_real.sh [--secondes N]` (starts `osmo-hlr osmo-stp osmo-msc osmo-mgw osmo-bsc`, restarts `osmo-bts-trx`, runs `run.sh` with `INSNS=60000`, watches for 120 s and prints a verdict). `./run_real.sh --stop`.

2.37.1.5 What proves it works

Line	File	Source
`pont : en attente de l'ARM sur /tmp/calypso_dsp.sock (API RAM : /dev/shm...)`	dsp.log	`src/pont.c:2114`
`pont : RESET #1 (DL_STATUS=...) fn=... pc=0xff80` then `pont : DSP boote (premier IDLE)`	dsp.log	`src/pont.c:1608,1921`, `LAUNCH.md:97`
`[trx] pont DSP : API RAM partagee (...) + socket /tmp/calypso_dsp.sock`	qemu.log	`qosmo/hw/arm/calypso/calypso_trx.c:318`
`[montant] RACH ra=0x.. bsic=..` ; `pont.log` goes from `rach=0` to `rach=N`	dsp.log / pont.log	`LAUNCH.md:66-67`
`[montant] TCH : le firmware poste la tache 13 a fn=..., BSP bascule sur TS2`	dsp.log (during a call)	`LAUNCH.md:68-69`

2.37.1.6 Troubleshooting

Symptom	Cause / fix	Source
link fails on `gsm0503_*`	pkg-config cannot find libosmocoding (error hidden)	`Makefile:14`
`Illegal instruction` at startup	`-march=native` binary from another machine: rebuild here (`./install.sh`), or `--portable` for a binary that travels	`Dockerfile:726-728`, `iso_modules/52-qemu.sh`
`c54x_exe n'a pas ouvert /tmp/calypso_dsp.sock` ("did not open")	ROM missing from `--rom-dir`? `./install.sh --only rom` (or `rom/fetch-rom.sh --dest /opt/GSM`); see dsp.log	`run.sh:104`
rebuild without touching the binary of a running bench	`./install.sh --dest /tmp/c54x-neuf` then compare / replace once stopped	`install.sh --dest`
`somme de controle differente` ("checksum mismatch")	modified dump or wrong version	`fetch-rom.sh:55-56`
`QEMU n'a pas rejoint le DSP` ("QEMU did not join the DSP")	QEMU without `CALYPSO_DSP_EXTERN=1`, or DSP not ready yet	`run.sh:144-145`
`pont.py ne s'est pas annonce` ("pont.py did not announce itself")	inherited `PONT_PY` pointing at `pont.py` (gr-gsm): `env -u PONT_PY`	`run.sh:206`, `start-native.sh banc_dsp()`
`FBSB RESP: result=255` in a loop	no bridge or no BTS: normal without a network	`LAUNCH.md:138`
`DL bursts=0`	no BTS is transmitting to 5700-5702: check the `phy` section of your osmo-bts-trx	`LAUNCH.md:157-158`
SB decoded only one time in three to five	known limitation; `L23_SYNC_RETRIES_SELECTION=8` set by run.sh	`README.md:17-19`, `run.sh:183-189`
`le port VTY 4347 ... deja pris` ("already taken")	another mobile; change `bind 127.0.0.1` in the cfg	`run.sh:177-181`
dirty state after a crash	`./run.sh --stop` purges sockets and `/dev/shm/calypso_*`	`run.sh:210-237`

2.37.1.7 TO BE CONFIRMED

- (fixed on 2026-10-03) `GDB_TELNET_PY` pointed at `/opt/GSM/qosmo-dsp/tools/gdb-telnet.py` (repository gone): `run.sh:48` now uses `$QOSMO/tools/gdb-telnet.py`, and the `telnet 0 44444` console starts with step 2 (`GDB=0` to do without it).
- `mobile_pont.cfg` sends GSMTAP to `172.20.0.1` (the bench's docker gateway): replace with `127.0.0.1` outside a container.
- Which `INSNS` value to keep (16000 / 60000 / 80000 / 120000 depending on the source).

2.38 /opt/GSM/osmo-operator/wiki/Installer-grgsm_exe.md

5902 octets, 105 lignes

2.38.1 Installing grgsm_exe — and the gr-gsm setup

Version of 2026-10-03. Back: Telephone-emule. The repo carries its own installer, `grgsm_exe/install.sh` (also called by the `Dockerfile`, stage `l1`, `Dockerfile.run`, `start-docker.sh`, the ISO and `install.sh --telephone`). The `Dockerfile:N` numbers refer to the Dockerfile at commit `4266c8b`.

2.38.1.1 First: three things carry this name

Name	What it is	Makes the mobile camp?	Source
the **`grgsm_exe`** repo (`https://github.com/bbaranoff/grgsm_exe`)	qosmo's gr-gsm layer 1 (`fl/grgsm_l1.c`, `fl/grgsm.c`) compiled **outside QEMU**, 118 KB, no ARM and no firmware: "the L1 writes into the void"	**no** — a bench and replay tool	`grgsm_exe/README.md:1-42`
the `/usr/local/bin/grgsm_exe` command	wrapper = `python3 pont/pont_dsp.py --no-record --dsp-port 6702`; `PONT_DSP_PORT=0 grgsm_exe = pont/pont.py`	it is the **bridge**	`c54x_exe/LAUNCH.md:142-158`; **outside the repo**, absent from a fresh install
"grgsm_exe bridge" / the `--grgsm` setup	gr-gsm L1 **inside** QEMU (qosmo `--enable-l1-grgsm`) + `pont/pont.py`	**yes** (the historical chain)	`osmo-operator/environment/paths.env:` `start-native.sh:111`

To connect a phone to your network with the gr-gsm setup, you need the **third** one: qosmo + firmware + osmocom-bb (already done in Installer-qosmo) and the `pont/pont.py` bridge from osmo-operator. The `grgsm_exe` repo is optional.

2.38.1.2 1. The grgsm_exe repo (bench tool)

Prerequisites: qosmo cloned (source), gcc. No Osmocom library (`Makefile:12`: `-lpthread -lm -lrt`).

With the installer:

```
git clone https://github.com/bbaranoff/grgsm_exE /opt/GSM/grgsm_exe
cd /opt/GSM/grgsm_exe
./install.sh --check         # qosmo source, cc, make — changes NOTHING
./install.sh                 # make -B, then grgsm_exe --trames 50 run safely
```

step	does	Dockerfile line
`build`	`make -B QOSMO=…` — `-B` because the binary is **tracked by git**: on clone it is newer than qosmo's sources and `make` alone concluded "à jour" (up to date) (`Dockerfile.run:189` did `clean all`)	745-746
`verify`	`grgsm_exe --trames 50` → "bilan sur 50 trames" (report over 50 frames)	—

The check does not disturb a running bench: grgsm_exe takes UDP 4730/4731 with `SO_REUSEADDR` and opens `/dev/shm/calypso_*` with `O_CREAT` + `ftruncate` (`calypso_l1_grgsm.c`); as root it therefore runs in **private** mount and network namespaces (`unshare`, a fresh `/dev/shm`); otherwise, if a bench is visible, the run is skipped and says so. Options: `--qosmo DIR` (`QOSMO`), `--dest DIR` (build a copy), `--only`, `--skip`, `-v`; logs in `/tmp/grgsm_exe-install/`.

By hand (what the script does):

```
make -B                     # QOSMO=/path/to/qosmo make -B   (README.md:41-42)
./grgsm_exe --trames 5000   # README.md:6-10
./grgsm_exe --verbeux
```

Check that it opens its inputs (`README.md:21-27`):

```
$ ss -ulnp | grep 473
127.0.0.1:4730   users:(("grgsm_exe",...))    # GSMTAP
127.0.0.1:4731   users:(("grgsm_exe",...))    # SCH
```

With nothing publishing on 4731 or in `/dev/shm`, `si_valid` stays false and the report says so (`README.md:38-39`). **Do not run it at the same time as QEMU in the gr-gsm setup**: both take UDP 4730/4731 (deduced from `README.md:24-27` and `LAUNCH.md:87-88`).

2.38.1.3 2. The gr-gsm setup (full phone)

Nothing more to compile than for qosmo: the `qemu-system-arm` binary built with `--enable-l1-grgsm` **is** layer 1. The bridge is `pont/pont.py` (python3, numpy, libosmocoding — no GNU Radio, `pont/gsm.py:50,59`).

Launch via the c54x_exe repo's script (it handles both setups, `run.sh:4-13`):

```
cd /opt/GSM/c54x_exe
MODE=grgsm PONT=1 PONT_BSC_CFG=/etc/osmocom/osmo-bsc.cfg ./run.sh
```

In the grgsm setup, `run.sh` skips step 1 (no c54x_exe), launches QEMU **without** `CALYPSO_DSP_EXTERN`, waits for `backend gr-gsm` in qemu.log (`run.sh:146-147`), then osmocon, mobile, and `pont/pont.py` (`run.sh:37-38`).

Expected:

Line	File	Source
`[l1] backend gr-gsm : GSMTAP udp/4730, SCH udp/4731`	qemu.log	`calypso_l1_grgsm.c:1207`, `LAUNCH.md:95`
`pont TRX : ports 5700/5701/5702, ARFCN 514, BSIC 7` then `STATS fn=… | DL bursts=N`	pont.log	`LAUNCH.md:157-158`

Watch out for the mobile config: `run.sh` uses `mobile_pont.cfg` in both setups (`run.sh:33`), and that file imposes `io-tch-format ti` (DSP frames). On gr-gsm, it is the bridge that encodes speech in **rtp** format; with `ti` "les trames passent […] et le son sort robotisé" (the frames go through […] and the sound

comes out robotic) (`mobile_pont.cfg`, the `io-tch-format` comment). For the gr-gsm setup, use the bench config `osmo-operator/configs/mobile.cfg` (copied to `/root/.osmocom/bb/mobile.cfg`, `Dockerfile:1170`) via `MOBILE_CFG=` — **TO CONFIRM**: this config contains `__…__` templates filled in by `generate_configs.sh`; it must then be filled in by hand (IMSI, Ki, `stick`, VTY).

2.38.1.4 Troubleshooting

Symptom	Cause / fix	Source
`la couche 1 gr-gsm ne s'est pas annoncee` (the gr-gsm layer 1 did not announce itself)	QEMU compiled without `--enable-l1-grgsm`, or launched with `CALYPSO_DSP_EXTERN=1`	`run.sh:147`, `Dockerfile:698`
`pont.py est le pont du montage grgsm : pour le DSP, lancer pont/pont_dsp.py` (pont.py is the grgsm-setup bridge: for the DSP, launch pont/pont_dsp.py)	`--dsp-port` passed to pont.py	`pont/pont.py:25-26`
`grgsm_exe` command not found	the wrapper is versioned nowhere (`LAUNCH.md:28`); launch `python3 pont/pont.py` directly	—
79% CRC KO, LU failing	bridge from outside the repo (an old copy); always launch the one from the osmo-operator tree	`start-native.sh`, block “LE PONT VIENT DU DEPOT” (the bridge comes from the repo)

2.39 /opt/GSM/osmo-operator/wiki/Installer-qosmo.md

10509 octets, 200 lignes

2.39.1 Installing qosmo (QEMU Calypso) — and the osmocom-bb firmware

> Version of 2026-10-03. Back: Telephone-emule. The repository ships its own installer, `qosmo/install.sh`: it is also what the `Dockerfile` (stage `qemu`), `Dockerfile.run`, `start-docker.sh` and `install.sh --telephone` call. The manual commands below are **what the script does**, step by step. `Dockerfile:N` numbers refer to the Dockerfile at commit `4266c8b`, back when it still carried these commands (it now calls the installer).

qosmo (`https://github.com/bbaranoff/qosmO`) is a QEMU fork that adds the `calypso` machine: the TI Calypso SoC (ARM946) on which the **original, unpatched osmocom-bb firmware** boots (`qosmo/README.md:1-5`). The same binary serves both setups: built with `--enable-l1-grgsm`, it embeds the gr-gsm layer 1; started with `CALYPSO_DSP_EXTERN=1`, it cuts it out and delegates to the external DSP `c54x_exe` (`osmo-operator/Dockerfile:679-682`).

It also contains the **sources** that `c54x_exe` and `grgsm_exe` compile (`hw/arm/calypso/l1-dsp/`, `l1-grgsm/`, `contrib/hors-qemu/`): install it first.

2.39.1.1 With the installer

```
git clone https://github.com/bbaranoff/qosmO /opt/GSM/qosmo
cd /opt/GSM/qosmo
./install.sh --check          # prerequisites and state of each step — changes NOTHING
sudo ./install.sh --deps      # missing apt packages (the list: ./install.sh --print-deps)
sudo ./install.sh             # venv, configure, make, make install, /usr/local/bin, check
```

Steps (`./install.sh --list`), each with an “already done?” test, its prerequisites, the work and an after-the-fact check — the contract of `osmo-operator/install_modules/_lib/inst.sh`:

step	does	Dockerfile line	already done if
`venv`	`python3 -m venv ~/.venv-qemu` + `pip install tomli`	694-696	`tomli` importable in the venv
`configure`	`mkdir -p build && cd build && ../configure --target-list=arm-softmmu --enable-l1-grgsm --prefix=… --disable-werror --disable-docs`	697-699	`build/config.log` has **the same** options
`build`	`make -j$(nproc)` in `build/` (incremental)	700	— (always rerun, only rebuilds what changed)
`install`	`make install` (prefix `/opt/GSM/qemu-install`)	701	installed binary identical to `build/`
`bin`	copies `qemu-system-arm` and `qosmo` into `/usr/local/bin`	702-703	identical copies; *not applicable* if the directory is not writable
`verify`	`build/qemu-system-arm -M help` must list `calypso` … `(couche 1 : grgsm)`	—	—

Options (and equivalent variable): `--prefix DIR` (`PREFIX`, default `$GSM_ROOT/qemu-install`), `--build-dir DIR` (`BUILD_DIR`, build out of tree — never in the source tree itself: `configure` run from the root wipes a `build/` it created, `configure:16-45`), `--venv DIR|none` (`VENV`, default `~/.venv-qemu`), `--bin-dir DIR|none` (`BIN_DIR`, default `/usr/local/bin`), `--destdir DIR` (install under another root, like the ISO), `--cc gcc-13` (forced compiler: an image's `build/` was configured with gcc-13, `Dockerfile.run`), `-j N`, `--only STEPS`, `--skip STEPS`, `--reinstall`, `--with-deps`, `-v` (output on screen; otherwise one log per step in `/tmp/qosmo-install/`, whose tail is shown on failure). `GSM_ROOT` (default `/opt/GSM`) sets the default prefix.

Without root: `./install.sh` works if you own `/opt/GSM`; the `bin` step is then *not applicable* (binaries left in `$PREFIX/bin`), or use `--prefix ~/qemu-calypso --bin-dir none`.

2.39.1.2 What the script does, by hand

2.39.1.2.1 Prerequisites

Ubuntu 24.04 (the bench base, `Dockerfile:11-15`). Packages (`./install.sh --print-deps`, taken from `Dockerfile:159,166,189-190`):

```
sudo apt-get install -y --no-install-recommends \
  build-essential git pkg-config python3 python3-venv python3-pip \
  libglib2.0-dev libpixman-1-dev libslirp-dev ninja-build socat
```

`socat` is used to query the QEMU monitor (`qosmo/run_modules/40-qemu.sh:91-96`). `./install.sh --check` checks what really matters for `configure` (cc, make, ninja, pkg-config, `glib-2.0` and `pixman-1` as seen by pkg-config, the `venv`/`ensurepip` module): a missing package that is installed some other way is only a note.

2.39.1.2.2 Build

Taken from `osmo-operator/Dockerfile:691-705`:

```
sudo mkdir -p /opt/GSM && sudo chown "$USER" /opt/GSM
git clone https://github.com/bbaranoff/qosmO /opt/GSM/qosmo
cd /opt/GSM/qosmo
python3 -m venv ~/.venv-qemu && . ~/.venv-qemu/bin/activate
pip install --no-cache-dir tomli
mkdir -p build && cd build
../configure --target-list=arm-softmmu --enable-l1-grgsm \
```

```
    --prefix=/opt/GSM/qemu-install --disable-werror --disable-docs
make -j"$(nproc)"
make install
```

The script (`bin` step) then copies the binaries into the PATH (`Dockerfile:702-703`):

```
sudo cp /opt/GSM/qemu-install/bin/qemu-system-arm /usr/local/bin/qemu-system-arm
sudo cp /opt/GSM/qemu-install/bin/qosmo           /usr/local/bin/qosmo
```

> **Keep the tree and `build/`.** The launchers use `/opt/GSM/qosmo/build/qemu-system-arm` (`c54x_exe/run.sh:28`, `qosmo/environnement/bench.env:4`), and c54x_exe/grgsm_exe compile the sources from this tree. The Dockerfile carries the whole tree, build/ included (`Dockerfile:686-688`).

Documented configure variants (`qosmo/README.md:9-13`): no option (bare platform), `--enable-l1-dsp` (C54x **inside** QEMU, not used by the current bench). The bench only uses `--enable-l1-grgsm`.

2.39.1.2.3 Check

```
./install.sh --only verify    # -M help must list « calypso … (couche 1 : grgsm) »
ls -la /opt/GSM/qosmo/build/qemu-system-arm
/opt/GSM/qosmo/build/qemu-system-arm -M help | grep calypso
qosmo -V          # C launcher: « qosmo <version> (<tree>) » (tools/qosmo/qosmo.c, option -V)
```

The machine description says which layer 1 is compiled in (`hw/arm/calypso/calypso_mb.c:111`): `couche 1 : grgsm` ("layer 1: grgsm") expected.

Starting the machine (needs the firmware, next section) — command from `c54x_exe/LAUNCH.md:75-79`, grgsm setup (without `CALYPSO_DSP_EXTERN`):

```
/opt/GSM/qosmo/build/qemu-system-arm -M calypso -cpu arm946 \
  -display none -parallel none -serial pty -serial pty \
  -monitor unix:/tmp/qemu-monitor-pont.sock,server,nowait \
  -kernel /opt/GSM/firmware/board/compal_e88/layer1.highram.elf
```

Expected on stderr (`LAUNCH.md:89-96`):

```
char device redirected to /dev/pts/N (label serial0)
[l1] backend gr-gsm : GSMTAP udp/4730, SCH udp/4731
```

and `printf 'info status\n' | socat - UNIX-CONNECT:/tmp/qemu-monitor-pont.sock` → `VM status: running` (`40-qemu.sh:93-95`). Stop with Ctrl-C.

2.39.1.3 Firmware and osmocom-bb

QEMU runs `layer1.highram.elf`; osmocon loads `layer1.highram.bin` via romload; the mobile speaks L1CTL through osmocon. All three pieces come from osmocom-bb.

Prebuilt firmware (`Dockerfile:585-588`; it is "the only place looked at", `Dockerfile:590-598`) — a clone, nothing to compile. From osmo-operator: `sudo ./install.sh --only firmware` (`install_modules/45-calypso.sh`); by hand:

```
git clone --depth 1 https://github.com/bbaranoff/firmware /opt/GSM/firmware
ls /opt/GSM/firmware/board/compal_e88/layer1.highram.{elf,bin}
```

osmo-gapk, before osmocom-bb (the bench mobile uses `tch-voice io-handler gapk`, `c54x_exe/mobile_pont.cfg`) — `Dockerfile:445-452`:

```
cd /opt/GSM && git clone https://gitea.osmocom.org/osmocom/gapk osmo-gapk
cd osmo-gapk && autoreconf -fi && ./configure --enable-alsa && make -j"$(nproc)" && sudo make install && sudo
 ↪  ldconfig
```

(package `libasound2-dev` required, `Dockerfile:169`.)

osmocom-bb, host tools only (`Dockerfile:548-558`) — requires libosmocore installed:

```
cd /opt/GSM && git clone https://gitea.osmocom.org/phone-side/osmocom-bb
git -C osmocom-bb apply /opt/GSM/osmo-operator/patches/osmocom-bb-clck-gen-frame-tolerance.patch   # optional:
↪  only touches fake_trx (Dockerfile:541-547)
cd osmocom-bb/src && make nofirmware
sudo install -m755 host/layer23/src/mobile/mobile /usr/local/bin/mobile    # Dockerfile:1145-1149
```

`osmocon` stays in the tree: `/opt/GSM/osmocom-bb/src/host/osmocon/osmocon` (path expected by `c54x_exe/run.sh:31` and `qosmo/environnement/bench.env:10`).

Check: `ls /opt/GSM/osmocom-bb/src/host/osmocon/osmocon && mobile --version` (**TO BE CONFIRMED**: mobile's `--version` option).

2.39.1.4 Useful variables

Variable	Default	Role	Source
`CALYPSO_DSP_EXTERN=1`	unset	cuts out the gr-gsm L1, shares the API RAM with `c54x_exe`	`LAUNCH.md:84-86`
`CALYPSO_PONT_LOCKSTEP=1`	set by run.sh if `LOCKSTEP=1`	QEMU waits for the DSP on every frame	`c54x_exe/run.sh:56-57`
`CALYPSO_PONT_RETRY_DIV`	64 (run.sh)	retry rate towards the DSP	`run.sh:130-133`
`CALYPSO_GDB_PORT`	1234	ARM gdbstub (`off` to disable)	`40-qemu.sh:59`

The ~300 `CALYPSO_*` variables are described in `qosmo/hw/arm/calypso/doc/VARIABLES_ENVIRONNEMENT.md` (`qosmo/environnement/README.md:42-45`); the manifest actually applied is printed by QEMU (`grep calypso-manifest qemu.log`).

2.39.1.5 Troubleshooting

Symptom	Cause / fix	Source
CPU at 0, crash at `0x840000`, osmocon loads nothing	missing `-kernel`	`LAUNCH.md:82-83`
`QEMU a démarré puis s'est arrêté` (“QEMU started then stopped”)	invalid machine or firmware; see the tail of `qemu.log`	`40-qemu.sh:102-106`
SDL “parallel0” window grabbing the keyboard	forgot `-display none -parallel none`	`40-qemu.sh:60-67`
`binaire QEMU absent` (“QEMU binary missing”)	build not done; `./install.sh --only build` (or `ninja -C build qemu-system-arm`)	`40-qemu.sh:10-11`
`[FAIL] configure … manque : glib-2.0 vu de pkg-config`	missing `-dev` package: `sudo ./install.sh --deps`	`install.sh --check`
`[SKIP] qemu-system-arm et qosmo dans le PATH (… non inscriptible)`	not root: binaries in `$PREFIX/bin`, or `--bin-dir DIR`	`install.sh`, `bin` step
ninja fails on the very first unit in a `build/` that came from an image	`build/` configured with gcc-13, `cc` = gcc-11: `./install.sh --only build --cc gcc-13`	`Dockerfile.run:171-175`
`/usr/local/bin/qosmo-grgsm` instead of `qosmo`	`make -C tools/qosmo-launch install` fallback in `start-native.sh` (unversioned directory); use `make install` in `build/`	`start-native.sh:692-694`

Symptom	Cause / fix	Source
`osmocon` stuck mid-download	an osmocon killed during romload leaves the UART stub mid-block: restart QEMU, then osmocon	`LAUNCH.md:120-121`

2.40 /opt/GSM/osmo-operator/wiki/Lancer-a-la-main.md

11909 octets, 273 lignes

2.40.1 Launching the emulated phone by hand, process by process

> Version of 2026-10-03. Back: Telephone-emule. Runnable version, network included: lancer-pas-a-pas.ipynb.
>
> Each command is the one **built** by `c54x_exe/run.sh` (DSP setup, called by `osmo-operator/start-native.sh`), with variables **expanded** to the values `start-native.sh` sets in DSP mode. Line numbers cited are as of 2026-10-03. Based on `c54x_exe/LAUNCH.md`.

Convention: one terminal per process (or `&` + redirection as below). Logs go to `/tmp/c54x-pont/`, like `run.sh` (`run.sh:42`).

```
mkdir -p /tmp/c54x-pont
```

2.40.1.1 Common variables (DSP setup)

`start-native.sh` exports them before handing over to `c54x_exe/run.sh` (block « LE BANC DSP : RÉGLAGES MESURÉS, PUIS PASSAGE DE MAIN », i.e. “DSP bench: measured settings, then hand-over”); `run.sh:57` derives `CALYPSO_PONT_LOCKSTEP=1` from them. They are inherited by **all** the following processes:

```
export CALYPSO_BSP_STREAM=1        # one TS0 burst per frame, in FN order
export CALYPSO_RHEA_DMA_XFER=1     # without it: no SB, no BCCH, no SI
export CALYPSO_PONT_LOCKSTEP=1     # QEMU only advances the frame once the DSP is done (run.sh:50-57)
```

2.40.1.2 Step 0 — your network (assumed installed)

The phone launches none of it. Bench order: STP, HLR, MGW, MSC, BSC (`qosmo/run_modules/10…14`), then the BTS (`60-bts.sh`). With systemd, as `c54x_exe/run_real.sh` does:

```
for u in osmo-hlr osmo-stp osmo-msc osmo-mgw osmo-bsc; do systemctl is-active --quiet "$u" || systemctl start
↪  "$u"; done
systemctl restart osmo-bts-trx
```

Without systemd, start the BTS as `qosmo/run_modules/60-bts.sh:49` does:

```
osmo-bts-trx -c /etc/osmocom/osmo-bts-trx.cfg
```

Move on when: the BTS VTY answers (`telnet 127.0.0.1 4241`, osmo-bts default VTY port — **TO BE CONFIRMED** for your config) and the BSC has accepted OML (`show bts 0` on the BSC VTY, 4242). The BTS is already sending its TRXC commands to 5701; nobody answers yet, which is normal: the bridge (step 5) will receive them.

Check: `ss -ulnp | grep -E ':580[0-2]'` → osmo-bts-trx on 5800-5802 (osmo-bts-trx default; `pont/trx.py:184` sends the clock back there).

2.40.1.3 Step 1 — `c54x_exe --arm` (the DSP)

Clean up first (`run.sh:96-99`):

```
rm -f /dev/shm/calypso_api_ram /tmp/calypso_dsp.sock /dev/shm/calypso_tch_cfg
```

Command (`run.sh:102`, with `INSNS=120000` from start-native.sh, `IQ=none`, `AMP=30000`, `VERB=-v`):

```
cd /opt/GSM/c54x_exe
CALYPSO_IQDUMP_FCCH=1 CALYPSO_BSP_ATTENTE_MS=40 \
  ./c54x_exe --arm --insns 120000 --iq none --amp 30000 -v > /tmp/c54x-pont/dsp.log 2>&1 &
echo $! > /tmp/c54x-pont/dsp.pid
```

- Publishes `/dev/shm/calypso_api_ram` and `/tmp/calypso_dsp.sock`; listens on **UDP 6702** (BSP) (`LAUNCH.md:42-45`).
- ROM read from `/opt/GSM` (`--rom-dir`, `src/main.c:105`).

Move on when (`run.sh:104`, 5 s wait):

```
test -S /tmp/calypso_dsp.sock && grep -a "en attente de l'ARM" /tmp/c54x-pont/dsp.log
# pont : en attente de l'ARM sur /tmp/calypso_dsp.sock (API RAM : /dev/shm...)
ss -ulnp | grep 6702
```

("en attente de l'ARM" = "waiting for the ARM".)

2.40.1.4 Step 2 — QEMU / qosmo (the ARM + osmocom-bb layer1)

Command (`run.sh:121-136`, `GDB=1`, `GDB_STUB=1234`, `MONITOR=/tmp/qemu-monitor-pont.sock`):

```
rm -f /tmp/qemu-monitor-pont.sock
CALYPSO_PONT_RETRY_DIV=64 CALYPSO_DSP_EXTERN=1 \
  /opt/GSM/qosmo/build/qemu-system-arm -M calypso -cpu arm946 -display none -parallel none \
    -serial pty -serial pty -monitor unix:/tmp/qemu-monitor-pont.sock,server,nowait \
    -gdb tcp:127.0.0.1:1234 \
    -kernel /opt/GSM/firmware/board/compal_e88/layer1.highram.elf > /tmp/c54x-pont/qemu.log 2>&1 &
echo $! > /tmp/c54x-pont/qemu.pid
```

Optional gdb console (`run.sh:138-142`) — the default path in `run.sh:48` is dead, use qosmo's:

```
python3 /opt/GSM/qosmo/tools/gdb-telnet.py --port 44444 --stub 1234 \
  --elf /opt/GSM/firmware/board/compal_e88/layer1.highram.elf > /tmp/c54x-pont/gdb.log 2>&1 &
```

Move on when (`run.sh:143-145`, 10 s wait):

```
grep -a "label serial0" /tmp/c54x-pont/qemu.log          # char device redirected to /dev/pts/N (label serial0)
grep -a "pont DSP : API RAM partagee" /tmp/c54x-pont/qemu.log
```

("API RAM partagee" = "shared API RAM".) Then extract the modem pty (`run.sh:149`):

```
sed -n 's/.*redirected to \(\/dev\/pts\/[0-9]*\) (label serial0).*/\1/p' /tmp/c54x-pont/qemu.log | head -1 >
↪  /tmp/c54x-pont/modem.pty
cat /tmp/c54x-pont/modem.pty
```

On the DSP side you should see `RESET #1 ... pc=0xff80` then `DSP boote (premier IDLE)` ("DSP booted (first IDLE)") (`LAUNCH.md:97`, `src/pont.c:1608,1921`); they may only show up once the firmware is loaded (step 3) — exact timing **TO BE CONFIRMED**. Other expected QEMU lines (`LAUNCH.md:89-96`): `[trx] pont DSP : RESET_DSP relache par le firmware -> PONT_RESET`, `[trx] pont DSP : le TDMA du firmware prend le relais du timer de boot`.

Check the machine: `printf 'info status\n' | socat - UNIX-CONNECT:/tmp/qemu-monitor-pont.sock` → `VM status: running`.

2.40.1.5 Step 3 — `osmocon` (romload + L1CTL relay)

Command (`run.sh:161-164`):

```
rm -f /tmp/osmocom_l2
stdbuf -oL -eL /opt/GSM/osmocom-bb/src/host/osmocon/osmocon -m romload -i 100 \
  -p "$(cat /tmp/c54x-pont/modem.pty)" -s /tmp/osmocom_l2 \
  /opt/GSM/firmware/board/compal_e88/layer1.highram.bin > /tmp/c54x-pont/osmocon.log 2>&1 &
echo $! > /tmp/c54x-pont/osmocon.pid
```

Move on when (`run.sh:165`, 30 s wait; `LAUNCH.md:118-119`):

```
grep -a "your code is running now" /tmp/c54x-pont/osmocon.log
# Received ident ack / Progress: 100% / Received branch ack, your code is running now!
test -S /tmp/osmocom_l2
```

The firmware's last printf can also be read from the monitor (`LAUNCH.md:98-99`): `printf 'xp /96bx 0x008305f0\n' | nc -U /tmp/qemu-monitor-pont.sock` → `DSP API Version: 0x4e2a 0x491a`.

2.40.1.6 Step 4 — `mobile` (layers 2/3)

Command (`run.sh:177-191`, `L23_SYNC_RETRIES_SELECTION=8` in DSP mode):

```
ss -ltn | grep -q ':4347 ' && echo "VTY 4347 deja prise"      # run.sh:178-181
L23_SYNC_RETRIES_SELECTION=8 stdbuf -oL mobile -c /opt/GSM/c54x_exe/mobile_pont.cfg \
  > /tmp/c54x-pont/mobile.log 2>&1 &
echo $! > /tmp/c54x-pont/mobile.pid
```

Move on when: the process is still alive after 2 s (`run.sh:192-193`); within 5 s, on the osmocon side: `L1CTL_PM_REQ`, `L1CTL_RESET_REQ: FULL!`, `L1CTL_FBSB_REQ (arfcn=514 …)` (`LAUNCH.md:135-137`). Without the bridge: `FBSB RESP: result=255` in a loop, expected (`LAUNCH.md:138`).

Check: `telnet 127.0.0.1 4347` then `show ms`.

2.40.1.7 Step 5 — `pont_dsp.py` (BTS ↔ DSP)

Command (`run.sh:200-206`: the bridge is started from the osmo-operator root, with `--dsp-port 6702`, **without** `--no-record` since `PONT_AIRREC=1`):

```
cd /opt/GSM/osmo-operator
PONT_BSC_CFG=/etc/osmocom/osmo-bsc.cfg PONT_ARFCN=514 PONT_BSIC=7 \
  python3 pont/pont_dsp.py --dsp-port 6702 > /tmp/c54x-pont/pont.log 2>&1 &
echo $! > /tmp/c54x-pont/pont.pid
```

`PONT_BSC_CFG`, `PONT_ARFCN`, `PONT_BSIC` are at their defaults here (`pont/config.py:46-52`); adapt them to **your** cell. I/Q recording writes to `/root/record.cfile` by default (`config.py:81`): add `--no-record` if you don't want it.

Move on when (`run.sh:206`, 10 s wait):

```
grep -a "pont TRX : ports" /tmp/c54x-pont/pont.log
# pont TRX : ports 5700/5701/5702, ARFCN 514, BSIC 7, avance UL 3 trames
grep -a "horloge armee" /tmp/c54x-pont/pont.log     # BTS 127.0.0.1 : horloge armee vers le port 5800
grep -a "CTRL " /tmp/c54x-pont/pont.log | head      # TRXC commands from the BTS (POWERON, RXTUNE…)
```

("horloge armee" = "clock armed"), then `STATS fn=… | DL bursts=N` lines with N growing (`LAUNCH.md:157-158`).

> **Noise (optional)**: to degrade the downlink, start `c54x_exe` with `CALYPSO_BSP_PORT=16702` and `python3 /opt/GSM/qosmo/tools/injecteur_bruit.py --cible dsp --mode ber --ber 0.01` **before** this bridge (it takes 6702); `run.sh` does it with `BRUIT_MODE=…` (step b). See Injecteur-bruit.

2.40.1.8 Step 6 — check the service

What	Command	Expected	Source
SI read	`grep -a "New SYSTEM INFORMATION" /tmp/c54x-pont/mobile.log`	SI 1, 2, 3, 4…	`run_real.sh`
LAI	`grep -aoE "lai=[0-9]+-[0-9]+-[0-9]+" /tmp/c54x-pont/mobile.log | tail -1`	`lai=001-01-<your LAC>`	`run_real.sh`
RACH sent	`grep -a "\[montant\] RACH" /tmp/c54x-pont/dsp.log ; grep -ao 'UL bursts=[0-9]* tard=[0-9]* rach=[0-9]*' /tmp/c54x-pont/pont.log`	rach > 0	`run_real.sh`, `LAUNCH.md:66-67`
camp	`show ms` (VTY 4347)	`C3 camped normally`	`tests/modules/60-camp.sh:10`
LU	`grep -a "LOCATION UPDATING ACCEPT" mobile.log ; show ms`	`MM idle, normal service` ; IMSI in `show subscriber cache` (MSC VTY 4254)	`run_real.sh`, `tests/modules/70-attache.sh:9-10`
call	VTY 4347: `enable`, `call 1 600` (600 = the bench echo; otherwise your number)	`call control state: ACTIVE` in `show ms` ; `call 1 hangup`	`tests/modules/_lib.sh:48,83`, `90-appel.sh:15,22`
A5/1	`dsp.log` : `[montant] TCH : le firmware poste la tache 13 …`		`LAUNCH.md:68-69`

2.40.1.9 Stop and clean up

Reverse order: bridge, mobile, osmocon, gdb, QEMU, DSP, then purge (`run.sh:210-237`):

```
for n in pont mobile osmocon gdb qemu dsp; do kill "$(cat /tmp/c54x-pont/$n.pid 2>/dev/null)" 2>/dev/null; done
sleep 1
rm -f /dev/shm/calypso_api_ram /tmp/calypso_dsp.sock /tmp/osmocom_l2 /tmp/qemu-monitor-pont.sock \
      /tmp/c54x-pont/modem.pty /tmp/osmocom_sap /tmp/ms_data /dev/shm/calypso_horloge \
      /dev/shm/calypso_rach /dev/shm/calypso_sdcch_ul /dev/shm/calypso_tch_facch_ul \
      /dev/shm/calypso_tch_sacch_ul /dev/shm/calypso_tch_ul /dev/shm/calypso_tch_cfg
```

Why each file matters: a leftover `/tmp/ms_data` makes the next mobile's bind fail silently; a leftover `calypso_horloge` slaves the bridge to a dead session (`run.sh:218-227`).

2.40.1.10 Variant: gr-gsm setup, by hand

Same steps **without step 1** and without the three `export`s at the top (`run.sh:92-93,113`):

- **Step 2**: the same QEMU command **without** `CALYPSO_DSP_EXTERN=1` or `CALYPSO_PONT_*`; wait for `backend gr-gsm` in qemu.log (`run.sh:146-147`): `[l1] backend gr-gsm : GSMTAP udp/4730, SCH udp/4731`. QEMU then listens on UDP 4730/4731.
- **Step 3**: identical.
- **Step 4**: `mobile -c <cfg>` **without** `L23_SYNC_RETRIES_SELECTION` (`run.sh:186-189`); config with `io-tch-format rtp` (see Installer-grgsm_exe, **TO BE CONFIRMED**).
- **Step 5**: `cd /opt/GSM/osmo-operator && python3 pont/pont.py > /tmp/c54x-pont/pont.log 2>&1 &` (`run.sh:37-38,201-204`; rejects `--dsp-port`, `pont.py:25-26`). In the full bench, `start-native.sh` waits until osmo-bts-trx is (re)seen, then runs `setsid python3 -u pont/pont.py` to `/dev/shm/pont.log`.
- Expected on the bridge side: `pont TRX : ports 5700/5701/5702, ARFCN 514, BSIC 7`, then `STATS fn=… | DL bursts=N`.

2.40.1.11 Mapping to the scripts

Step	`c54x_exe/run.sh`	`qosmo/run_modules` (gr-gsm setup via start-native.sh)
DSP	`etape1 :92-106`	—
QEMU	`etape2 :108-153`	`40-qemu.sh:44-89` (via the `/usr/local/bin/qosmo` launcher if present)
osmocon	`etape3 :155-170`	`50-osmocon.sh:36-42` (plus `-d tr`)
mobile	`etape4 :172-195`	`70-l2.sh:65` (`mobile -c $cfg -d $CALYPSO_MOBILE_DEBUG`)
pont	`etape5 :197-208`	`start-native.sh` block `CALYPSO_BRIDGE=pont`
BTS	— (your network)	`60-bts.sh:49`

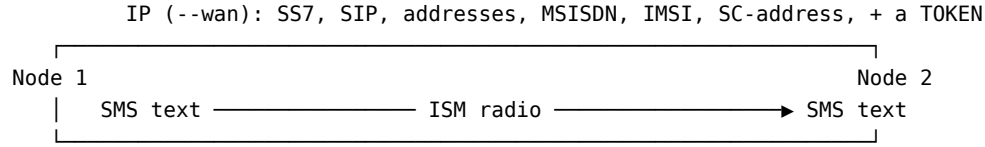
In DSP mode, `start-native.sh` plays qosmo's plan **without** `qemu,pty,osmocon,l2` (`DSP_MODULES_RETIRES`, `start-native.sh:602`) then `execs c54x_exe/run.sh`: that is exactly step 0 (qosmo's plan brings up the network and the BTS) followed by steps 1 to 5.

2.41 /opt/GSM/osmo-operator/wiki/Payload-ISM.md

5810 octets, 114 lignes

2.41.1 SMS payload over ISM radio, the rest over IP (`--payload-ism`)

Between two machines running osmo-operator, `--wan` already carries the whole inter-node link over IP: SS7 signaling, SIP voice, and the SMS relay (`scripts/sms-interop-relay.py`) which carries the message text in a JSON over TCP. The `--payload-ism` option keeps all of that over IP **except the SMS content**: the text travels over an **ISM-band radio link** (433 / 868 / 915 MHz, 2.4 GHz, license-free), while the two relays still coordinate over TCP.



2.41.1.1 How it works

The SMS relay already extracts the text from the TPDU of a mobile-originated SMS to reinject it as mobile-terminated on the other node. With `--payload-ism`:

1. **Transmission.** The relay draws a correlation token, transmits the text over the ISM radio stamped with that token, and sends over TCP only the token (plus the addresses). The TCP text field goes out empty.
2. **Reception.** The other relay receives the TCP JSON, sees `payload_via=ism`, and waits on the radio link for the text carrying that token before injecting the MT.

If the radio fails on transmission, the relay falls back to IP for that SMS (it never stays silent). If the text does not arrive over radio within the timeout (`recv_timeout`, 30 s by default), reception answers with an error.

The radio link is decoupled from the hardware: `scripts/ism_payload.py` talks to the **data interface** of your radio, frames the text into a single JSON line with a CRC32 (a truncated or corrupted frame is ignored), and correlates by token.

2.41.1.2 The backends

You choose the backend with `network/setup-payload-ism.sh`, which writes the `[payload]` section of `sms-routing.conf` (read by the relay).

backend	for what	setup
`serial`	an ISM module on UART (LoRa, RFM69, CC1101, HC-12…) on `/dev/tty*`	`sudo ./network/setup-payload-ism.sh --serial /dev/ttyUSB0 --baud 9600`
`exec`	an SDR pipeline you provide (HackRF/LimeSDR/USRP + GNU Radio, rtl_433…)	`sudo ./network/setup-payload-ism.sh --exec --tx-cmd … --rx-cmd …`
`fifo`	two named pipes, to hook up your radio some other way (`socat`…)	`sudo ./network/setup-payload-ism.sh --fifo --tx /tmp/ism_tx --rx /tmp/ism_rx`
`loopback`	try the SMS chain **without radio**, on a single machine	`sudo ./network/setup-payload-ism.sh --loopback`

For the `exec` backend, `setup-payload-ism.sh --exemples` drops two templates `/opt/GSM/ism/ism_tx.sh` and `ism_rx.sh`: `ism_tx.sh` receives a frame (a JSON line) on its standard input and must transmit it; `ism_rx.sh` must write each received frame to its standard output. Until they are filled in with your radio chain, the text does not get through.

2.41.1.3 Launch

On **each** node, with the WAN already in place:

```
# 1. set up the radio link (once, or redone by start-native)
sudo ./network/setup-payload-ism.sh --serial /dev/ttyUSB0 --baud 9600

# 2. launch the bench: WAN over IP + payload over ISM
sudo ./start-native.sh --wan --payload-ism
```

`--payload-ism=serial` (or `=exec`, `=fifo`, `=loopback`) additionally writes the `[payload]` section at launch, with the device/commands taken from the environment: `ISM_SERIAL`, `ISM_SERIAL_BAUD`, `ISM_TX`, `ISM_RX`, `ISM_TX_CMD`, `ISM_RX_CMD`. Without a backend, `--payload-ism` reuses the section already written.

To go back to text over IP: `sudo ./network/setup-payload-ism.sh --off`.

2.41.1.4 Verify without radio

```
python3 scripts/ism_payload.py --selftest                # in-memory loop
python3 scripts/ism_payload.py --selftest --link fifo \
        --tx /tmp/ism_a --rx /tmp/ism_b                  # loop over pipes
sudo ./network/setup-payload-ism.sh --loopback           # full SMS chain, one node
```

The self-test sends three texts (including 140 bytes and accented characters) and checks the rejection of a frame with a bad CRC.

2.41.1.5 The `[payload]` section of `sms-routing.conf`

```
[payload]
mode = ism                ; ism = text over radio; absent/other = text over IP
link = serial             ; serial | exec | fifo | loopback
serial = /dev/ttyUSB0     ; (serial) the UART device
; tx = /tmp/osmo_ism_tx    ; (fifo) transmit pipe
; rx = /tmp/osmo_ism_rx    ; (fifo) receive pipe
; tx_cmd = /opt/GSM/ism/ism_tx.sh   ; (exec) transmit command
; rx_cmd = /opt/GSM/ism/ism_rx.sh   ; (exec) receive command
recv_timeout = 30         ; wait for the text over radio before failing (s)
```

2.41.1.6 Regulatory

Only transmit on an ISM band and power allowed where you are (433 and 868 MHz in Europe, 902-928 MHz in region 2, 2.4 GHz), or into a dummy load / Faraday cage. It is your responsibility as the bench's operator.

2.41.1.7 Files

file	role
`scripts/ism_payload.py`	the ISM payload link: backends, frame + CRC, correlation by token, self-test
`scripts/sms-interop-relay.py`	SMS relay: token over TCP, text over radio when `[payload] mode=ism`
`network/setup-payload-ism.sh`	writes `[payload]`, sets the serial baud rate, drops the SDR templates, self-test
`start-native.sh`	the `--payload-ism[=backend]` option

2.42 /opt/GSM/osmo-operator/wiki/README-notebook.md

7892 octets, 163 lignes

2.42.1 The install notebook

← README · Notebook: installer-osmo-jusquau-dsp.ipynb · See also: native, docker, tests

`wiki/installer-osmo-jusquau-dsp.ipynb` builds the whole bench **step by step, in your browser**: from a fresh Ubuntu 24.04 to a verified call on the emulated Calypso DSP (c54x_exe + TI mask ROM). Each step is one cell, followed by a check cell, so you see every component being built and can stop and look at any point.

It is a teaching path, not the fastest one. To simply get a working bench, use the ISO or the Docker images built by `build.sh`.

Target	Ubuntu 24.04 LTS (noble), x86_64, **fresh**
Disk	~12 GB free under `/opt`
Time	~1 h on 8 cores (Osmocom core 20–30 min, QEMU 10–15 min)
Network	gitea.osmocom.org, github.com, ftp.freecalypso.org, archive.ubuntu.com

> **Do not run it on a machine that already has a bench installed** (the ISO, a previous native install). Every step ends in `make install` into `/usr/local` followed by `ldconfig`: it would overwrite the libraries your existing bench uses. Use a disposable container (below) or a fresh VM. Step 0 warns when it finds Osmocom already in `/usr/local`.

2.42.1.1 Run it in a disposable container (recommended)

The container runs as root, so the notebook needs no sudo. `--privileged` is required by the later steps: `setcap` (step 2), SCTP for osmo-stp, tun/GTP for the GGSN, `ip_forward`. `--shm-size` keeps `/dev/shm/calypso_*` from filling Docker's 64 MB default.

On the host:

```
docker run -it --name nb-ubuntu --privileged --shm-size=2g -p 8888:8888 ubuntu:24.04 bash
```

In the container:

```
apt-get update && apt-get install -y python3-venv git
python3 -m venv /venv && /venv/bin/pip install -q jupyterlab
git clone https://github.com/bbaranoff/osmo-operator /nb
cd /nb && /venv/bin/jupyter lab --ip=0.0.0.0 --allow-root --no-browser \
  --LabApp.default_url=/lab/tree/wiki/installer-osmo-jusquau-dsp.ipynb
```

Open the `http://127.0.0.1:8888/lab/tree/…?token=…` URL it prints: the notebook opens directly. Run the cells in order with **Shift+Enter** and read each check before moving on.

If SCTP is refused, load the module on the **host**: `sudo modprobe sctp`.

A bare `ubuntu:24.04` container is addressed like a native machine: `network/osmo-ip-plan.sh` adds the node's private addresses itself (192.168.2.1 and 192.168.2.10, which osmo-ggsn and osmo-sgsn bind to). Only an image built from this repository (`OSMO_IMAGE=1`, set by the Dockerfile) leaves addressing to `start-docker.sh` / compose.

2.42.1.1.1 Stop and resume

The container is named and not `--rm`, so a closed terminal or a crashed Jupyter does not lose the build already done in `/opt/GSM`:

```
docker start -ai nb-ubuntu
cd /nb && /venv/bin/jupyter lab --ip=0.0.0.0 --allow-root --no-browser \
  --LabApp.default_url=/lab/tree/wiki/installer-osmo-jusquau-dsp.ipynb
```

Cells are idempotent: re-run the one you stopped at. When you are done: `docker rm nb-ubuntu`.

2.42.1.1.2 Testing local changes

Step 1 clones osmo-operator from GitHub, so the notebook always installs the **pushed** version. To test an edited notebook before pushing, mount your clone instead of cloning it:

```
cd /path/to/osmo-operator
docker run -it --name nb-ubuntu --privileged --shm-size=2g -p 8888:8888 \
  -v "$PWD/wiki:/nb" ubuntu:24.04 bash
# then, in the container, the same commands without the git clone, and
#   --LabApp.default_url=/lab/tree/installer-osmo-jusquau-dsp.ipynb
```

The notebook file is then edited in place on the host, but the patches, configs and scripts it uses still come from GitHub.

2.42.1.2 Run it on a fresh Ubuntu 24.04

Follow the *Before opening the notebook* cell: Jupyter in a venv (Ubuntu 24.04 refuses `pip install` outside one), and sudo unlocked **in the terminal that starts Jupyter**, because a notebook cannot answer a password prompt:

```
sudo apt-get install -y python3-venv git
python3 -m venv ~/.venv-jupyter && ~/.venv-jupyter/bin/pip install jupyterlab
sudo -v
~/.venv-jupyter/bin/jupyter lab
```

`sudo -v` only lasts 15 minutes without a sudo call, and the QEMU build alone can run longer. For a full run, prefer a temporary `NOPASSWD` rule (`sudo visudo -f /etc/sudoers.d/osmo-nb`) and remove it afterwards.

2.42.1.3 Run it unattended and keep the outputs

To execute every cell without the browser and save the result with all its outputs:

```
/venv/bin/jupyter nbconvert --to notebook --execute \
  --ExecutePreprocessor.timeout=-1 --output executed.ipynb installer-osmo-jusquau-dsp.ipynb
/venv/bin/jupyter nbconvert --to html executed.ipynb     # readable in any browser
```

It prints nothing while it runs and stops at the first failing cell with a `CellExecutionError` showing that cell and its error.

2.42.1.4 The steps

Step	What it does	Needed for the DSP call
0	`/opt/GSM`, the shared environment file, sudo check	yes
1	clone osmo-operator, apt dependencies and the `osmocom` account through `install.sh`	yes
1b	GNU Radio build dependencies	no (gr-gsm chain only)
2	capture capabilities for `dumpcap`	yes
3	the Osmocom core, same versions and options as the Dockerfile, with the force-rand-toy patches	yes
4	osmo-gapk, before osmocom-bb	yes
5	the SMSC (gsup-smsc-proto, sms-coding-utils)	for SMS
6	osmocom-bb host tools: osmocon, mobile, trxcon	yes
7	prebuilt Calypso firmware	yes
7b	the transceiver tree, `trx` firmware and host `transceiver` (`./start.sh --bts`)	no
8	qosmo, the Calypso QEMU	yes
9	c54x_exe and the DSP ROM from FreeCalypso	yes
9b	FreeCalypso Magnetite for the C139 (TMS470 under Wine), its flash image, rvinterf (`--ms-magnetite`)	no
10	grgsm_exe	no
11	runtime setup: configs, run-time scripts, directories	yes
11b	audio sinks	only to hear the call
12	start the network, the BTS, then the DSP bench: `./start.sh --bts-classique --ms`, or the Magnetite / `--bts` variants	yes
13	camp, location update, call to 600 (echo test)	yes
14	full test ladder from bbaranoff/tests	no
15	stop	—

The launcher is `./start.sh` since 2026-10-21; `start-native.sh` is a symlink to it and still works.

To start the bench **process by process** instead of through `start.sh`, run the companion notebook lancer-pas-a-pas.ipynb once the install is done: it installs nothing, starts the network daemons one by one, then the emulated phone (DSP, QEMU, osmocon, mobile, bridge), and checks each one before the next, with optional sections on

the `--bts` variant (12b), the classic BTS seen from the bridge (12a) and the rssi application (`banc-rssi`, last section). The same steps as a page: Lancer-a-la-main.

2.42.1.5 Troubleshooting

A cell hangs with no output. It is probably waiting for a sudo password it can never receive. Interrupt the kernel, run `sudo -v` (or add the `NOPASSWD` rule) in the terminal that started Jupyter, and re-run the cell.

`configure: error: Package requirements (…) were not met.` A `-dev` package is missing (for example `sqlite3` → `libsqlite3-dev` for osmo-hlr). Install it, then re-run the cell. On a fresh system this means the apt list of step 1 is incomplete: please open an issue.

Step 11b prints "no audio server reachable". Expected in a container or on a server. The core, camp, location update, SMS and call set-up do not need audio.

Binder or Colab. They can display the notebook but cannot run it to the end: Binder gives no root and about 2 GB of RAM, Colab runs Ubuntu 22.04 without `--privileged` or SCTP.

2.43 /opt/GSM/osmo-operator/wiki/Reference.md

49231 octets, 1080 lignes

> Full reference, moved here from the top-level `README.md` on 2026-10-18. The new README is an index pointing to the topic READMEs and to this wiki.

2.43.1 osmo-operator-desktop.iso

Figure 6: Build ISO

Full desktop ISO :
Google Drive link

Four Part Github desktop ISO :
Github link Desktop Part 00
Github link Desktop Part 01
Github link Desktop Part 02
Github link Desktop Part 03
Github link Sha256

> [!IMPORTANT] Download **every** `.part-NN` file, not just the first one. GitHub caps release files at 2 GiB, so the image ships in pieces.

```
cat osmo-operator-desktop.iso.part-* > osmo-operator-desktop.iso
sha256sum -c --ignore-missing SHA256SUMS
```

A complete **GSM + LTE operator on a single live image**, with a GNOME desktop for everything that can't be driven from the VTY. **No SDR and no physical phone**: the radio is emulated end to end, all the way down to the baseband.

Figure 7: The bench in action: CSFB call, I/Q spectra, srsUE, Wireshark, Linphone

Live ISO password	`osmo`
postmarketOS phone password	`147147`
Minimum host	x86_64, 8 GB RAM, 4 cores, nested virtualization in a VM

Image identity	`IMAGE_VERSION="OSMO_EGPRS_V2"` · `OSMO_ROLE=operator` · `OSMO_LITE=0`

Contents

- Download
- Run the image — QEMU, VirtualBox, VMware, USB stick
- Getting started — the three dock launchers
- What's inside — 2G, DSP, 4G, CSFB, postmarketOS
- Building and running from source

2.43.1.1 Run the image

The live root is a tmpfs (~6 GB in `toram` mode); logs and captures are bounded and purged at every boot. The bench runs two QEMUs (Calypso + pmOS): in a VM, **enable nested virtualization**, or the phone VM falls back to software emulation and becomes very slow.

2.43.1.1.1 QEMU

```
qemu-system-x86_64 -cdrom osmo-operator-desktop.iso -m 8G -enable-kvm \
  -cpu host -smp 4 -nic user,hostfwd=tcp::8080-:8080
```

2.43.1.1.2 VirtualBox

1. New VM: Linux / Ubuntu 64-bit, 8 GB, 4 CPUs, 3D off.
2. System → Processor: tick *Nested VT-x/AMD-V* (if greyed out: `VBoxManage modifyvm "<name>" --nested-hw-virt on`).
3. ISO on the optical drive — no disk needed.
4. NAT port forward: host 8080 → guest 8080.

2.43.1.1.3 VMware

New VM from the ISO, Linux / Ubuntu 64-bit, 8 GB, 4 cores, tick *Virtualize Intel VT-x/EPT or AMD-V/RVI*, skip Easy Install.

2.43.1.1.4 USB stick (hybrid ISO)

```
sudo dd if=osmo-operator-desktop.iso of=/dev/sdX bs=4M status=progress conv=fsync
```

On Windows, use Rufus in *DD Image* mode. Boots UEFI or BIOS. On real hardware KVM is available directly — the most comfortable setup.

2.43.1.2 Getting started

Three launchers are pinned in the dock. 2G and 4G are independent and can start in any order; **launch the smartphone last**, once both networks are up.

	Launcher	What it starts
	2G — the red handset	The GSM core and the emulated radio: osmo-stp / bsc / msc / hlr / sgsn / ggsn / pcu, the Calypso under QEMU, the two BTS and the two osmocom-bb mobiles, Asterisk, the SMSC.
	4G — the signal bars	open5gs, srsENB and srsUE over ZMQ, and the SGs interface to the MSC (the CSFB).
☎	**Smartphone** — the Adwaita phone icon	The postmarketOS VM: oFono on the virtual modem, data on 4G, voice and SMS on 2G.

Run the 2G handset on the Calypso DSP instead of gr-gsm:

```
cd /opt/GSM/osmo-operator && sudo ./start-native.sh --dsp
```

`--menu` asks interactively (layer 1 `grgsm_exe` or `c54x_exe`). Check that everything is in place with `checks/diag-stp-operator.sh`.

2.43.1.3 What's inside

2G / GSM — Osmocom

- Full core: osmo-stp (M3UA, SCTP :2905), osmo-bsc, osmo-msc, osmo-hlr, osmo-sgsn, osmo-ggsn (GTPv1-C/U), osmo-pcu, osmo-mgw (MGCP :2427).
- Two DCS1800 cells in LAC 1: BTS 0 (Cell ID 6001, BSIC 7) and BTS 1 (Cell ID 6012, BSIC 8), each on its own osmo-bts-trx.
- **A real emulated Calypso as layer 1**: `qemu-system-arm -M calypso -cpu arm946` (qosmo) runs `layer1.highram.elf`, loaded by `osmocon` exactly as on a C123. `pont.py` turns bursts into I/Q for `fake_trx.py`, then `trxcon` and two osmocom-bb `mobile` instances on the terminal side.
- GPRS/EDGE: BSSGP/NS to the PCU, PDP contexts through osmo-ggsn.
- Voice: external MNCC → Asterisk (SIP :5060, FR/HR) via osmo-sip-connector.
- SMS: SMS-over-GSUP → proto-smsc-daemon, SMPP :2775 with a test ESME, inter-operator relay :7890.
- A3A8 with deterministic RAND (`*-force-rand-toy` patches): the bench is reproducible from one boot to the next.
- Live GSMTAP capture, ready for Wireshark.

The DSP — the Calypso's TMS320C54x on TI's real mask ROM

- Default demodulator: gr-gsm (grgsm_exe). With `./start-native.sh --dsp`, the baseband's own DSP takes over: a C54x core emulated outside QEMU (c54x_exe), running the mask ROM dumped from a real Calypso (PROM0-3, DROM, PDROM).
- ARM (QEMU) and DSP share the API RAM through `/dev/shm/calypso_api_ram` and run in lockstep, one TDMA frame at a time, over `/tmp/calypso_dsp.sock`. Downlink bursts reach the DSP's serial port (BSP) over UDP 6702 via `pont_dsp.py`.
- What the ROM does by itself (bench runs of 2026-09-23): FCCH/SCH acquisition, BCCH decoding (SI1-4), camping, location update, SMS MO/MT, MO/MT calls with A5/1 and speech audible both ways.
- **Still a work bench, not the demo path**: every TCH/F frame is flagged BFI even though speech stays intelligible, SACCH occasionally drops a call, the SB window is rarely armed, and the DSP needs 4.3–4.6 ms of host time per 4.62 ms frame. Details in the c54x_exe README.

4G / LTE — open5gs + srsRAN

- open5gs core (MME, SGW-C/U, SMF, UPF, PCRF, HSS…) on dedicated loopbacks, Prometheus :9090, WebUI :9999, MongoDB behind the HSS.
- PLMN 001/01, TAC 7. NAS null ciphering (EIA2/EIA1 + EEA0) on purpose: it's an analysis bench, NAS reads in clear in Wireshark.
- srsENB (10 MHz, EARFCN 3350, band 7) and srsUE over ZMQ, no SDR. Milenage test subscriber IMSI 001010001000001; the UE gets an IP in a netns and reaches the Internet through NAT.

CSFB — tying 2G and 4G together

- SGs between osmo-msc and the MME (:29118), mapping `TAI (001-01, TAC 7) → LAI (001-01, LAC 1)`.
- Combined EPS+IMSI attach: an incoming call is paged by the MME and the UE falls back to the BTS.
- Two subscribers (100101, 100102), one per cell, wired for an inter-cell call with fallback.

postmarketOS — the phone

- x86_64 pmOS VM with a patched kernel (`CONFIG_PPP` enabled, missing upstream); prebuilt kernel in `/opt/user_interface/kernel/pmos/`.
- 4G data: NetworkManager → pppd (`ATD*99***1#`) → virtio-console `osmo.data` → `osmo-phonesim-banc.py` → UE netns → srsUE → srsENB → UPF → Internet.
- 2G voice and SMS: virtio-console `osmo.modem`, where `osmo-phonesim-banc.py` plays a 27.007 AT modem driven by oFono. No dongle.
- Audio: two duplex PulseAudio chains (GAPK bridge to the osmocom-bb mobile, and an echo-cancelled chain).
- Result: data on 4G, voice and SMS on 2G, same identity — a real CSFB handset's behaviour at modem level.

2.43.2 osmo-operator — a full GSM network on one machine

A complete Osmocom GSM core (BTS, BSC, MSC, HLR, MGW, SMSC, Asterisk) plus an **emulated Calypso handset** — no radio hardware, no SIM card, no phone. The handset runs the real OsmocomBB `layer1` firmware on an emulated TI Calypso baseband, camps on the cell, authenticates, ciphers, sends SMS and holds a voice call.

Everything below is the **standalone bench**: one operator, one machine. That is what you want first, and it is what most of this document is about. Multi-operator SS7 interconnect exists and is covered in §9, but you do not need it to get a phone on the air.

2.43.2.1 1. Quick start

2.43.2.1.1 1.1 From the published image

```
sudo  docker pull ghcr.io/bbaranoff/osmo-operator/osmocom-nitb:latest
git clone https://github.com/bbaranoff/osmo-operator
cd osmo-operator
sudo ./launch/start-docker.sh
```

Then enter the container and bring the stack up:

```
sudo docker exec -ti osmo-operator-1 bash

cd /opt/GSM/osmo-operator
./start-native.sh --regen     # regenerate configs from the templates (first run)
./start-native.sh --stop      # make sure nothing is left over
./start-native.sh             # start the bench
```

`start-native.sh` attaches to tmux at the end. You are looking at a live GSM network with a mobile camped on it.

2.43.2.1.2 1.2 Native, without Docker

The repository installs directly on Ubuntu 24.04 (noble — the same base as the Dockerfile):

```
sudo ./install.sh                 # everything: deps, sources, build, binaries, configs, desktop
./install.sh --list               # show the steps, do nothing
./install.sh --check              # what is already in place
sudo ./install.sh --reinstall     # replay everything
sudo ./install.sh --only bureau   # desktop icons and shortcuts only
```

The launcher is the same `./start-native.sh`, which delegates to `/opt/GSM/qosmo/run.sh`.

2.43.2.1.3 1.3 Stopping

```
./start-native.sh --stop      # stops the stack AND the DSP bench, with or without --dsp
./start-native.sh --status    # what is running
```

`--stop` never needs `--dsp` to be repeated: it tears down the C54x bench too, so the next start finds a clean machine.

2.43.2.2 2. The two layer-1 chains

This is the one choice that matters on the standalone bench. Both run the same core network, the same BTS and the same `mobile`; they differ in **who demodulates the air interface**.

	`--grgsm`	**`--dsp`** (default since 2026-10-01)
Layer 1	gr-gsm inside QEMU + `grgsm_exe` bridge	real TI mask-ROM C54x, **outside** QEMU
QEMU's job	ARM7 + gr-gsm layer 1	ARM7 only (`CALYPSO_DSP_EXTERN=1`)
Bridge	`pont/pont.py`	`pont/pont_dsp.py`
Status	end to end: camping, LU, auth, A5/1, SMS, voice call	camping, SI1-4, LU; TCH calls reach end to end but every speech frame is flagged `B_BFI`
Use it for	getting work done	studying the real DSP

```
./start-native.sh             # real C54x mask ROM (default)
./start-native.sh --dsp       # the same, explicitly
./start-native.sh --grgsm     # gr-gsm layer 1 (the former default)
```

2.43.2.2.1 2.1 Three trees, one QEMU

Since 2026-09-25 there is a **single QEMU tree**. The old `qosmo-grgsm` and `qosmo-dsp` forks are gone — they had no common root commit, so a platform fix had to be reapplied by hand in the other tree and, in practice, was not.

tree	what it is
`/opt/GSM/qosmo`	the QEMU fork: `-M calypso`, ARM7TDMI + platform. Carries `run.sh`, `run_modules/`, `cfgs/`. The layer 1 is a `configure` option, not a repository: `--enable-l1-grgsm` or `--enable-l1-dsp`.
`/opt/GSM/c54x_exe`	the TMS320C54x running **outside** QEMU on the real TI mask ROM (`rom/calypso_dsp.*.bin`). Compiles `qosmo`'s sources — it does not copy them.

tree	what it is
`/opt/GSM/grgsm_exe`	the gr-gsm layer 1 outside QEMU. 118 KB against the 93 MB `qemu-system-arm` that normally hosts it.

`qemu-system-arm -M help` names the layer 1 actually linked in — the shortest way to know what you built:

configure	`-M help` shows
(nothing)	`couche 1 : aucune`
`--enable-l1-grgsm`	`couche 1 : grgsm`
`--enable-l1-dsp`	`couche 1 : c54x`

2.43.2.2.2 2.2 The gr-gsm chain (default)



2.43.2.2.3 2.3 The DSP chain (`--dsp`)

`--dsp` does **not** select a different QEMU. The stack comes up exactly as in gr-gsm mode — core, BTS, side-car, tmux — and only the four Calypso modules (`qemu`, `pty`, `osmocon`, `l2`) are handed over to `c54x_exe/run.sh`, which runs five steps of its own:

```
1. c54x_exe --arm      the DSP: /tmp/calypso_dsp.sock + shared API RAM,
                       DL bursts on UDP 6702
2. qemu-system-arm     the ARM + osmocom-bb layer1, gr-gsm layer 1 cut out
3. osmocon             romload on the serial pty, L1CTL relay
4. mobile              osmocom-bb layers 2/3
5. pont_dsp.py         TRXD 5700-5702 facing osmo-bts-trx, bursts down to the
                       DSP, uplink through the /dev/shm side-bands
```

```
./start-native.sh --dsp                  # the full stack, then the DSP bench
MODE=dsp PONT=1 /opt/GSM/c54x_exe/run.sh  # the mobile chain alone, no network
BANC_DSP=none ./start-native.sh --dsp     # the stack only, no DSP bench
```

`--dsp` starts **no bridge of its own**: `c54x_exe/run.sh` runs its own on 5700-5702 (step 5). A second one on the same ports would die on *Address already in use* and leave a corpse behind at every start.

Step details and variables: `wiki/Start-native.md` § 8-dsp.

2.43.2.2.4 2.4 Profiles

The layer-1 flag is independent of the **profile**, which chooses how many base stations come up and how:

profile	stack
`faketrx-qemu` (default)	core + BTS#0 QEMU + BTS#1 faketrx — also `hybrid`
`faketrx`	core + `fake_trx` + `trxcon` + `mobile`
`qemu`	the Calypso QEMU pipeline alone
`noproc`	core only — also `core`

```
./start-native.sh faketrx
./start-native.sh --profile qemu
./start-native.sh --list        # show the plan without running it
./start-native.sh --dry-run     # walk through it with no side effects
```

2.43.2.2.5 2.5 The Calypso as the BTS transceiver (`--osmo-trx`)

[2026-10-19] A third shape of the DSP chain: the emulated phone no longer camps on BTS#0, it *is* BTS#0's radio. QEMU boots `trx.highram.elf` (the `trx` app of Sylvain Munaut's transceiver branch, tree `/opt/GSM/osmocom-bb-transceiver`), and the host app `transceiver` from the same tree replaces both `mobile` and the bridge: it talks L1CTL to the phone through osmocon and exposes the TRXD UDP interface (5700-5702) that `osmo-bts-trx` already expects. The rest of the stack (BSC, MSC, side-car) comes up unchanged.

```
./start-native.sh --bts          # [2026-10-20] alias of --osmo-trx ; implies --dsp ; c54x_exe/run.sh runs with OSMO_TI
./start-native.sh --bts --ms       # the BTS radio, then the second phone over the air (INSTANCE=2, see 2.7)
./start-native.sh --bts --ms --ts 2  # [2026-10-20] one Calypso TRX per timeslot: TS1 (TCH/F) served by a 3rd QEMU (OSMO
sudo armbian/build-armbian.sh      # [2026-10-20] SD image via github.com/armbian/build (userpatches/customize-image
```

(`--ms` alone, or no option, is the usual mobile + bridge setup.)

What changes in `c54x_exe/run.sh` when `OSMO_TRX=1`: step 2 loads `trx.highram.elf`, step 4 launches `transceiver -l /tmp/osmocom_l2 -m 0x01 -M 0x01 -b 7 --no-sync -e 3` instead of `mobile`, step 5 (the bridge) is skipped because the transceiver holds the same ports. `OSMO_TRX_TS_MASK` (default `0x01`: TS0 only, BCCH/CCCH) widens the timeslot set later; `OSMO_TRX_ARGS` and `TRANSCEIVER` override the arguments and the binary.

The host app needed four things the 2013 code did not have; they live in `patches/osmocom-bb-transceiver-qemu-bench.patch`, applied by the Dockerfile right after the clone and by `install_modules/40-patches.sh` natively: `--no-sync` (there is no cell to FBSB on inside QEMU: BTS mode as soon as the phone resets), `--ts-mask`, `--l1ctl/--base-port`, and `SETTSC`, `SETFORMAT` (TRXD v0 answered) and `NOMTXPOWER` acknowledged, since `osmo-bts-trx` 1.10 treats them as critical.

Verified on 2026-10-20 (bench run): the DSP extcode runs on `c54x_exe` (extended NDB at 0x2400 `active=1, tsc=7`), BTS#0 gets its clock from the firmware's frame number and comes up; the TRX emits FB and SB one burst in ten each plus the BCCH/CCCH normal bursts of the BTS (`[rom-tx] ... NB n, SB n, FB n` in `/tmp/c54x-pont/dsp.log`). Four things had to change to get there:

- `-m` only sets the **RX** mask now; the TX mask is `-M` (`OSMO_TRX_TX_MASK`, default `0x01`). The firmware (`prim_bts.c`) opens its TX window at TPU slot `(tx_start + 5) & 7`, three slots ahead of the downlink timeslot it serves, and used to serve TS0 only with the upstream multi-window mask `0xe3` (`tx_start == 5`, slot 3) while putting `tn = 1` for any other mask: with `-m 0x01` applied to TX too, every burst was a dummy. The patched firmware computes `SLOT = (8 - tx_start) & 7`, `tn = tx_start + SLOT`, so a single window (`0x01`) serves TS0; `c54x_exe` only plays one TX window per frame.
- `osmo-bts-trx` sends its bursts `fn-advance` frames ahead of the clock; the default (2) left them 1-4 frames early and the firmware discarded most of them (`STALE BURST` in osmocon.log). `configs/osmo-bts-trx.cfg` now has `osmotrx fn-advance 20` / `rts-advance 5` (the osmo-trx values).
- `osmo-bts-trx` **exits** when its transceiver disappears ("No more clock from transceiver"), i.e. after every `banc-stop`; `run.sh` relaunches it at step 4b when BTS#0 is absent (`OSMO_TRX_BTS_RELANCE=0` to disable). Wait for the old process to finish its shutdown (up to a minute) before `banc-trx`.
- The SCH the extcode encodes comes out half wrong on `c54x_exe`: compared with `gsm0503_sch_encode()` of the `sch` words of the extended page, the first 39 coded bits are exact and the last 39 (upper info bits, CRC-10 parity, tail) are ~50 % wrong, so the mobile's ROM read `crc_ko` at TOA 23 on every SB. The encoder lives in the DSP extension code (`dsp_extcode.c`), not in the ROM: a C54x instruction of that loop is emulated wrongly (to find). Meanwhile `c54x_exe` re-encodes the SCH before publishing it (`MONTANT_ROM_TX_SB_RECODE`, default on in TRX mode; `[rom-tx] SB recode` counts the wrong bits).

- The TRX's uplink: the mobile's bursts arrive through the air tagged with their GSM frame number (the mobile's `a_a5fn`), and the BSP of instance 1 has no SB to learn the firmware's clock from. `c54x_exe` now decodes the `sch` word of each SB it publishes (fn F emitted at tick T) and sets the BSP's fn/tick offset to `F - T + delta` (`calypso_bsp_trx_calage()`, `CALYPSO_BSP_TRX=1`, `CALYPSO_BSP_TRX_DELTA` default -1: the uplink burst of fn X is read in the DSP frame where the downlink burst fn X+1 goes out), so an uplink burst is played at the frame where `l1s_bts_resp` tags it with its own fn. `[ts0] mode TRX : offset fn-tick pose a ...` in `/tmp/c54x-pont/dsp.log`.
- The firmware's BTS "sync" task (`prim_bts_sync.c`, one normal-burst read on TS0 every 51 frames: it is what produces the `DATA_IND` the host turns into `IND CLOCK`) fed its TOA and frequency measurements into the TA and AFC loops. On the bench there is no cell to follow, and the first mobile burst that landed in that window (`freq_err=3967`) shifted the TPU and the frequency, the task stopped reporting, the clock stopped and osmo-bts-trx exited ("No more clock from transceiver"). The patch removes both loops from that task and keeps the measurement trace and the `DATA_IND`.
- The host re-sends `BTS_MODE` on every `SETRXGAIN`, `SETSLOT`, `HANDOVER` and `NOHANDOVER` command, and osmo-bts-trx sends one right after the RACH (channel activation). The firmware's handler did `l1a_mftask_set(0)` and re-enabled the BTS tasks, which killed the sync task mid-run: the clock stopped two seconds after every channel request. The handler is now idempotent: once in BTS mode it only updates the parameters.
- The real killer of the clock, found last: once the BSP of the TRX instance had stored its first uplink burst, `bsp_attendre_trame()` waited up to 40 ms at every tick whose frame had no burst (the mobile-side rule "a late BTS frame is worth waiting for"). Uplink bursts are rare (a RACH, then four bursts per SDCCH block), so the DSP/ARM lockstep collapsed two seconds after the mobile's first RACH: the firmware programmed nothing any more (121 DSP instructions per frame), no `DATA_IND`, no clock, osmo-bts-trx exited. In TRX mode (`g_trx_cale`) the BSP no longer waits: a frame without a burst is silence.
- With all four bursts of a block reaching the BTS at the right frame numbers, the SABM still did not decode: a capture of the TRXD uplink (TRX → BTS, UDP source port 5702) showed nearly all-zero bits, and the result words the extension code leaves for `l1s_bts_resp` (`d[5..33]`) were empty while the mobile's bursts were full. Like its SCH encoder, the extcode's normal-burst demodulator yields nothing right on `c54x_exe`. Second prosthesis (`calypso_bsp_trx_ul_prothese()`, `CALYPSO_BSP_TRX_UL=0` to disable): the BSP remembers the uplink NB it played, and at the next frame's phase A `c54x_exe` writes its exact bits into the result page the ARM is about to read (`[ts0] prothese montante` in dsp.log; the "bits already right" counter measures how far the emulated demodulator is from working). The RACH keeps its own path (I/Q to the host) and was never affected.
- `BTS_RX_LAG` (prim_bts.c) is 3 instead of 2 and the BSP delta -2: one frame of margin for the air's latency, without which three bursts out of four of each uplink block arrived after their window.

2026-10-20, 21:08 — first call: `call 1 600` from the mobile's VTY (4347) reaches CALL PROCEEDING, CONNECT and CONNECT ACKNOWLEDGE, the echo test answers, and speech comes back (240 000 PCM samples from the DSP in 30 s). What made it work, after the second-phone attempt below was abandoned: **one phone serving TS0 and TS1** (`OSMO_TRX_TCH=1`, masks `0x03`): the patched firmware fills every slot of its TX window and reports a second uplink slot (`rx[2]`, `data[72]`) with `tn = rx_start + 1`; `c54x_exe` publishes the downlink bursts of both slots from the extension page (`tx_page_publier`, `CALYPSO_TRX_TX_SLOTS=2`) instead of capturing the ROM's single TSP burst, and writes the mobile's TS1 bursts into the page (`prothese montante TS1`); the host attaches every masked timeslot to phone 1; the mobile's c54x_exe tags its bursts with the timeslot it transmits on (bits 4-6 of the ring's `src` byte, `montant_tn_dedie()`), which the air relays as the TRXD `tn`; the air decodes the ASSIGNMENT COMMAND on the TS0 SDCCH/4 and announces the TCH through `calypso_tch_cfg`. The result page is now chosen by **frame-number parity**, self-calibrated by voting on the extcode's own energy detections: the former `d_dsp_page`-based choice flipped from one boot to the next, which is why the location update "worked one time in two". The uplink margin is two frames (`BTS_RX_LAG 4`, delta -3). The VSP audio bridge (`pont.vsp_audio`) is now also started by the air step (5b): without it the call was established but silent.

The second-phone attempt (kept, off by default, `OSMO_TRX_TS1=1`): TS0 only carries CCCH and SDCCH/4, and the firmware opens a single RX window per frame, so the upstream design is followed: a **second phone** serves TS1 (`transceiver -2`, upstream hard-coded to `/tmp/osmocom_l2.2`, now `-L PATH`). `OSMO_TRX_TS1=1` on instance 1 starts a third chain (`INSTANCE=3`, `OSMO_TRX_PHONE=2`: DSP, QEMU with `trx.highram.elf`, osmocon on `/tmp/osmocom_l2_3`, no transceiver) before step 4, and the transceiver gets `-2 -L /tmp/osmocom_l2_3` (phone 2: TX and RX mask `0x02`, which the patched `prim_bts.c` maps to SLOT 0 / tn 1). Phone 2 emits no SB, so its BSP takes a fixed frame offset (`CALYPSO_BSP_TRX_OFFSET=-1`, the value measured on phone 1) instead of the SB calibration. `pont/air.py` grows a third endpoint (`--c-ring`, `--c-port`): phone 2's bursts go to the mobile tagged

tn 1, and the mobile's bursts go to phone 2 once the air has decoded the ASSIGNMENT COMMAND on the TS0 SDCCH/4 (`pont.gsm.xcch_decode`) and announced the TCH to the mobile's c54x_exe through `calypso_tch_cfg` (the same 16-byte record the bridge used to write); a RACH from the mobile hands the uplink back to phone 1. BSC: TS1 is `TCH/F`.

Still open: `DSP Error Status: 24` printed every frame by the trx firmware (DMA_PROG|DMA_TASK, known since 2026-09-22, harmless so far), and the C54x emulation bug behind the SCH prosthesis.

2.43.2.2.6 2.6 The C123 screen and keypad under QEMU

[2026-10-19] Three SoC devices were added to `qosmo` (`hw/arm/calypso/`): `calypso_armio.c` (the ARMIO block at 0xFFFE4800: the 5x5 key matrix, IRQ 8, as `calypso/keypad.c` scans it), `calypso_i2c_ctrl.c` (the real I2C controller at 0xFFFE2800, previously swallowed by the catch-all stub) and `calypso_st7558.c` (the 96x65 LCD, I2C slave 0x3c). Any firmware app that draws and reads keys now works: `rssi`, `menu`, `hello_world`, `trx`.

- The screen is published in `/dev/shm/calypso_lcd` (`CALYPSO_LCD_SHM`) on every flush, and also on the QEMU graphic console (`-display sdl`).
- Keys arrive on the Unix datagram socket `/tmp/calypso_keys.sock` (`CALYPSO_KEYS_SOCK`): `press 5`, `release 5`, `tap menu 200`. Key names: `0-9`, `star`, `hash`, `menu`, `lsb`, `rsb`, `up`, `down`, `left`, `right`, `ok`, `power`. `CALYPSO_KEYS_DEBUG=1` traces them on stderr.
- `tools/osmo-c123-gtk.py` (run with `/usr/bin/python3`) shows the photo of the phone from `data/c123.jpg`, overlays the live LCD in the `lcd` zone of `data/c123-zones.json` and sends a press/release for each key clicked; the PC keyboard works too. `--calibrate` redraws the zones with the mouse.

```
FIRMWARE_APP=rssi bash /opt/GSM/c54x_exe/run.sh     # steps 1-3 only: DSP, QEMU, osmocon
/usr/bin/python3 tools/osmo-c123-gtk.py              # the phone window
echo "tap menu" | socat - UNIX-SENDTO:/tmp/calypso_keys.sock   # or by hand
```

2.43.2.2.7 2.7 Two Calypsos and the air between them

[2026-10-19] With `--osmo-trx` the emulated phone is the BTS radio, so a second emulated phone is needed to be the mobile. `c54x_exe/run.sh INSTANCE=2` runs a complete second chain (DSP, QEMU, osmocon, mobile) next to the first. It re-executes itself in a private mount namespace where `/dev/shm` is the host's `/dev/shm/calypso2`, so none of the hard-coded `/dev/shm/...` paths of c54x_exe or qosmo had to change; the host's real `/dev/shm` stays readable at `/run/calypso-shm-hote`. Everything else is shifted: `/tmp/c54x-pont2`, `/tmp/osmocom_l2_2`, `/tmp/calypso_dsp2.sock`, gdb 1235/44445, keypad socket `/tmp/calypso_keys2.sock`, BSP UDP port 6712, mobile VTY +100, 4347 (a derived copy of `mobile_pont.cfg`).

The air is `pont/air.py`: each c54x_exe publishes the bursts its ROM emits in the ring `calypso_tx_rom` (now including the SB, dummy and FB bursts of a BTS, and one per frame in TRX mode), and each BSP accepts bursts in TRXDv0 over UDP. The relay feeds A's ring to B's BSP and B's ring to A's BSP. It is step 5 of instance 2 (`AIR=1`), in place of the bridge, which must not touch 5700.

```
banc-trx          # instance 1 : the BTS radio (restart it after a rebuild of c54x_exe or trx.highram)
banc-mobile       # instance 2 : the mobile, plus the air ; banc-mobile-stop / -status / -logs
c123-2            # its screen and keypad
```

First run on 2026-10-20: the air carries the TRX's bursts to the mobile (`/tmp/c54x-pont2/air.log`: `A->B n (NB SB DUMMY FB)`), the mobile's ROM detects the FCCH (`FB1 ... TOA=8759`) and receives the SB in its SB window (`[sbwin] ... FENETRE SB (one-shot)` in `/tmp/c54x-pont2/dsp.log`); the SB only decodes once the TRX side re-encodes it (see 2.5). The two instances are not slaved to each other: each runs real time (4.62 ms per frame) and the mobile's BSP waits up to 40 ms for a late frame. `air.py` keeps the relayed frame numbers continuous across a restart of `banc-trx` (the TRX publishes its tick as fn, which restarts at 0; without the offset the mobile's BSP waited for a frame in the future: "le DSP court devant le BTS").

2026-10-20, 19:30 — first location update through the two Calypsos: `LOCATION UPDATING ACCEPT (lai=001-01-1)`, `got TMSI 0x28a6ecbc`, `TMSI REALLOCATION COMPLETE` in `/tmp/c54x-pont2/mobile.log`, with the BTS radio being the first emulated phone and the air relay in between. What it took, in order, is listed in 2.5 (TX mask, fn-advance, BTS relaunch, SCH re-encoding, uplink calibration, no 40 ms wait, idempotent BTS_MODE, uplink bit prosthesis). Not yet exercised: SMS and calls over this path.

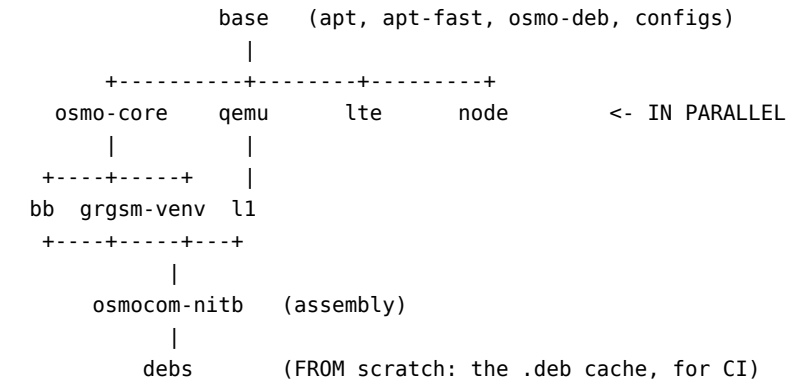
Later the same day, once the SCH was re-encoded: `=> SB 0x...: BSIC=7`, SI1/3/4/13 read, `CHANNEL REQUEST`, `IMMEDIATE ASSIGNMENT` received, the TRX host demodulating the mobile's RACH (`Access Burst Indication`) and its SDCCH bursts (`Normal Burst Indication`). The location update then failed because the BSC allocated an SDCCH/8 on **TS1**, which a one-timeslot TRX never serves: `configs/osmo-bsc.cfg` now has `timeslot 1 phys_chan_config TCH/F`, so the SDCCH comes from the SDCCH/4 of TS0. Restart osmo-bsc (systemd) for a config change to apply; osmo-bts-trx exits when OML drops and must be relaunched (`run.sh` step 4b does it at start).

2.43.2.3 3. Building the images

```
sudo ./build.sh               # build osmocom-nitb (apt-fast, compose v2, .deb cache)
sudo ./build.sh --no-cache    # recompile everything and rewrite the cache
sudo ./build.sh --arch=arm64  # cross-build through buildx + qemu-user-static
```

2.43.2.3.1 3.1 The build is a graph, not a chain

`Dockerfile` is split into BuildKit stages, so the long compilations that have nothing to do with each other run **at the same time**:



`bb`, `l1` and `grgsm-venv` are deliberately **not** parallel to `osmo-core`: `osmocom-bb` links `mobile/trxcon` against libosmocore, `c54x_exe` asks `pkg-config` for `libosmocoding libosmocore`, and gr-gsm's CMake looks for both. Pulling them out would break the build, not speed it up.

`--target osmocom-nitb` is **mandatory** — `debs` is the last stage in the file, so without it Docker builds an empty image. `build.sh` and `compose.yaml` already pass it.

2.43.2.3.2 3.2 The `.deb` cache

Everything the image compiles (libosmocore, osmo-*, gapk, QEMU and its tree, osmocom-bb, the gr-gsm venv, the firmware) comes out as a `.deb` through `packaging/osmo-deb.sh` and stays **on the host**, in `/var/cache/osmo-debs` (`OSMO_DEB_CACHE` to move it):

```
sudo ./build.sh             # 1st build: everything compiles, .deb go to the cache
sudo ./build.sh             # rebuild: dpkg -i, nothing compiles
sudo ./build.sh --no-cache  # recompile everything, rewrite the cache
./compose.sh debs           # what is in the cache
```

Each package carries **both** the installed files and the `/opt/GSM` source tree, so a cached rebuild still gives you a workshop you can read, patch and `make` in — not an image where only `/usr/local` survives. That property is also what lets the build stages hand their work to the assembly stage.

Standalone `.deb`, to install without git:

```
./packaging/build-debs.sh         # -> packaging/dist/
sudo dpkg -i packaging/dist/*.deb  # refuses to install over a git clone at the same path
```

2.43.2.3.3 3.3 Docker Compose

```
sudo ./compose.sh build [--no-cache] [--lite] [--stp]
sudo ./compose.sh up [--ms 2] [--phy faketrx]    # hub + osmo-operator-1, HLR populated
sudo ./compose.sh down | ps | logs | shell
```

2.43.2.3.4 3.4 ISO images

`sudo ./build-iso.sh` does **one** docker build (`osmocom-nitb`) and derives each rootfs from it: interstp, normal, lite (the normal one with the workshops removed), desktop. On the installed disk the accounts are the ones from the Calamares *Users* screen plus an unlocked root; LUKS is offered in the partitioning choices; a *Graphics drivers* page offers `nvidia-driver-610` when `lspci` sees a card.

2.43.2.4 4. Daily use

2.43.2.4.1 4.1 tmux

```
docker exec -ti osmo-operator-1 tmux attach
docker exec -ti osmo-operator-1 tmux -S /tmp/osmocom_tmux attach -t osmocom
```

key	window
`Ctrl-b 0`	faketrx
`Ctrl-b 1`	MS1 (trxcon + mobile)
`Ctrl-b 2`	Asterisk
`Ctrl-b 3`	SMSC (proto-smsc-daemon + relay)
`Ctrl-b w`	list all windows
`Ctrl-b d`	detach

`--no-attach` starts without attaching — that is what `osmo-banc.service` uses, since nobody is in front of it.

2.43.2.4.2 4.2 VTY

port	component
`4239`	OsmoSTP
`4242`	OsmoBSC
`4243`	OsmoMGW
`4254`	OsmoMSC
`4258`	OsmoHLR

```
OsmoMSC# show subscriber all
OsmoMSC# subscriber msisdn 10001 sms sender msisdn 10002 send Hello

OsmoHLR# subscriber imsi 001010000000001 show
OsmoHLR# show gsup-clients
```

Subscribers are provisioned automatically when the stack comes up, by `run_modules/21-abonnes-hlr.sh` — there is nothing to run by hand. How many it creates follows the numbering plan, and `--regen` re-applies it.

2.43.2.4.3 4.3 Ports on the wire

port	carries
`5700-5702` UDP	TRXD, between `osmo-bts-trx` and the bridge
`4729` UDP	GSMTAP, the Um interface
`4730` UDP	full SI set into the shunt (`si_bridge.py`)
`4731` UDP	SCH BSIC/FN out of the gr-gsm receiver
`6702` UDP	DL bursts into the DSP BSP (`--dsp`)
`8080` TCP	the web dashboard (`osmo-egprs-web`)

2.43.2.4.4 4.4 Wireshark

Started automatically on the Docker bridge with `sctp or udp port 4729`.

```
gsmtap                  # radio traffic (Um)
gsm_map                 # MAP messages
m3ua / sccp             # SS7, if you run multi-operator
```

2.43.2.4.5 4.5 Logs

Every component writes under `LOG_DIR`, which is `RUN_DIR/logs` — that is `/run/user/<uid>/osmo-nitb/logs` when the launching user has an `XDG_RUNTIME_DIR`, `/tmp/osmo-nitb/logs` otherwise (see `environment/paths.env`). The directory is on tmpfs on purpose: tmux and the desktop panel follow the files live, and tmpfs is the only filesystem here where inotify works reliably. `run.sh --stop` archives the directory under `../archives/<timestamp>`, truncates oversized archives and prunes old ones.

`mobile.log` is the journal of osmocom-bb `mobile` (MS#1). It is written by qosmo's `run_modules/70-l2.sh` (`mobile -c <cfg> -d <mask> >> mobile.log`), or, with `--dsp`, by `c54x_exe/run.sh` under `/tmp/c54x-pont`. Its format is fixed by the `log stderr` block that `start-native.sh` guarantees in every generated `mobile.cfg`: colour per subsystem, category prefix, the compact 17-digit timestamp (`logging print extended-timestamp 1`, `YYYYMMDDhhmmssnnn`; not `logging timestamp 1`, which libosmocore 1.12 renders as a 25-column ctime string), and the seven categories the bench follows in debug (mm, rr, cc, sms, plmn, sim, sap). Readers strip the ANSI codes and the timestamp themselves: the desktop panel (`tools/osmo-fft-snap.py`), `qosmo/tools/rapport-run.sh`, `scripts/banc-repro.sh`. The attach rung of `tests/banc-max.sh` (`modules/70-attache.sh`) also quotes the `LOCATION UPDATING ACCEPT` / `REJECT` lines of this file next to the VTY state, so an attach that was obtained and then lost reads differently from a location update that never got an answer.

Where the journal is, for tools of the session, is published by `start-native.sh` at start and removed at stop — the same way as the MS#1 VTY port:

path	holds
`/run/osmo-banc/ms1-vty`	the VTY port of MS#1 (4247, 4347 with `--dsp`)
`/run/osmo-banc/mobile-log`	the absolute path of `mobile.log`

Both are also pushed into systemd's environment (`OSMO_BB1_VTY_PORT`, `OSMO_MOBILE_LOG`). Setting `OSMO_MOBILE_LOG` by hand overrides the published path for the panel.

2.43.2.5 5. Diagnosing the radio side

Read `pont/README.md` before chasing a radio fault through a CRC counter. The causes documented there — overwritten Kc, the SACCH L1 header, C0 filling, an inert `CALYPSO_CANNED` — *all* present themselves as "the bridge has CRC errors".

```
./start-native.sh --check-paths     # verify the declared dependencies
./start-native.sh --verbose         # show the modules' output
```

Logs:

```
/var/log/osmocom/qemu.log        # ARM/DSP emulation, BSP, IRQ
/var/log/osmocom/bridge.log      # QEMU ticks, IND CLOCK, DL/UL bursts
/var/log/osmocom/run.sh.log      # orchestration
```

symptom	likely cause	check
`bridge: timeout`	QEMU never started the TPU/DSP	`grep TINT0 /var/log/osmocom/qemu.log`
`FBSB result=255` (no cell)	layer 1 is not detecting the FB	`grep "IMR change" /var/log/osmocom/qemu.log | wc -l`
`osmo-bts-trx: PC clock skew too high`	the bridge stopped sending `IND CLOCK`	restart the bridge
`/tmp/osmocom_l2` never created	the firmware is not booting	`head -50 /var/log/osmocom/qemu.log`
Address already in use on 5700	two bridges — a `--dsp` run left one behind	`./start-native.sh --stop`

QEMU monitor:

```
docker exec -ti osmo-operator-1 socat - unix-connect:/tmp/qemu-calypso-mon.sock
```

`--assembly-logs` gives a trace of the emulated ARM (`-d in_asm`: each block once, when it is translated). `ASSEMBLY_LOGS_FLAGS=in_asm,exec,nochain` also logs every execution; that makes the ARM ~150 times slower (14 x real time for the whole ARM+DSP lockstep). The bridge clock then follows the DSP down to `PONT_HORLOGE_MAX` times real time (4 by default, 16 when `ASSEMBLY_LOGS=1`, see `pont/trx.py`; the trace itself goes to `/var/tmp/c54x-pont`, not the tmpfs), but the wall-clock timers of `mobile` (T3210) and of the core network (T3101) still run, so the attach may still time out — use it to chase the firmware, not to measure the network.

2.43.2.6 6. The `qosmo` launcher

`40-qemu.sh` does not call `qemu-system-arm` directly: it calls the C launcher `qosmo`, installed next to it by `ninja install`. The launcher knows which layer 1 the QEMU it starts was built with, because `QOSMO_DSP` follows the `configure` option — nobody has to tell it twice.

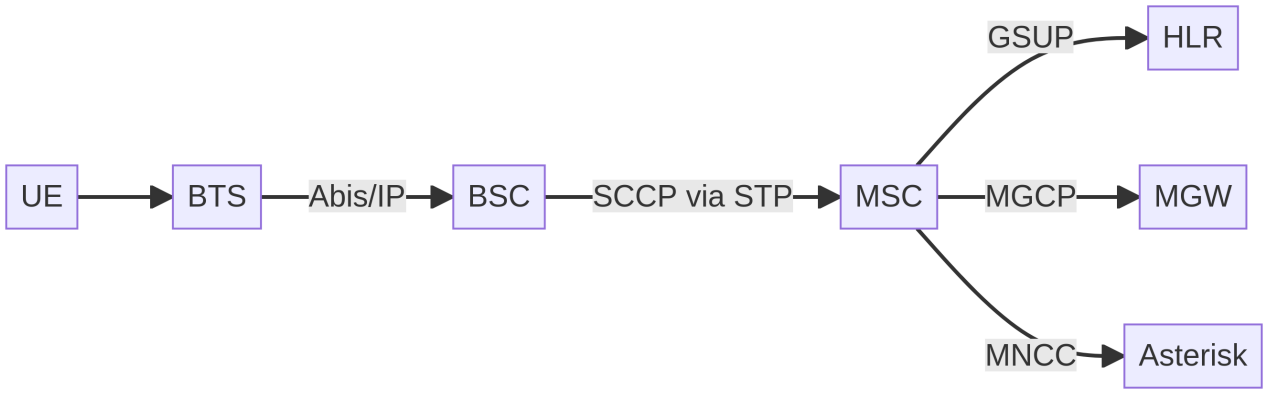
```
qosmo -k /opt/GSM/firmware/board/compal_e88/layer1.highram.elf
qosmo --help
```

Defaults are the ones that already work: `-M calypso`, `-cpu arm946`, `-gdb tcp::1234`, `-serial pty -serial pty`, a unix monitor, L1CTL on `/tmp/osmocom_l2`, TRXDv0 on `0.0.0.0:6702`. It reads `l1s/last_rach` out of the ELF (no `nm` needed), relays QEMU's output unchanged, and publishes stable links to the ptys under `<RUN_DIR>/`.

`start-native.sh` resolves it (`--launcher <bin>`, or `QOSMO_LAUNCHER`), builds it if the source is there and the binary is not, and passes it to `run.sh`. Without a launcher, `40-qemu.sh` falls back to the historical `qemu-system-arm` command line.

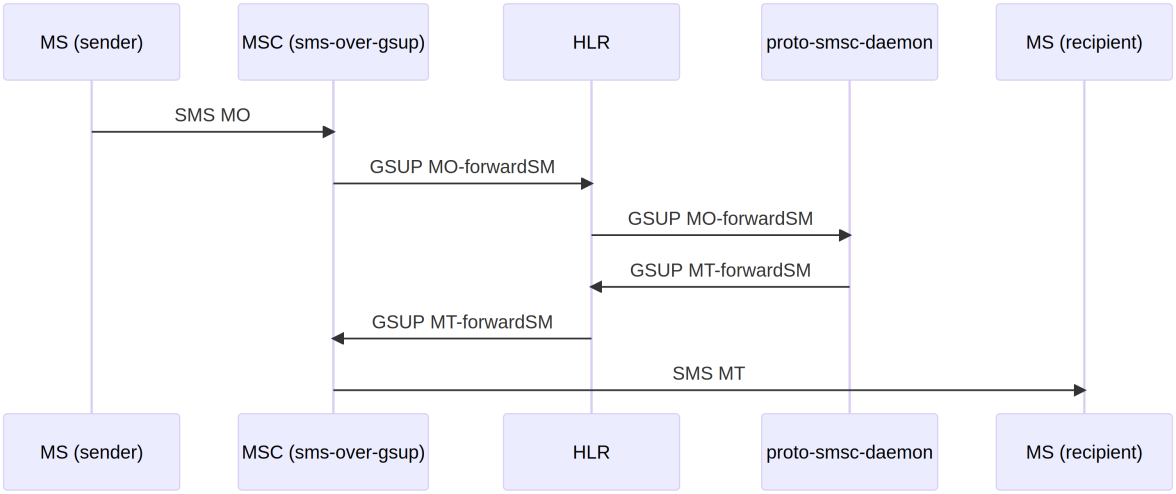
2.43.2.7 7. One PLMN

Every simulated operator is the standard Osmocom stack, all components in a single container talking over `127.0.0.1`.



The local STP is the intra-PLMN signalling hub. It listens on `127.0.0.1:2905` immediately, without waiting for the Docker interface — which is attached after the container starts, and used to be a race condition.

2.43.2.7.1 7.1 SMS



2.43.2.7.2 7.2 Voice

```
MS → BTS → BSC → MSC → MNCC → Asterisk → MNCC → MSC → BSC → BTS → MS
                                  ↕ MGCP
                               OsmoMGW (RTP)
```

2.43.2.7.3 7.3 Numbering

number	use
`N0001…N9999`	GSM subscribers of operator N
`100` / `200`	local softphones
`600`	echo test
`9XXXXX`	inter-operator out from a softphone

The first digit of `NXXXX` identifies the operator. The `[gsm_in]` dialplan works out on its own whether the destination is local or inter-op.

2.43.2.8 8. PHY modes

For the MS side, selected through `PHY_MODE` inside the container before calling `run.sh`:

`PHY_MODE`	stack	use
`faketrx` (default)	`fake_trx` → `trxcon` → `mobile`	multi-MS, fast, no DSP
`virtphy`	`osmo-bts-virtual` ↔ `virtphy` ↔ `mobile`	multi-MS over UDP multicast
`qemu`	`osmo-bts-trx` ↔ bridge ↔ Calypso QEMU ↔ `mobile`	emulated baseband, 1 MS

`PHY_MODE=qemu` forces `N_MS=1`: the bridge uses fixed UDP ports, so one emulated Calypso per container. For several virtual handsets, run several containers.

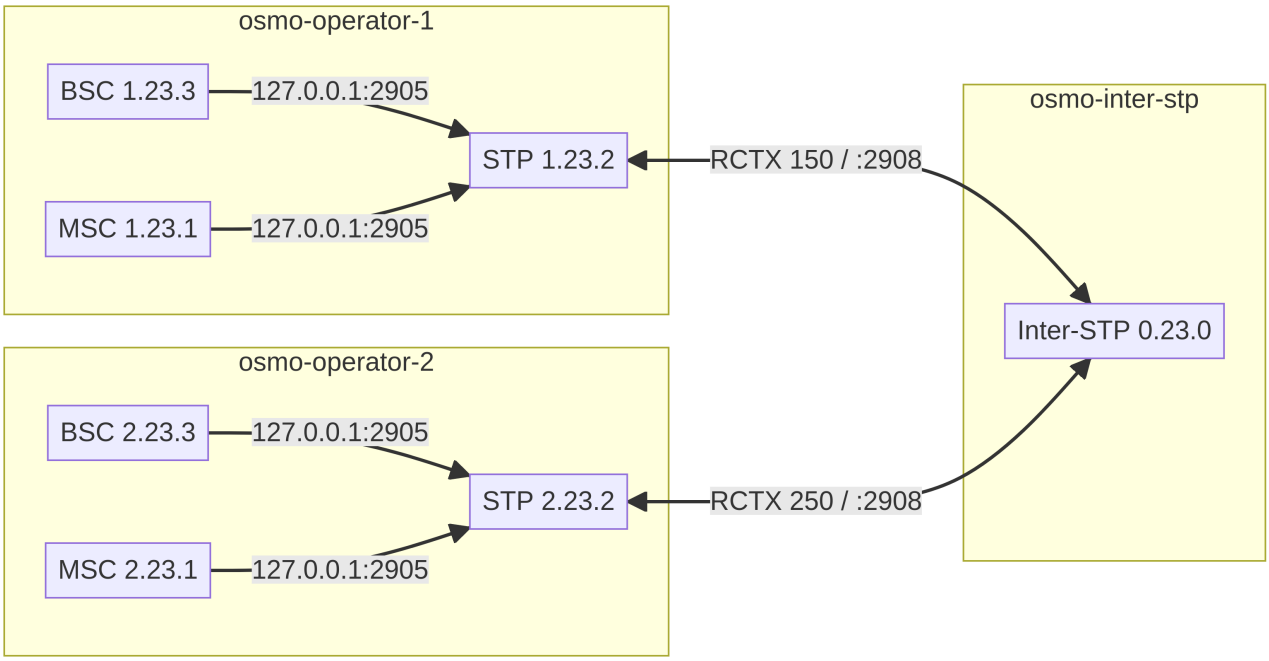
2.43.2.9 9. Multi-operator and SS7

Everything above is one operator. The bench also runs **N operators (1 to 9)**, each with its own core network in its own container, joined by a central Inter-STP that routes M3UA/SCCP between them. No configuration file is edited by hand: pjsip, dialplan, SMS routing and the inter-STP config are all generated from N.

```
sudo ./launch/start-docker.sh          # choose bridge mode, enter N operators
sudo ./launch/start-docker.sh stop
```

This is essentially a *DHCP for SS7* — the inter-operator SS7 setup normally done by hand, automated and reproducible.

2.43.2.9.1 9.1 Topology



2.43.2.9.2 9.2 Addressing

Docker network	range	role
`gsm-inter`	`172.20.0.0/24`	interop backbone (inter-STP M3UA)
`gsm-net-opN`	`172.20.N.0/24`	operator N private network

Operator N → backbone `172.20.0.(10+N)`, private `172.20.N.10`. The Inter-STP is at `172.20.0.10`.

2.43.2.9.3 9.3 Point codes and routing contexts

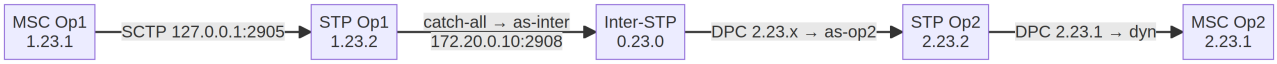
ITU 14-bit, `zone.network.node`:

node	point code	RCTX	formula
Inter-STP	`0.23.0`	`rctx_msc`	`N×100+10`
MSC OpN	`N.23.1`	`rctx_stp`	`N×100+20`
STP OpN	`N.23.2`	`rctx_bsc`	`N×100+30`
BSC OpN	`N.23.3`	**`rctx_inter`**	**`N×100+50`**

`rctx_inter` is the critical one: it must match in the operator's `osmo-stp.cfg` and be consistent with the connection to the inter-STP.

The inter-STP uses **AS without a routing key** (catch-all per operator); routing is on the DPC alone.

2.43.2.9.4 9.4 Message path



2.43.2.9.5 9.5 Generated configuration

`apply_config_templates()` in `start-docker.sh` resolves every N-dependent placeholder in a **single `sed` pass** (`__PC_MSC__`, `__RCTX_INTER__`, `__ARFCN__` = `512+N×2`, `__CONTAINER_IP__`, …), then appends the sections that depend on the *total* number of operators: N-1 PJSIP trunks, the `[interop_out]` dialplan, the SMS routing table, and N AS + N×3 SS7 routes in the inter-STP config.

Startup order is critical — the Inter-STP must be listening on `:2908` **before** the operator STPs try to connect.

2.43.2.9.6 9.6 Inter-operator SMS and voice

`sms-interop-relay.py` watches the `proto-smsc-daemon` MO log, parses the GSM 03.40 TPDU, looks up `sms-routing.conf` (longest-prefix match) and forwards over TCP 7890 to the target operator's relay, which resolves MSISDN→IMSI on the HLR VTY and calls `proto-smsc-sendmt`.

Voice crosses on a SIP trunk between the operators' Asterisk instances. The `[interop_out]` dialplan routes on the first digit of the dialled number.

2.43.2.9.7 9.7 Diagnosing PROHIB routes

```
sudo docker ps | grep inter-stp
sudo docker exec osmo-operator-1 sh -c 'echo "show cs7 instance 0 asp"   | telnet 127.0.0.1 4239'
sudo docker exec osmo-inter-stp  sh -c 'echo "show cs7 instance 0 asp"   | telnet 127.0.0.1 4239'
sudo docker exec osmo-operator-1 sh -c 'echo "show cs7 instance 0 route" | telnet 127.0.0.1 4239'
sudo docker exec osmo-operator-1 ping -c1 172.20.0.10
```

A healthy operator STP route table:

```
N.23.1/14   as-dyn-…  avail  avail  dyn   ← MSC (dynamic)
N.23.3/14   as-dyn-…  avail  avail  dyn   ← BSC (dynamic)
0.0.0/0     as-inter  avail  avail        ← catch-all to the inter-STP
```

symptom	likely cause	fix
route `0.0.0/0` PROHIB	inter-STP down, or wrong IP	check `INTER_STP_IP` and the backbone IP
no dynamic routes	MSC/BSC not reaching the local STP	check the ASPs point at `127.0.0.1`
ASP DOWN on the inter-STP	Docker race condition	`docker restart` the operator

2.43.2.9.8 9.8 WAN, mesh and multi-machine

Beyond a single host, `start-native.sh` also meshes several machines:

```
./start-native.sh --wan                  # interactive: N nodes, IPs, prefixes
./start-native.sh --wan mynode1.conf      # from other nodes' cards
./start-native.sh --wan-nodes "1:IP:IND …" # scriptable
./start-native.sh --gen-conf              # write this node's card, start nothing
./start-native.sh --virtualbox=3          # mesh with VirtualBox VMs
./start-native.sh --air-mesh              # mesh the BURSTS: a shared radio medium,
                                          # a mobile can hear and pick another node's BTS
```

`--node N` picks the node number **at launch**, not at build time, so one ISO serves all nine nodes.

2.43.2.10 10. Other extensions

4G (srsRAN + Open5GS) — shipped in `/opt/LTE`, launched with `osmo-lte` and `osmo-epc`. CSFB to the 2G core through SGs.

Web dashboard — `osmo-egprs-web` on `:8080`, started by `systemctl restart osmo-egprs-web`.

postmarketOS handset UI — a patched `pmbootstrap` in `/opt/user_interface/pmos`.

2.43.2.11 11. Related documents

document	subject
`pont/README.md`	**A5 ciphering, Kc and SACCH** — a measurement report. Read it before chasing a radio fault through a CRC counter.
`wiki/`	Home, Build, Environment, Start-native, Results — using the bench day to day
`environment/README.md`	environment variables and path resolution
`services/README.md`	the systemd units shipped
`navigation/QUICKSTART.md`	getting started
`qosmo/README.md`	the QEMU fork: one base, two layer 1s

---

osmo-operator — a teaching and research GSM bench: a full core network and an emulated Calypso handset on one machine, with multi-operator SS7 interconnect when you want it.

2.44 /opt/GSM/osmo-operator/wiki/Resultats.md

4697 octets, 86 lignes

2.44.1 Reference results — `checks/check_all.sh`

Two runs on the same machine (Ubuntu 24.04, native, hostname `operator`), on 2026-09-15. They serve as a baseline: a healthy bench produces these lines, and the deviations commented here are known.

	Run 1 — 17:57	Run 2 — 18:12
Topology	1 native operator, **without** inter-STP	3 operators (Op1 native + Op2/Op3 containers) + local inter-STP
`global`	OK — 29 pass / 0 fail / 1 warn / 1 skip (85 s)	OK — 88 pass / 0 fail / 5 warn / 0 skip (217 s)
`ss7`	OK — 7 pass / 1 warn / 3 skip	OK — **31 pass / 0 warn / 0 skip**, full 3×3 matrix
`interstp`	skipped (no hub)	**ECHEC** (FAIL) “aucun AS actif” (no active AS) — false negative, see § 2.3
`operator`	ECHEC (FAIL) “INTERCO SS7 : DOWN” — expected without a hub	skipped (this node is the hub)
`annuaire`	skipped (no directory entry)	OK — 4 subscribers/op, 2 warn
`wan`, `dump`, `resume`	skipped	skipped
Verdict	**healthy** for a single node	**healthy**, interconnect operational; 1 diag bug, 2 warn to address

---

2.44.1.1 1. Run 1 — single native node

The whole core is green: STP 2 ASP / 2 AS, no PROHIB; HLR with VLR and SMSC over GSUP; MSC and BSC `ASP_ACTIVE`, SSN 254; BTS 0 Enabled/OK, OML + RSL; PCU on BTS; SGSN 1 NS entity; SIP connector, MNCC, SMS relay `:7890`, Asterisk; the seven daemons running.

`diag-stp-operator.sh` concludes “INTERCO SS7 : DOWN” with three causes: `asp-to-inter` in `shutdown`, `remote-ip 127.0.0.1` ≠ hub `172.20.0.10`, no SCTP. All three are the **same** cause: there is no inter-STP. The failure is expected and disappears at run 2.

Only one real warn: **GGSN, 0 APN**.

2.44.1.2 2. Run 2 — three operators + inter-STP

2.44.1.2.1 2.1 What proves the interconnect works

- Each STP has **3 ASP / 3 AS** active (the third is `as-inter`) and a default route to the inter-STP “présente et avail” (present and avail).
- `ss7_check.sh`: hub `0.0.0` reachable, 3 AS / 3 ASP, no PROHIB; on all three operators `as-inter → hub : ACTIVE`; connectivity matrix Op1/Op2/Op3 all `via`, 31 pass, 0 warn.
- `annuaire.sh`: operators 2 and 3 at **3/3 AS active**, point codes `1.2.x` / `1.3.x`, RCTX 250 / 350, hub 172.20.0.10 — matching the README’s formulas.

2.44.1.2.2 2.2 The warnings

Warn	Where	Interpretation
GGSN: 0 APN configured	Op1, Op2, Op3	same point as run 1: 2G data (SGSN → GGSN) does not bring up a PDP without an APN. `APN=internet` in `globals.conf` then `--regen`. 4G (Open5GS/ogstun) is not affected
SGSN: 0 NS entity	Op2, Op3	the containers' PCU did not open its NS-VC to the SGSN (Op1, native, has 1). No effect on voice/SMS/SS7; to look at together with the previous point if 2G data is wanted on these operators
2 subscribers without a directory entry	Op2, Op3	the HLR knows `00101000N000001/2` (provisioned by the old IMSI scheme) in addition to the `0010N000N000001/2` that have an entry. The script says so: bench restarted without `start-docker.sh`. Cosmetic

2.44.1.2.3 2.3 The false negative of `diag-interstp.sh`

The report shows `interstp ECHEC — INTERCO SS7 : DOWN - aucun AS actif` (FAIL — SS7 INTERCO: DOWN - no active AS), even though the same script has just written three `[ OK ] … AS_ACTIVE` lines and `ss7_check.sh` validates the interconnect end to end. Two clues in its output:

- `[FAIL] service inactif (inactive)` (service inactive): it queries `systemctl` for the hub's unit, which does not exist when the inter-STP is launched by `start-multi.sh` (the `osmo-inter-stp` container) and not by `osmo-interstp.service`.
- `== ASP ATTACHES ==` lists `asp-to-inter` (client to `127.0.0.1:2908`), `as-rkm-110` and `as-rkm-130`: these are the AS **of operator 1's STP** (RCTX 110 = MSC, 130 = BSC), not the hub's. The script read the VTY `127.0.0.1:4239`, which on a hybrid native + Docker node is Op1's, and found no `as-opN` — hence "aucun AS actif".

Conclusion: **the interconnect is UP**, the verdict is a defect in the script in a hybrid topology (hub in a container, operator 1 native on the same host). To fix in `checks/diag-interstp.sh`: target the VTY of the `osmo-inter-stp` container (or `--hub-vty`) when `OSMO_ROLE=operator` and the hub is "queryable here", and not count `systemctl` as a FAIL when the hub is a container.

2.44.1.3 3. Replay

```
bash ./checks/check_all.sh                        # ~2 min alone, ~6 min with three operators
./checks/check_all.sh --only=interstp --verbose   # the check in detail
./checks/check_all.sh --dump                      # + vty-debug-dump and operator_summary
```

Full log: `/tmp/osmo-check-all-<date>.txt`.

2.45 /opt/GSM/osmo-operator/wiki/SIM.md

7102 octets, 144 lignes

2.45.1 SIM: the bench's card in an Omnikey HID reader, and softSIM

The bench mobile (`mobile`, osmocom-bb) has no card reader: its SIM is either the **test SIM** from the config file (`sim test`, IMSI and Ki written in the clear, the default), or a SIM **served over a socket** by the SAP protocol (*SIM Access Profile*, `sim sap`). It is this second path that `start-native.sh --sap` sets up, with the SAP server from softSIM (Ruby), which can serve two things:

type	what the mobile sees	when
pcsc	the **real card** in the PC/SC reader: a sysmoISIM-SJA5 in a HID OMNIKEY 6121	a card is in the reader
sim	the **software SIM** `sim-op1.xml` (IMSI of MS#1, COMP128v1 with the HLR Ki)	no card

`--sap` chooses on its own (pcsc if a card is present, otherwise sim); `--sap=pcsc` (`--omnikey`) and `--sap=sim` force it. Without `--sap`, nothing changes: test SIM.

2.45.1.1 1. Install

```
sudo tools/osmo-sim-install.sh        # or: sudo ./addition.sh --sim
```

Installs, without breaking anything already present: `pcscd` + `libccid` (the PC/SC daemon and the Omnikey's CCID driver), `pcsc-tools`/`opensc` (to see the reader and the ATR), Ruby + `ruby-libxml` + the `smartcard` gem (softSIM as a reader), Osmocom's **pySim** in `/opt/GSM/pysim-venv` (from its git: the "pySim" on PyPI is a different project), and softSIM itself in `/opt/GSM/softsim` with the patches from `tools/softsim/`. The ISO does the same (`iso_modules/51-softsim.sh` + `80-chroot.sh`, `--no-softsim` to skip it).

Verified on 2026-10-14: OMNIKEY 6121 (USB `076b:6632`), sysmoISIM-SJA5, ATR `3b9f96801f878031e073fe211b674a357530350`
Ruby 3.2.3, smartcard gem 0.5.6, sysmo-usim-tool 190c95b.

2.45.1.2 2. View the card

```
tools/osmo-sim.sh lecteur     # the readers, the ATR, the card type, ICCID, IMSI
tools/osmo-sim.sh lire        # everything (pySim-read): PLMN, SMSP, ACC, services…
tools/osmo-sim.sh shell       # interactive pySim-shell (ADM1 passed if known)
```

2.45.1.3 3. Program the card to the bench's plan

The bench provisions its subscribers in osmo-hlr with a fixed plan (`qosmo/run_modules/21-abonnes-hlr.sh`, `start-native.sh` section `ms_imsi`):

	MS#1 of operator 1
IMSI	`001 01 0001 000001` → `001010001000001`
Ki	`00112233445566778899aabbccdd` + MS + operator → `…dd0101`
algorithm	COMP128v1 (`aud2g comp128v1`)
MSISDN	node×100000 + operator×100 + MS → `100101`

For the card to register, it must carry **exactly** this identity:

```
tools/osmo-sim.sh programmer --dry-run          # shows, writes nothing
tools/osmo-sim.sh programmer --adm1 12345678    # writes (asks for confirmation)
tools/osmo-sim.sh programmer --ms 2 --op 1      # another MS of the plan
tools/osmo-sim.sh verifier                      # card = plan? HLR? ready for --sap?
```

What it writes goes through sysmocom's tool, **sysmo-usim-tool** (`/opt/GSM/sysmo-usim-tool/sysmo-isim-tool.sja5.py`, installed by the installer): IMSI, MNC length, Ki (2G/3G/4G5G), **2G algorithm COMP128v1** (3G Milenage), OPc; then it reads back the algorithm and Ki (`-t -k`). It is the one that knows how to set the algorithm: `pySim-prog` writes only the Ki and the card would stay in Milenage, the HLR would answer *authentication failed*. Without the tool (or `OSMO_SIM_PYSIM_PROG=1`), it falls back to `pySim-prog` (ICCID kept, IMSI, PLMN, name, SMSP, MSISDN, Ki) then `pySim-shell` for the algorithm byte `0x11` in `DF.SYSTEM/EF.SIM_AUTH_KEY` (6f20) and `ADF.USIM/EF.USIM_AUTH_KEY_2G` (af22).

Done on 2026-10-14 on card 8949440000001549193: `verifier` says "prête" (ready).

2.45.1.3.1 The ADM1 key

Writing to a sysmoISIM requires the **ADM1 key of that card** (8 digits), which sysmocom provides with the card (data file of the order). **Three wrong tries block the card for good**: the script never guesses, it refuses without a key. Where to put it, in the order it is looked for:

1. `--adm1 K` on the command line;
2. `OSMO_SIM_ADM1=K` in the environment;
3. `/etc/osmocom/sim-adm1` (`chmod 600`): one line, or `ICCID key` per line for several cards;
4. `~/.config/osmo/sim-adm1`.

2.45.1.3.2 A card you don't want to reprogram

It keeps its IMSI and Ki: you must then declare them to the HLR.

```
tools/osmo-sim.sh hlr --creer --ki <card's Ki>     # subscriber created via the VTY (4258)
```

(`hlr` without `--creer` only says whether the HLR knows the IMSI read on the card.)

2.45.1.4 4. Launch

```
./start-native.sh --sap            # card present → pcsc; otherwise software SIM
./start-native.sh --sap=pcsc       # the card, mandatory
./start-native.sh --dry-run --sap  # what would be done, without launching anything
```

`start_sap` runs **before** the stack: it launches `demo_server.rb` (softSIM) on `/tmp/osmocom_sap` (`SAP_SOCK`), with log `/tmp/osmo-sap.log` (`SAP_LOG`, `SAP_VERBOSITY=3`: connection, card ready, each timestamped APDU), and the mobile connects to it at startup. The server serves the next client after a disconnect (patch `0002`): a baseband restart requires nothing.

In DSP mode (the default), the MS#1 that runs is the one from c54x_exe (`mobile_pont.cfg`, `sim test`): `--sap` derives `/root/.osmocom/bb/mobile_pont_sap.cfg` from it (`sim sap`, `sap-socket`) and passes it to the bench via `MOBILE_CFG`. In grgsm mode, it is `mobile.cfg` generated from `configs/mobile.cfg` (`sim sap` in the template, reset to `sim test` without `--sap`) that carries it; MS#2 stays on the test SIM.

`./start-native.sh --stop` also stops the SAP server; a startup always relaunches it fresh (it is killed with the rest in "Arrêt de la pile avant démarrage" — stopping the stack before startup).

Serve the card by hand, without the stack (to watch the APDUs):

```
tools/osmo-sim.sh sap              # foreground, Ctrl-C to stop
```

file	what

2.45.1.5 5. What the patches fix

file	what
`tools/softsim/0001-simos-comp128v1-a38.patch`	software SIM: COMP128v1 with `SOFTSIM_KI` (any RAND from the HLR), READ RECORD, `--file` passed through, acceptance loop
`tools/softsim/0002-sap-serve-next-client.patch`	the server serves the next client instead of exiting
`tools/softsim/0003-sap-log-timestamps.patch`	APDUs named and timestamped (ms) in the log
`tools/softsim/0004-pcsc-disconnect-idempotent.patch`	pcsc mode: `disconnect` called twice per client (DISCONNECT_REQ then socket close) crashed the server on a null handle, the mobile could no longer reconnect
`tools/softsim/ruby_compat.rb`	Ruby 3.2: `File.exists?`, `Fixnum`
`tools/softsim/make_sim_op1.py`	`sim-op1.xml`: IMSI, valid ICCID, PLMN, SPN from the demo `sim.xml`
`patches/osmocom-bb-plmn-a6-reinsert.patch`	mobile: SIM reinserted without RPLMN (SAP reconnection), the PLMN search ended “unhandled”
`patches/osmocom-bb-sim-plmn-from-bcd.patch`	mobile: SIM PLMNsel/FPLMN read with the right function (000-00 before)

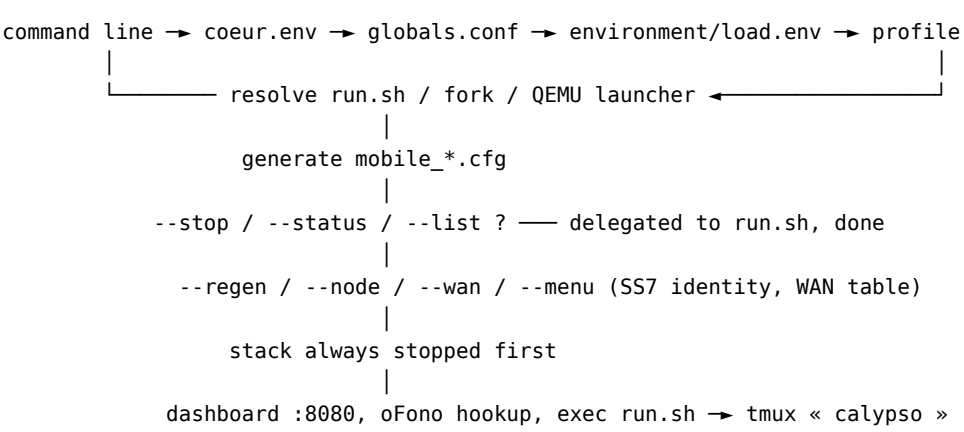
The two osmocom-bb patches are applied by `install_modules/40-patches.sh` (native) and the `Dockerfile`; `mobile` must be recompiled afterwards.

2.46 /opt/GSM/osmo-operator/wiki/Start-native.md

13624 octets, 273 lignes

2.46.1 `start-native.sh` — step by step

`start-native.sh` prepares, `run.sh` executes. This document follows the script phase by phase: what it reads, what it decides, what it writes, and at what point your variable is taken into account — or overwritten.



2.46.1.1 0. Where we are, and a removed safeguard

The script moves into its own directory (`HERE`). It **no longer** checks that it runs inside Docker (check removed on 2026-08-14: the ISO has no Docker and this was the only usable launcher). Known consequence: started by mistake on a Docker host, it dies on « `run.sh` introuvable » (run.sh not found), which does not point to the real cause. On a host with Docker, the launcher is `./launch/start-docker.sh`.

2.46.1.2 1. Loading the environment

In this order, and **the order decides who wins** (details: Environnement.md):

1. **`/etc/osmocom/coeur.env`** (`OSMOCOM_CFG` to move it) — written by `build-iso.sh`, carries `N_MS`. Loaded here and not by `~/.bashrc`, otherwise the icon, systemd and `start-multi.sh` did not see it: the HLR only had 100101 and the second mobile got created on the fly without a Ki, unreachable. The file uses `:=`: an `N_MS=3` set in the environment wins.
2. **Command-line options** (`--profile`, `--dsp`, `--node…`).
3. **`globals.conf`** — the reference for its 26 network variables (MCC, MNC, OP_NAME, ENCRYPTION, ARFCN, BAND, LAC, APN, MS_COUNT, N_OPERATORS, HOST_IP…), **except those the caller has already set to a non-empty value** (since 2026-09-22): `start-native.sh` records them, reads the file, then sets them again, and prints « globals.conf surchargé par l'environnement : … » (globals.conf overridden by the environment). `ENCRYPTION="a5 0" ./start-native.sh --dsp` does give a5 0. An empty value overrides nothing (empty = computed, often from the operator number). Everything else passes through untouched. To change: `vim globals.conf` or `./generate_configs.sh ARFCN=520`.
4. **`environment/load.env`** — `modes.env` (the profile), `paths.env` (where things are), then the per-domain files. All with `:=`: the first one to set a value wins, hence the command line.
5. Minimal fallback if `load.env` is missing (bare ISO, bare container): `GSM_ROOT=/opt/GSM`, `CALYPSO_FORK=qosmo-grgsm`, resolution of `OQC_ROOT`.
6. Remaining defaults: `RUN_DIR=/run/osmo-direct`, `LOG_DIR=$RUN_DIR/logs` (tmpfs — tmux scrollback depends on it), `ENCRYPTION="a5 1"` (fallback: the real value comes from `globals.conf`), `CALYPSO_BRIDGE=pont` (`none` with `--dsp`), `MS_COUNT=2`, `HOST_IP=127.0.0.1`.

2.46.1.3 2. Profile

Default: **`faketrx-qemu`** (alias `hybrid`) — core + BTS#0 QEMU + BTS#1 faketrx. `--profile <nom>` or the positional mode sets it; `CALYPSO_PROFILE` or the legacy `MODE` set it from the environment (`MODE` wins if set). `--dsp` no longer touches the profile (since 2026-09-22): it keeps the default `faketrx-qemu`, side-car included. An explicitly named profile always wins.

Profile	What runs
`faketrx-qemu` / `hybrid`	core + osmo-bts-trx ← QEMU Calypso + fake_trx/trxcon/mobile
`faketrx`	core + fake_trx + trxcon + mobile; no QEMU
`virtphy`	core + osmo-bts-virtual + virtphy + mobile
`qemu` / `calypso`	the Calypso pipeline alone, core in no-process
`core` / `noproc`	STP + HLR + MSC + MGW + BSC (+ Asterisk), no radio
`hw`	physical SDR
`virtual`	SS7 multi-operator via `ip netns` (`N_OPERATORS`)

2.46.1.4 3. Resolving `run.sh` and the QEMU launcher

Order: explicit `RUN_SH` > resolved `OQC_ROOT` > `$GSM_ROOT/<fork>/run.sh`. `OQC_ROOT` is looked up **next to the repository** (`../qosmo-grgsm`) then under `/opt/GSM`. If `run.sh` is not executable: FAIL, with the path tried.

The C launcher (`--launcher <bin>`, `QOSMO_LAUNCHER`, default `/usr/local/bin/<fork>`) is compiled if missing and `<fork>/tools/qosmo-launch` is present. Without a launcher, `40-qemu.sh` falls back to the legacy `qemu-system-arm` command line.

2.46.1.5 4. Path validation

`--check-paths` stops here after checking every declared dependency (Osmocom binaries, `layer1.highram.elf` firmware, `calypso_dsp.txt` ROM, bridge).

2.46.1.6 5. Generating `mobile_*.cfg`

`mobile_group1.cfg` (MS#1, behind QEMU) and MS#2 (faketrx) are written to `$OQC_ROOT/cfgs/` from the templates, with IMSI/Ki/IMEI from `globals.conf` (or computed per operator). The layer 2 client is `mobile` by default; `--ccch_scan`, `--bcch_scan`, `--cell_log` **replace** it (they take its L1CTL socket and its VTY — mutually exclusive).

2.46.1.7 6. Summary

The plan is printed: profile, fork, `run.sh`, launcher, bridge, number of MSs, ciphering, ARFCN. `--list` stops there (delegated to `run.sh --list`); `--dry-run` continues without side effects.

2.46.1.8 7. Delegated actions

`--stop` and `--status` are passed to `run.sh` and the script exits.

2.46.1.8.1 7-menu. `--menu`

Asks **once** for the choices the script would otherwise have guessed: profile, node, operator, inter-STP, WAN, regeneration. The proposed values are those of the automatic launch: accepting without changing anything = not passing the option. Ignored without a terminal.

2.46.1.8.2 7-regen. `--regen`

By default the existing configs are **kept**: `run.sh` would regenerate them at startup and erase the SS7 identity set just before. `--regen` stops the stack, then rewrites everything from the templates (current node and operator).

2.46.1.8.3 7-node. `--node N` / `--op N` / `--hub-ip`

The node number (1 to 9) is **deduced**: environment, then `/etc/osmo-role`, then the WAN table compared against the local addresses. `--node` forces it. The script rewrites the SS7 identity (point codes `1.<nœud><op>.<rôle>`, routing contexts) and points the ASP at the inter-STP (`--hub-ip`, otherwise deduced: docker or VM). This is what makes a single ISO possible for all nine nodes.

2.46.1.8.4 7-wan. `--wan`

Never on by default. A `--wan` on the command line applies the WAN **for that run only**: the table it writes keeps `WAN_AUTO=0`, so the next plain start has no WAN. Only an ISO built with `--wan` sets `WAN_AUTO=1` in `/etc/osmo-wan.conf`, which brings the WAN up at every start (and stays 1 when the table is rewritten). To make a machine a permanent node by hand, set `WAN_AUTO=1` in that file; to stop it, set `WAN_AUTO=0`. Three forms:

```
./start-native.sh --wan                                  # questions: nodes, IPs, prefixes, my number
./start-native.sh --wan-nodes "1:IP:IND 2:IP:IND" --wan-id 2   # scriptable
./start-native.sh --wan mynode1.conf --wan mynode3.conf  # from the other nodes' sheets
./start-native.sh --gen-conf [FICHIER]                   # writes MY sheet, launches nothing
```

`--regen-conf` rebuilds the table from scratch (the old one is backed up next to it): use it when a powered-off peer leaves a PROHIB route on the STP. Full procedure for two nodes, LAN or Internet: WAN.

The table (`--wan-conf`, default `/etc/osmo-wan.conf`) is applied, the SS7 addressing towards the inter-STP set up, the SIP + SMS mesh generated. Dialing `<indicatif><numéro>` reaches that number on the matching node. The profile stays `faketrx-qemu`.

`--air-mesh[=PORT]` additionally meshes the **bursts**: the radio medium becomes shared, a mobile can hear and pick another node's BTS (peers from the WAN table, range in `data/air-mesh.txt`, two extra mobiles — `pont/airmesh.py`).

`--virtualbox[=N]` sets up the segment and N−1 VMs before the WAN; refuses to run **inside** a VM (`--vbox-node N` = this machine's number).

2.46.1.8.5 7-tcpdump

A tcpdump wrapper is installed for GSMTAP (`udp/4729`) and M3UA captures.

2.46.1.8.6 7-stop

The stack is always stopped before starting (`run.sh --stop`). There is no « hot restart »: `--force` only restarts modules that are already running instead of skipping them.

2.46.1.9 8. Launch

1. The node's private addresses are printed.
2. **Dashboard** on `:8080` (Firefox opens it from `launch.sh`); the I/Q spectra on `:8081` (`fft-web/fft_web.py`).
3. **oFono mobile hookup** — the physical phone (postmarketOS, PPP) or the Android QEMU sees the bench as a modem.
4. **Audio (slimmed-down PulseAudio)** — runs `scripts/audio-chain.sh 5` (log `/tmp/osmo-audio-chain.log`): stops speech-dispatcher, switches `osmo_rec` to 8 kHz mono, loads `module-stream-restore` without `restore_device`. `AUDIO_ALLEGER=0`: don't slim down (the chain is set up anyway). `AUDIO_AEC_METHOD`: `webrtc` by default (`speex` possible). `AUDIO_PENDANT_APPEL=1`: the heavy chain is only set up between gapk-auto's « Endpoint actif » and « Endpoint disparu » (endpoint active / gone) (default 0). `audio-chain.sh --reinit`: start again from a clean chain without restarting PulseAudio.
5. **Hand-off to `run.sh`**: `exec`, with the profile and the whole environment. Any `CALYPSO_*` variable passed as a prefix reaches `run.sh`, its `run_modules/` and QEMU **untouched** (`CALYPSO_LANG=en`, `CALYPSO_NO_ATTACH=1`, `CALYPSO_BRIDGE=none`…).

`run.sh` opens the `calypso` tmux: `osmo-bts-trx` → QEMU (PTY + monitor) → wait for `/tmp/osmocom_l2` → bridge → wait for UDP 6700 ticks → `mobile`, then the core, Asterisk, SMSC. Unless `--no-attach` / `CALYPSO_NO_ATTACH=1` (which `osmo-banc.service` sets: nobody is sitting in front of it), the script attaches to the tmux.

2.46.1.9.1 8-dsp. The DSP bench

With `--dsp`, `RUN_SH` stays the fork's one and receives `--skip qemu,pty,osmocon,l2 --no-attach`. Then `exec c54x_exe/run.sh` (`BANC_DSP`, if executable) takes over with `MODE=dsp PONT=1`, `IQ=none`, `INSNS=120000`, `LOCKSTEP=1`, `CALYPSO_BSP_STREAM=1`, `CALYPSO_RHEA_DMA_XFER=1` (the latter is no longer read by the core, see Environnement.md) — set with `:=`, so they can be overridden. The five steps:

1. `c54x_exe --arm` — the DSP: socket `/tmp/calypso_dsp.sock`, shared RAM API, DL bursts on `udp/6702`;
2. `qemu-system-arm` from `/opt/GSM/qosmo` (`CALYPSO_DSP_EXTERN=1`) — the ARM alone;
3. `osmocon`;
4. `mobile` (`mobile_pont.cfg`, VTY **4347**);
5. `pont/pont_dsp.py`.

`CALYPSO_BRIDGE` is `none`: it is not `start-native.sh` that starts the bridge. `--stop`, `--regen` and the stop before starting also stop the DSP bench, even without `--dsp`; `--status` and `--check-paths` show it (with `--dsp`). The summary and oFono target VTY 4347 (`OSMO_MOB_VTY_PORT`). `BANC_DSP=none`: the stack alone.

Logs: the real files are at the dashboard paths (`/run/user/0/osmo-nitb/logs/{qemu,osmocon,mobile}.log`, `/dev/shm/pont.log`), with links in `/tmp/c54x-pont`. `--assembly-logs` (`ASSEMBLY_LOGS=1`, `ASSEMBLY_LOGS_FLAGS`, `ASSEMBLY_LOGS_FILTRE`) writes `/tmp/c54x-pont/qemu-asm.log`, but the bench then loses real time.

2.46.1.10 Short manual

```
./start-native.sh [options] [mode]

Modes: faketrx-qemu (default) | faketrx | qemu | noproc | core | hybrid

Stack selection
  --dsp           (default since 2026-10-01) C54x DSP bench: the C54x runs outside QEMU (/opt/GSM/c54x_exe, TI mask ROM)
                        and decodes FCCH/SCH/BCCH itself; the mobile camps (SI1-4, lai=001-01-1)
```

```
                      and does its location update. Same profile and same stack as
                      grgsm, only qemu,pty,osmocon,l2 move to c54x_exe/run.sh
                      (BANC_DSP=none: the stack alone)
  --assembly-logs     ARM asm trace (/tmp/c54x-pont/qemu-asm.log); no more real time
  --grgsm             gr-gsm layer 1 inside qosmo (the old default)
  --sap, --softsim    SIM served over SAP (softSIM): the card in the PC/SC reader
                      (HID Omnikey) if one is inserted, otherwise the software SIM;
                      --sap=pcsc / --sap=sim force it, --no-sap (default): test SIM.
                      See [SIM](SIM.md).
  --launcher <bin>    QEMU C launcher (default /usr/local/bin/<fork>)
  --profile <nom>     forces the profile
  --mobile | --ccch_scan | --bcch_scan | --cell_log   layer 2 client (mutually exclusive)

Lifecycle
  --list  --dry-run  --stop  --status  --force  --verbose  --no-attach  --check-paths
  --regen             regenerates the configs from the templates (kept otherwise)
  --menu              asks the questions instead of guessing

Topology
  --node N  --op N  --hub-ip IP
  --wan | --wan FICHE… | --wan-nodes "…" | --wan-id N | --wan-conf FICHIER | --gen-conf [F]
  --air-mesh[=PORT]  --virtualbox[=N]  --vbox-node N
```

Getting back in:

```
tmux attach -t calypso         # Ctrl-b d detaches, the bench keeps running
journalctl -u osmo-banc -f     # if started as a service
./start-native.sh --status
./start-native.sh --stop
```

Logs: `$LOG_DIR` (`/run/osmo-direct/logs`, tmpfs); QEMU: `/var/log/osmocom/qemu.log`, bridge: `/var/log/osmocom/bridge.log`, orchestration: `/var/log/osmocom/run.sh.log`.

2.47 /opt/GSM/osmo-operator/wiki/Telephone-emule.md

10768 octets, 162 lignes

2.47.1 Installing the emulated phone on your own Osmocom stack

> Version of 2026-10-03. Each command comes from a script or a README in the repos, with the source cited. Anything that could not be verified is marked **TO CONFIRM**. The `Dockerfile:N` numbers refer to the Dockerfile at commit `4266c8b` (before it started calling the repos' installers).

This page is for anyone who **already** has an Osmocom network (osmo-bsc, osmo-msc, osmo-hlr, osmo-stp, osmo-mgw, installed from packages or from source) and wants to connect an **emulated Calypso phone** to it, without the ISO, without Docker and without the osmo-operator stack.

Page	Contents
Telephone-emule (here)	overview, requirements for **your** network, installation order
Installer-qosmo	QEMU, the Calypso machine (the ARM + osmocom-bb firmware)
Installer-c54x_exe	the C54x DSP outside QEMU, its TI ROM, the `pont_dsp.py` bridge
Installer-grgsm_exe	the gr-gsm setup, and what "grgsm_exe" really means
Lancer-a-la-main	the chain launched process by process, with commands unfolded

Page	Contents
installer-osmo-jusquau-dsp.ipynb	notebook: the whole Dockerfile replayed natively, from apt down to a call in the DSP

2.47.1.1 What runs on the phone side

```
your network                                  what you install here
─────────────                                 ──────────────────────────────────────────

osmo-bsc / msc / hlr / stp / mgw
       | Abis/IPA
osmo-bts-trx ── TRXD/TRXC/CLK UDP 5700-5702 ──► pont_dsp.py ── UDP 6702 ──► c54x_exe (DSP + TI ROM)
(5800-5802)                                    (osmo-operator/pont)          | /dev/shm/calypso_api_ram
                                                                             | /tmp/calypso_dsp.sock
                                  osmocon ◄── serial pty ── qosmo (QEMU -M calypso, CALYPSO_DSP_EXTERN=1)
                                          | /tmp/osmocom_l2          └─ layer1.highram.elf (osmocom-bb)
                                  mobile (osmocom-bb, VTY 4347)
```

Diagram taken from `c54x_exe/README.md:25-35`. Two setups exist (`c54x_exe/LAUNCH.md:6-9`):

setup	layer 1	bridge	state (2026-10-03)
dsp (default since 2026-10-01)	`c54x_exe --arm`, real TI mask-ROM	`pont/pont_dsp.py --dsp-port 6702`	camps, LU, SMS MO/MT, MO/MT calls, A5/1 (`c54x_exe/README.md:11-21`); SB sync sometimes takes several tries
grgsm	gr-gsm compiled into QEMU	`pont/pont.py`	the historical chain, "goes all the way to the voice call" (`wiki/Home.md`, options)

2.47.1.2 What your network must provide

The phone is pinned to a specific cell and subscriber. Either your network conforms to it, or you change the values on the phone side (right-hand column).

Element	Expected	Where to change it on the phone side
BTS	**osmo-bts-trx** on the same machine (tested: osmo-bts 1.10.0, `osmo-operator/Dockerfile:272`)	—
TRX link	BTS config: `phy 0` / `osmotrx ip local 127.0.0.1` / `osmotrx ip remote 127.0.0.1`, **without** `base-port` (defaults 5800 local, 5700 remote) — `osmo-operator/configs/osmo-bts-trx.cfg:28-34`	`PONT_TRX_BIND`, `PONT_TRX_BASE` (`pont/config.py:48-49`)
Band, ARFCN	DCS1800, ARFCN **514**	`PONT_ARFCN` + the `stick 514` line of `c54x_exe/mobile_pont.cfg`
BSIC	**7**	`PONT_BSIC` (`pont/config.py:47`)
Timeslots of the BSC's first `bts`	TS0 `CCCH+SDCCH4`, TS1 `SDCCH8`, TS2-7 `TCH/F` (`osmo-operator/configs/osmo-bsc.cfg:236-263`)	the bridge **reads** them in `PONT_BSC_CFG` (default `/etc/osmocom/osmo-bsc.cfg`, `pont/gsm.py:223-242`); if the file is unreadable → the table at left by default

Element	Expected	Where to change it on the phone side
PLMN	MCC 001, MNC 01	`rplmn 001 01` in `mobile_pont.cfg`
Subscriber	IMSI `001010001000001`, COMP128v1, Ki `00112233445566778899aabbccdd0101`	`test-sim` in `mobile_pont.cfg`

Provision the subscriber in osmo-hlr (same VTY commands as `qosmo/run_modules/21-abonnes-hlr.sh:73-75`):

```
telnet 127.0.0.1 4258
enable
subscriber imsi 001010001000001 create
subscriber imsi 001010001000001 update msisdn 100101
subscriber imsi 001010001000001 update aud2g comp128v1 ki 00112233445566778899aabbccdd0101
```

The MSISDN `100101` is the bench's (`wiki/Home.md`, numbering plan); use your own.

Ports and files the phone occupies on the machine (do not use them in your stack): UDP 5700-5702 (bridge), UDP 6702 (DSP), UDP 4730/4731 (gr-gsm L1, grgsm setup), TCP 4347 (mobile VTY), TCP 1234 (QEMU gdbstub), TCP 44444 (gdb console), `/tmp/osmocom_l2`, `/tmp/osmocom_sap`, `/tmp/ms_data`, `/tmp/calypso_dsp.sock`, `/tmp/qemu-monitor-pont.sock`, `/dev/shm/calypso_*`, `/dev/shm/pont.log`, `/tmp/c54x-pont/` (sources: `c54x_exe/run.sh:42-62`, `c54x_exe/mobile_pont.cfg`, `pont/config.py`).

2.47.1.3 Installation order

Everything goes under a common root, `/opt/GSM` by default (paths hard-coded in the Makefiles and `run.sh`, all overridable: `GSM_ROOT=`, `QOSMO=`, `FIRMWARE_ELF=`, `OSMOCON=`…).

Each repo carries its own installer (`install.sh`, since 2026-10-03), with the same contract as `osmo-operator/install_modules/_lib/inst.sh`: `--check` (prerequisites, changes nothing), `--list`, `--deps` / `--print-deps` (apt packages, taken from the Dockerfile's list), `--only` / `--skip`, a log per step, a post-check. These are **the same scripts** the Dockerfile (stages `qemu` and `l1`), `Dockerfile.run`, `start-docker.sh` and the ISO call: a single list of commands per component.

2.47.1.3.1 In one pass, from osmo-operator

```
git clone https://github.com/bbaranoff/osmo-operator /opt/GSM/osmo-operator
cd /opt/GSM/osmo-operator
./install.sh --list --telephone       # what will be done
sudo ./install.sh --telephone         # firmware, qosmo, c54x_exe, grgsm_exe, osmocom-bb (mobile, osmocon)
./install.sh --check --telephone
```

`--telephone` = the steps `prereqs, sources, calypso, build, binaires, verify`: neither the global apt list (each Calypso component installs its own packages), nor the core patches (`40-patches` targets osmo-bts/msc/hlr/trx), nor `/etc/osmocom`, nor the desktop, nor 4G; `verify` (`OSMO_CORE=externe`) no longer looks for the core daemons, only **your** `osmo-bts-trx` and the Calypso chain (qemu-system-arm, c54x_exe, ROM, firmware, osmocon, mobile). The build dependencies of osmocom-bb / osmo-gapk (libosmocore-dev, libtalloc-dev, libasound2-dev…) are assumed present — otherwise `sudo ./install.sh --only deps` (the full Dockerfile list). The Calypso group alone: `sudo ./install.sh --only calypso` (`install_modules/45-calypso.sh`).

2.47.1.3.2 Component by component

1. **System dependencies** — each installer has its list: `./install.sh --print-deps`, `sudo ./install.sh --deps` (subsets of `osmo-operator/Dockerfile:156-205`).
2. **libosmocore** (with libosmocoding) — you already have it if your stack is compiled; otherwise the `libosmocore-dev` packages from the Osmocom repo (**TO CONFIRM**: versions; the bench uses 1.12.1, `Dockerfile:261`). `c54x_exe/install.sh --check` says right away whether pkg-config sees it.
3. **osmo-gapk** then **osmocom-bb** (osmocon, mobile) and the prebuilt **firmware** — see Installer-qosmo.
4. **qosmo** — `qosmo/install.sh`, Installer-qosmo.

5. **c54x_exe + DSP ROM + bridge** — `c54x_exe/install.sh`, Installer-c54x_exe.
6. (optional) **gr-gsm / grgsm_exe setup** — `grgsm_exe/install.sh`, Installer-grgsm_exe.
7. **Launch**: `PONT=1 ./run.sh` in `c54x_exe` (`c54x_exe/README.md:76-78`) or by hand, Lancer-a-la-main.

```
cd /opt/GSM
git clone https://github.com/bbaranoff/qosm0    qosmo     && sudo qosmo/install.sh --with-deps
git clone https://github.com/bbaranoff/c54x_exe c54x_exe  && sudo c54x_exe/install.sh --with-deps
git clone https://github.com/bbaranoff/grgsm_exE grgsm_exe && grgsm_exe/install.sh      # optional
git clone --depth 1 https://github.com/bbaranoff/firmware firmware
```

2.47.1.4 Proof that it works

In order (details in Lancer-a-la-main):

Step	Expected line	File
DSP ready	`pont : en attente de l'ARM sur /tmp/calypso_dsp.sock` (bridge: waiting for the ARM)	`/tmp/c54x-pont/dsp.log`
QEMU connected to the DSP	`[trx] pont DSP : API RAM partagee ...` (shared API RAM)	`/tmp/c54x-pont/qemu.log`
firmware loaded	`Received branch ack, your code is running now!`	`/tmp/c54x-pont/osmocon.log`
bridge ready	`pont TRX : ports 5700/5701/5702, ARFCN 514, BSIC 7`	`/tmp/c54x-pont/pont.log` (link to `/dev/shm/pont.log`)
BTS seen by the bridge	`BTS 127.0.0.1 : horloge armee vers le port 5800` (clock armed toward port 5800) (`pont/trx.py:185`)	same
cell read	`New SYSTEM INFORMATION ..., lai=001-01-<LAC>` (`c54x_exe/run_real.sh`)	`/tmp/c54x-pont/mobile.log`
camp	`C3 camped normally` in `show ms` (`tests/modules/60-camp.sh:10`)	VTY 4347
LU	`LOCATION UPDATING ACCEPT` (`run_real.sh`); `MM idle, normal service`	`mobile.log` / `show ms`
call	`call 1 600` then `call control state: ACTIVE` (`tests/modules/_lib.sh:48,83`)	VTY 4347

`c54x_exe/run_real.sh` automates these checks against a stack launched by **systemd** (`systemctl start osmo-hlr osmo-stp osmo-msc osmo-mgw osmo-bsc`, `systemctl restart osmo-bts-trx`) and prints a verdict `VRAI : lai=… = la config du reseau` (TRUE: lai=… = the network config). It reads MCC/MNC from `/etc/osmocom/osmo-msc.cfg` and LAC/BSIC from `/etc/osmocom/osmo-bsc.cfg`: adapt if your configs are elsewhere.

2.47.1.5 TO CONFIRM (overall)

- Compatibility with an osmo-bts-trx **other than 1.10.0** (TRXD negotiation: the bridge answers `SETFORMAT` with an echo, `pont/trx.py:182`, and only reads TRXD v0, `trx.py:199`).
- The bench's `osmo-bts/hlr/msc-force-rand-toy` patches (`Dockerfile:358-414`) are **not** needed by the phone (they make the RAND reproducible) — not tested without them.
- An osmo-bts-trx **on a different machine** from the phone: not planned for (the bridge returns the clock to the source IP of the CTRL command, `trx.py:184`, it might work with `PONT_TRX_BIND=0.0.0.0`).
- Audio during a call: `mobile_pont.cfg` uses ALSA `gsm_out`/`gsm_in`, defined by `osmo-operator/configs/asound.conf` (the `libasound2-plugins` plugin + PulseAudio). Without them, the mobile's behavior during a call is unverified.

2.48 /opt/GSM/osmo-operator/wiki/WAN.md

10217 octets, 233 lignes

2.48.1 WAN — linking two nodes

Two machines, each running its own stack (`start-native.sh`, one operator per node), linked so that the mobile on node 1 can call and text the mobile on node 2. The method works the same on a LAN and over the Internet. Over the Internet it goes through a **WireGuard** tunnel (§2), which is the only part that changes.

2.48.1.1 1. What crosses between the nodes

Flow	Protocol / port	Who listens	Used for
SS7 (M3UA)	**SCTP 2908**	the inter-STP (the hub), one for the whole WAN	SS7 interconnect, point codes `1.<node><op>.<role>`
Voice signalling	UDP/TCP **5060** (single-operator node); 5080, 5082… (N-operator node)	Asterisk	PJSIP trunks between nodes
Voice media	UDP **20000–20499** (op 1)	Asterisk	RTP
SMS	TCP **7890** (op 1; 789x for op x)	`sms-interop-relay.py`	SMS between nodes

Numbering works like a country code. Node N has prefix `NN` (11, 22…), and a mobile's MSISDN is `<node>00<op><ms>`. The first mobile of node 2 is therefore `20011`. From node 1 you reach it by dialling `22` + `20011` = **`2220011`**. The sending node strips the prefix before going out on the WAN.

2.48.1.2 2. Why a tunnel over the Internet

Two nodes on the same LAN (or on the VirtualBox host-only segment) reach each other directly: skip to §4. Over the Internet, without a tunnel, four things break:

1. **SCTP does not cross the NAT** of a consumer router. The SS7 link to the hub never comes up.
2. The address in the WAN table must be **both** reachable by the peer **and** held by a local interface. The node uses it to recognise itself (`wan_nodes_detect_self`), and the SCTP ASP binds to it (`local-ip`). A NATed public IP cannot be both.
3. Asterisk's SDP announces `external_media_address` = the table address. Behind NAT, RTP goes to a private address.
4. Ports 2908/5060/789x/RTP would be exposed to the whole Internet.

With WireGuard, each node holds a private address that is **local and routable** (`10.99.0.N`). SCTP travels inside the tunnel's UDP, and the WAN case becomes the LAN case.

2.48.1.3 3. Step 1 — the WireGuard tunnel

Example plan:

Node	Machine	Tunnel address	Extra role
1	A	`10.99.0.1`	runs the inter-STP (co-located hub)
2	B	`10.99.0.2`	—

At least one side must be reachable on UDP 51820: public IP, or a port forward on the router. Here, that is A.

On each machine:

```
sudo apt install wireguard
wg genkey | sudo tee /etc/wireguard/priv | wg pubkey | sudo tee /etc/wireguard/pub
sudo chmod 600 /etc/wireguard/priv
```

`/etc/wireguard/wg0.conf` on **A**:

```
[Interface]
Address    = 10.99.0.1/24
ListenPort = 51820
PrivateKey = <contents of /etc/wireguard/priv on A>

[Peer]                      # node 2
PublicKey  = <pub of B>
AllowedIPs = 10.99.0.2/32
```

`/etc/wireguard/wg0.conf` on **B**:

```
[Interface]
Address    = 10.99.0.2/24
PrivateKey = <contents of /etc/wireguard/priv on B>

[Peer]                      # node 1 (+ hub)
PublicKey  = <pub of A>
Endpoint   = <public IP or DNS name of A>:51820
AllowedIPs = 10.99.0.1/32
PersistentKeepalive = 25    # keeps the NAT mapping open on B's side
```

```
sudo systemctl enable --now wg-quick@wg0
```

> **Do not use a `192.168.x` range** for the tunnel. With no WAN table yet, `wan_detect_local_ip` prefers the first `192.168.*` address, which is the LAN one. With `10.99.0.0/24` there is no ambiguity. The command in §5 also gives the addresses explicitly.

2.48.1.4 4. Step 2 — check the link and open the firewall

```
ping -c3 10.99.0.2          # from A
ping -c3 10.99.0.1          # from B
sudo wg show                # a recent "latest handshake" on both sides
```

Firewall. `setup-wan-mesh.sh` calls `network/firewall-wan.sh`, which opens **neither SCTP 2908** (the hub) **nor 5060** (direct SIP to a single-operator node: only 5080 and up are opened). With an `INPUT` policy of ACCEPT (the iptables default) this has no effect. With `ufw` active, SS7 and voice are dropped. On the tunnel, the simplest fix is to trust the interface, on **both** machines:

```
# ufw
sudo ufw allow in on wg0 from 10.99.0.0/24
# or iptables
sudo iptables -C INPUT -i wg0 -s 10.99.0.0/24 -j ACCEPT 2>/dev/null \
  || sudo iptables -I INPUT -i wg0 -s 10.99.0.0/24 -j ACCEPT
```

On a LAN without a tunnel, open the ports by hand, from the peer's IP: SCTP 2908 on the hub, then UDP/TCP 5060, UDP 20000–20499 and TCP 7890 on every node.

2.48.1.5 5. Step 3 — start the nodes (node 1 first)

Both nodes get the same table. Each one gets its own number, and all of them point at the same hub. This is the scriptable form, with no questions and no detection, and it is better than node cards for a first setup (see §7).

A — node 1, runs the hub:

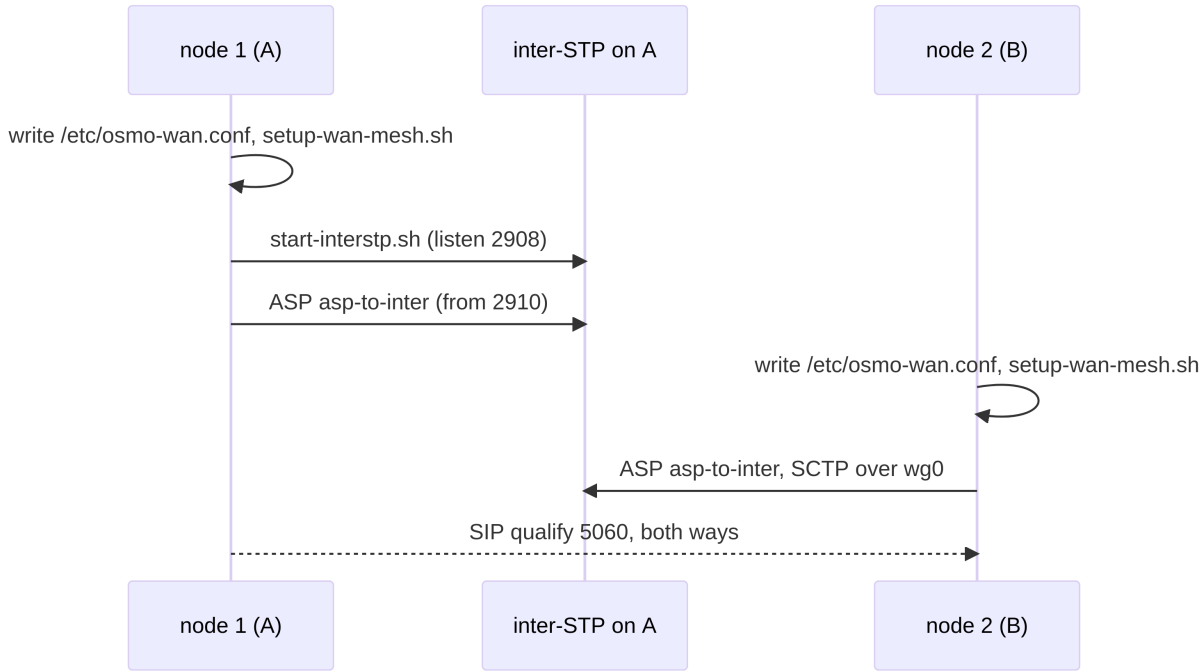
```
sudo ./start-native.sh --regen-conf \
     --wan-nodes "1:10.99.0.1:11 2:10.99.0.2:22" --wan-id 1 --node 1 \
     --hub-ip 10.99.0.1 --interstp
```

B — node 2, once A shows `Inter-STP … OK`:

```
sudo ./start-native.sh --regen-conf \
     --wan-nodes "1:10.99.0.1:11 2:10.99.0.2:22" --wan-id 2 --node 2 \
     --hub-ip 10.99.0.1 --no-interstp
```

- `--wan-nodes` applies the WAN **for this run only**: the next plain `./start-native.sh` starts without it. Pass the same options again (or set `WAN_AUTO=1` in `/etc/osmo-wan.conf`) for the node to mesh at every start.
- `--regen-conf` starts from an empty table. The old one is saved as `/etc/osmo-wan.conf.avant-regen-<date>`. Without this option, a peer seen once and then switched off stays in the table and leaves a **PROHIB** route at the STP. On the very first start the option is not needed, but it is harmless.
- The hub is **co-located** on node 1's address (`--hub-ip 10.99.0.1`). `set_stp_ip.sh` supports this: node 1's ASP leaves from port 2910 towards the neighbouring hub on 2908. There is no extra address to add and no `AllowedIPs` to widen.
- Order matters. If B starts before the hub, its ASP waits and `start-native.sh` prints `hub 10.99.0.1 muet` (hub silent). Restart B once the hub is up.

What each start does, in order: write `/etc/osmo-wan.conf` → `setup-wan-mesh.sh` (PJSIP trunks, `[wan_out]`/`[wan_in]` dialplan, SMS routes, REDIRECT, firewall) → inter-STP (A only) → `set_stp_ip.sh` (point codes, ASP `asp-to-inter` → `10.99.0.1:2908`) → start the stack.



2.48.1.6 6. Step 4 — check, layer by layer

Layer	Command	Expected
Tunnel	`sudo wg show`	handshake less than 2 min old

Layer	Command	Expected
SS7, from the hub (A)	`sudo checks/diag-interstp.sh`	both ASs `AS_ACTIVE`, no PROHIB/UNAVAIL route, no flapping
SS7, from a node	`sudo checks/ss7_check.sh`	`asp-to-inter` ACTIVE towards `10.99.0.1:2908`
SIP	`sudo asterisk -rx 'pjsip show endpoints'`	trunk to the peer `Avail` (qualify OK)
SMS	`cat /etc/osmocom/sms-routing.conf`	an entry for the remote node → `10.99.0.x:7890`
End to end	from mobile 1 on node 1, call `2220011`, then send an SMS to the same number	it rings and the SMS arrives on node 2; same the other way with `1110011`

Useful capture: `tcpdump -ni wg0 sctp or port 5060 or port 7890`. The bench's tcpdump wrapper also captures GSMTAP and M3UA.

Each step's logs are in `$LOG_DIR` (default `/tmp`): `wan-mesh.log`, `interstp.log`, `set-stp-ip.log`.

2.48.1.7 7. Variants

Node cards (`--gen-conf` / `--wan CARD`). Each node writes its card, the cards are exchanged, and each node imports the other's:

```
sudo ./start-native.sh --gen-conf mynode2.conf      # on B
sudo ./start-native.sh --wan mynode2.conf --node 1 --hub-ip 10.99.0.1   # on A
```

Before sending a card, check its `OSMO_NODE_IP` line. On a machine with no table yet, the detected address is the LAN one (`192.168.x`), not the tunnel's. Either fix the card by hand, or generate it **after** a first start with `--wan-nodes`: the table is authoritative once its address is up.

Same LAN, no tunnel. Same command with the LAN addresses (`--wan-nodes "1:192.168.1.10:11 2:192.168.1.20:22"`). `--hub-ip` is either node 1's address (co-located) or a free address in the /24, which `start-interstp.sh` will add.

More than two nodes. Add one entry per node to `--wan-nodes` (up to 9, prefixes 11 to 99), and one WireGuard `[Peer]` per node on A. With the hub on A, a star around A is enough for SS7. Voice and SMS go node to node, so every node must reach every other: either a full WireGuard mesh, or `AllowedIPs = 10.99.0.0/24` on the satellites with `net.ipv4.ip_forward=1` on A.

2.48.1.8 8. Common failures

Symptom	Cause	Fix
PROHIB route at the STP towards a node that is off	old entry in `/etc/osmo-wan.conf`	restart with `--regen-conf` **on the node that runs the hub**: a hub already running elsewhere is not rebuilt
`aucune IP locale dans la table - passez --wan-id N` (no local IP in the table)	the table address is not on any interface (public IP, tunnel down)	`--wan-id N --node N`, check `wg show`
`… repond au ping mais pas en SCTP 2908` (answers ping but not SCTP 2908)	the hub is not running on the machine that holds the address	start A first, or `sudo ./launch/start-interstp.sh --ip 10.99.0.1` on A
ASP going up and down (flapping in `diag-interstp.sh`)	firewall on 2908, or two hubs	§4; only one node with `--interstp`, the others `--no-interstp`
SIP trunk `Unavail`	5060 filtered (ufw), or wrong port targeted	§4; `pjsip show endpoints`, `tcpdump -ni wg0 port 5060`
The call rings but there is no audio	RTP filtered, or SDP with an address the peer cannot reach	open 20000–20499; the table address must be the tunnel's

Symptom	Cause	Fix
SMS lost between nodes	TCP 7890 filtered, or no route	`sms-routing.conf`, `nc -zv 10.99.0.2 7890`

3 Fichiers

3.1 /opt/GSM/osmo-operator/.gitignore

467 octets, 21 lignes → 21 lignes

```
# Cache des .deb compiles (voir packaging/osmo-deb.sh) : recopie depuis
# /var/cache/osmo-debs par build.sh avant chaque build, jamais versionne.
.deb-cache/*.deb
# Configs generees pour docker compose (compose.sh up)
data/compose/
packaging/dist/
__pycache__/
*.pyc
*.iso
vty-debug-dump-*.txt

# depot git a part (github.com/bbaranoff/dino), deploye par wrangler
dino/
*.bak-*
vty-debug-dump.txt
navigation/logs/
mynode*.conf
.claude/

# Jupyter
.ipynb_checkpoints/
```

3.2 /opt/GSM/osmo-operator/Dockerfile

81928 octets, 1291 lignes → 1291 lignes

```
# syntax=docker/dockerfile:1
# Dockerfile — IMAGE DE BASE osmocom-nitb (COUCHE STABLE)
# ──────────────────────────────────────────────────────────────────────
# Tout ce qui est LONG et rarement modifie vit ici : la TOTALITE des paquets apt
# (une seule liste, plus aucun apt dans Dockerfile.run), la pile Osmocom
# compilee depuis les sources, osmocom-bb + le firmware Calypso, QEMU et son
# device IPC, gr-gsm/GNU Radio, le runtime Node et le dashboard web, la config
# PulseAudio, les units systemd, les repertoires, le prompt.
# ~11 Go et ~40 min de build : on ne la rebatit que quand une dependance change.
#
# [2026-09-03] BASE ubuntu:24.04 (noble) - python 3.12, gcc 13 par defaut,
# bibliotheques renommees *t64 (libasound2t64, libgnutls30t64...). L ISO
# (build-iso.sh) part du MEME noble : le venv /root/.env et les .so de
# /usr/local sont copies tels quels dans le rootfs, ils doivent trouver la
# meme glibc et le meme python. Changer la base ici, c est la changer la-bas.
#
# CACHE .deb - packaging/osmo-deb.sh. Chaque dossier compile ci-dessous sort en
# paquet .deb dans /var/cache/osmo-debs (COPY depuis .deb-cache/, que build.sh
# synchronise avec /var/cache/osmo-debs de l HOTE). Au rebuild, `osmo-deb
# install` pose le paquet et saute le clone + la compilation ; `osmo-deb pack`
# et `osmo-deb snapshot` fabriquent le paquet la premiere fois. build.sh
# --no-cache passe OSMO_DEB_REFRESH=1 : tout est recompile et le cache reecrit.
# LES SOURCES RESTENT DANS /opt/GSM : `osmo-deb pack` embarque l arbre courant
# (/opt/GSM/<nom>, .git et objets compris) dans le paquet, et les snapshots
```

```
# nomment l arbre. Un rebuild depuis le cache redonne donc la meme image, avec
# ses ateliers - pas une image ou seul /usr/local subsiste.
#
# L'iteration quotidienne se fait dans Dockerfile.run, qui repart de cette image
# (`FROM osmocom-nitb`) et n'y rafraichit que les scripts, les configs, le pont
# et les arbres git qosmo / osmo-operator — en secondes, pas en 40 minutes.
#
# Cette image reste AUTONOME : elle a son propre ENTRYPOINT et start-nitb.sh la
# lance seule. Ne pas retirer ses COPY de configs/scripts sous pretexte que
# Dockerfile.run les refait : le recouvrement est voulu des deux cotes.

# ─────────────────────────────────────────────────────────────────────────────
# [2026-09-25] LE BUILD EST UN GRAPHE, PLUS UNE CHAINE — stages BuildKit
# ─────────────────────────────────────────────────────────────────────────────
# Ce fichier etait une seule cible de ~75 etapes : une chaine, donc un seul
# chemin d execution, meme sur une machine a 32 coeurs. Or quatre des cinq
# compilations longues n ont AUCUNE dependance entre elles - la pile Osmocom,
# QEMU (qosmo), le venv GNU Radio, la 4G (srsRAN + Open5GS). Elles s additionnaient
# pour rien.
#
#                  base   (apt, apt-fast, osmo-deb, configs)
#                    |
#         +----------+--------+---------+
#         |          |        |         |
#     osmo-core    qemu      lte      node          <- EN PARALLELE
#         |          |
#    +----+-----+    |
#    |    |     |    |
#   bb  grgsm-  +----+
#    |  venv    |
#    |    |    l1     (bb: osmocom-bb — l1: c54x_exe + grgsm_exe)
#    |    |     |
#    +----+-----+
#         |
#     osmocom-nitb      (assemblage : dpkg -i, puis configs)
#           |
#         debs          (FROM scratch : le cache .deb pour la CI)
#
# CE QUI REND LE DECOUPAGE POSSIBLE : packaging/osmo-deb.sh. Chaque bloc sort
# deja en .deb qui contient A LA FOIS les fichiers installes ET l arbre de
# sources de /opt/GSM (cmd_pack, « L ARBRE DE SOURCES PART AVEC »). Un stage
# parallele n a donc pas a exporter un systeme de fichiers : il exporte ses
# .deb, et l assemblage les repose avec un `dpkg -i`. C est EXACTEMENT ce que
# fait deja iso_modules/50-injection-image.sh pour le rootfs de l ISO — le
# precedent existe dans ce depot, on le reutilise.
#
# TROIS STAGES NE SONT PAS PARALLELES A osmo-core, et c est voulu — tous les
# trois se lient a libosmocore d une facon ou d une autre :
#   - osmocom-bb : `make nofirmware` lie mobile/trxcon a libosmocore ;
#   - c54x_exe   : son Makefile fait `pkg-config --libs libosmocoding libosmocore` ;
#   - gr-gsm     : son CMakeLists cherche libosmocore/libosmocoding/libosmogsm.
# Les en sortir donnerait un build qui casse, pas un build plus rapide. Ils
# tournent en revanche EN PARALLELE ENTRE EUX, une fois osmo-core fini.
#
# CE QUI NE CHANGE PAS : les versions, les patchs, les ./configure, le contenu
# de /opt/GSM, et --target osmocom-nitb (build.sh, compose.yaml) reste
# OBLIGATOIRE puisque `debs` est toujours la derniere etape du fichier.
# ─────────────────────────────────────────────────────────────────────────────


FROM ubuntu:24.04 AS base

# ROOT : ou vivent les sources dans l'image. Chemin FIXE et assume — dans un
# conteneur, il n'y a rien a rendre portable.
ARG DEBIAN_FRONTEND=noninteractive
ARG ROOT=/opt/GSM
```

```
# OSMO_IMAGE=1 : « image de ce depot, son reseau est pose par start-docker.sh /
# compose » - network/osmo-ip-plan.sh ne touche alors a aucune adresse. Un
# conteneur quelconque (ubuntu:24.04 nu) ne l a pas et s adresse comme un natif.
ENV container=docker \
    OSMO_IMAGE=1 \
    PKG_CONFIG_PATH=/usr/local/lib/pkgconfig \
    LD_LIBRARY_PATH=/usr/local/lib \
    GIT_TERMINAL_PROMPT=0

# ── apt-fast : les téléchargements apt en parallèle ──────────────────────
# Cette image installe plusieurs centaines de paquets ; apt les récupère un par
# un, apt-fast les met en parallèle via aria2. Le gain porte sur le
# téléchargement, pas sur dpkg — l'installation reste séquentielle.
#
# UN SEUL installeur pour tous les environnements : packaging/apt-fast-install.sh
# (cette image, Dockerfile.stp, le chroot de l ISO, l hote via build.sh). Il
# pose aussi /etc/apt/apt.conf.d/90osmo-operator, les reglages de
# telechargement communs. Repli integre : sans GitHub, apt-fast appelle apt-get
# et le build continue, plus lentement.
#
# LE CACHE APT DU BUILD. [2026-09-04] Chaque etape apt de ce fichier monte
# /var/cache/apt/archives en cache BuildKit (id osmo-apt-archives, partage avec
# Dockerfile.stp) : les .deb telecharges une fois y restent, sur l hote, dans le
# cache de BuildKit - une reconstruction ne retelecharge que ce qui a change.
# apt-fast-install retire docker-clean et pose Keep-Downloaded-Packages, sinon
# apt effacerait les paquets a peine installes. Necessite BuildKit (docker
# compose, ou docker-buildx : build.sh les installe) - le builder historique
# ne connait pas --mount.
COPY packaging/apt-fast-install.sh /usr/local/sbin/apt-fast-install
# Le miroir Ubuntu mesure par build.sh (packaging/apt-mirror.sh) : vide = les
# sources de l image de base (archive.ubuntu.com), qui peut etre inutilisable
# depuis l hote (3 Ko/s le 2026-09-04) sans qu apt le dise.
ARG OSMO_UBUNTU_MIRROR=
RUN --mount=type=cache,id=osmo-apt-archives,target=/var/cache/apt/archives,sharing=locked \
    chmod 755 /usr/local/sbin/apt-fast-install && apt-fast-install \
    && rm -rf /var/lib/apt/lists/*
ENV DEBIAN_FRONTEND=noninteractive

# 1. Dépendances système — TOUTES ici, y compris celles qu'installait
#    Dockerfile.run. Une seule liste, un seul endroit où la faire évoluer :
#    l'image d'exécution n'a plus à connaître apt du tout.
#
# tshark tire wireshark-common, qui pose une question debconf (capture non-root)
# et bloquerait un build non interactif : on y répond d'avance.
#
# ⚠ COUPLAGE EXTERIEUR — install_modules/10-deps.sh EXTRAIT cette liste du
# Dockerfile (« la liste fait autorite dans le Dockerfile ») avec
#   awk '/^RUN apt-get update && apt-get install/,/[^\\]$/'
# Ce motif ne connait QUE `apt-get` : depuis le passage a apt-fast, il ne
# capture plus que le bloc d'amorcage (aria2 curl ca-certificates) et
# l'installation NATIVE (hors docker) repart avec une liste tronquee —
# inst_deps_verify echoue sur build-essential/libtalloc-dev/libsctp-dev.
# Le correctif est dans 10-deps.sh (accepter `apt-(get|fast)` dans l'awk ET
# dans le sed qui suit), pas ici : ne pas repasser cette liste en apt-get pour
# contourner le probleme.
RUN echo 'wireshark-common wireshark-common/install-setuid boolean true' | debconf-set-selections
# deb-src AVANT l unique `apt-fast update` : le build-dep gnuradio (gr-gsm, fin
# du bloc ci-dessous) lit les index Sources. Il avait son propre RUN avec un
# deuxieme update + une deuxieme resolution apt : fusionne ici. Noble ecrit ses
# sources en deb822 (ubuntu.sources, ligne "Types: deb") ; l ancien
# /etc/apt/sources.list (jammy) reste gere au cas ou la base change (gnuradio
# est dans universe : tous les composants).
RUN if [ -f /etc/apt/sources.list.d/ubuntu.sources ]; then \
        sed -i 's/^Types: deb$/Types: deb deb-src/' /etc/apt/sources.list.d/ubuntu.sources; \
    else \
```

```
        sed -nE 's|^deb (http\S+) (\S+) .*|deb-src \1 \2 main restricted universe multiverse|p' \
            /etc/apt/sources.list | sort -u > /etc/apt/sources.list.d/deb-src.list; \
    fi
RUN --mount=type=cache,id=osmo-apt-archives,target=/var/cache/apt/archives,sharing=locked \
    apt-fast update && apt-fast install -y --no-install-recommends \
    # Outils de build
    build-essential git gcc g++ make cmake autoconf automake libtool pkg-config wget curl \
    # Dépendances Osmocom Core & Network
    libtalloc-dev libpcsclite-dev libsctp-dev libmnl-dev liburing-dev asterisk-moh* \
    libdbi-dev libdbd-sqlite3 libsqlite3-dev sqlite3 libc-ares-dev libgnutls28-dev \
    # Audio, Radio & SIP
    libortp-dev libfftw3-dev libusb-1.0-0-dev libsofia-sip-ua-dev libsofia-sip-ua-glib-dev \
    # Python & Outils système
    python3 python3-dev python3-scapy ca-certificates tmux systemd systemd-sysv \
    # Debug — gdb-multiarch pour attacher au gdb-stub QEMU (ARM Calypso)
    gdb-multiarch \
    # ALSA — requis par osmo-gapk pour l'I/O audio matériel
    # (libasound2t64 : nom noble de libasound2, transition time_t 64 bits)
    libasound2-dev libasound2t64 alsa-utils \
    # libgsm — codec GSM-FR natif (accélère gapk en mode gsmfr)
    libgsm1-dev libgsm1 \
    iptables iproute2 asterisk ffmpeg \
    # Sync build-iso : psmisc (pkill/killall, cleanup) + pulseaudio (chaîne audio gapk/parec)
    psmisc pulseaudio pulseaudio-utils binutils-arm-none-eabi \
    # Toolchains alternatives : osmocom-bb jolly/testing et fixeria/burst_ind ne
    # compilent qu'avec gcc-9 ; gcc-11 reste le compilateur par defaut du reste
    # (gcc-13 est celui de noble : il compile la pile Osmocom en tete de fichier,
    # les alternatives ci-dessous ne sont posees qu apres).
    # (Ex-bloc apt dedie, supprime : les `update-alternatives` plus bas echouaient
    #  « alternative path /usr/bin/gcc-9 doesn't exist » sans ces paquets.)
    gcc-9 g++-9 gcc-11 g++-11 \
    # log4cpp : etait installe a part avec le build-dep gnuradio (gr-gsm).
    liblog4cpp5-dev \
    # QEMU Calypso (bbaranoff/qosmO) : venv + numpy/scipy pour l'outillage
    # DSP, glib/pixman/slirp pour la cible arm-softmmu, ninja pour meson, socat
    # pour les PTY de scripts/run.sh. (Ex-bloc apt-get juste avant le build QEMU.)
    python3-venv python3-pip python3-numpy python3-scipy \
    libglib2.0-dev libpixman-1-dev libslirp-dev socat ninja-build \
    # Ex-Dockerfile.run — outillage d'exploitation du conteneur :
    #   telnet   -> repli VTY (run_modules/21-abonnes-hlr.sh) + transport dashboard
    #   whiptail -> menus de tools/vty-menu.sh et de start-docker.sh
    #   xz-utils -> `tar -xJf` du tarball Node 22 (bloc dashboard, en fin de fichier)
    #   libasound2-plugins -> greffon ALSA->PulseAudio, EXIGE par configs/asound.conf
    #        (`type pulse` sur gsm_out, gsm_in ET pcm.!default) : sans lui, toute
    #        ouverture ALSA (gapk, mobile) echoue et l'audio est mort des 2 cotes
    #   tshark/tcpdump/libcap2-bin -> capture ; libcap2-bin fournit le setcap ci-dessous
    telnet nano whiptail xz-utils libasound2-plugins \
    tcpdump tshark libcap2-bin \
    # Deps GNU Radio 3.10 (gr-gsm, venv /root/.env plus bas) : meme resolution
    # apt que le reste, pas de second RUN apt. (Le && ferme la liste pour
    # l extracteur de install_modules/10-deps.sh.)
    && apt-fast build-dep -y gnuradio \
    && rm -rf /var/lib/apt/lists/*

# Capture non-root pour le dashboard et tcpdump. Position OBLIGATOIRE : dumpcap
# n'existe qu'une fois wireshark-common installe (tire par tshark, juste au-dessus).
RUN setcap cap_net_raw,cap_net_admin+eip "$(command -v dumpcap)"

# ── Le cache .deb : l outil, puis les paquets deja construits ──────────────
# .deb-cache/ est le miroir, dans le contexte de build, de /var/cache/osmo-debs
# de l hote (build.sh l y recopie avant, et ramene les nouveaux paquets apres).
# Vide au premier build : tout se compile et se met en cache. Plein ensuite :
# chaque etape ci-dessous pose son paquet et ne compile rien.
# OSMO_DEB_REFRESH=1 (build.sh --no-cache) : le cache est ignore et reecrit.
ARG OSMO_DEB_REFRESH=0
COPY packaging/osmo-deb.sh /usr/local/sbin/osmo-deb
```

```
COPY .deb-cache/ /var/cache/osmo-debs/
RUN chmod 755 /usr/local/sbin/osmo-deb && osmo-deb list || true

# ── git : plus de forçage HTTP/1.1 ──────────────────────────────────────
# [2026-09-03] RETIRE. Le contournement http.version=HTTP/1.1 (contre "expected
# flush after ref listing", un flux HTTP/2 coupe par un intermediaire) etait
# pose ici en global et repete a chaque clone. Il n a plus lieu d etre : git
# parle HTTP/2 normalement, et forcer HTTP/1.1 ralentissait tous les clones
# pour un incident de reseau local. GIT_TERMINAL_PROMPT=0 (ENV, plus haut)
# reste : sans terminal, un clone qui veut un login doit ECHOUER net, pas
# attendre. postBuffer large pour les gros clones.
RUN git config --global http.postBuffer 524288000

SHELL ["/bin/bash", "-c"]
COPY configs/*conf /etc/asterisk/
# asound.conf DOIT AUSSI ETRE A LA RACINE. Le COPY ci-dessus le depose dans
# /etc/asterisk/ - ALSA ne regarde jamais la : il lit /etc/asound.conf (ou
# ~/.asoundrc). Sans ce second exemplaire, `aplay -D gsm_out` repond
#     ALSA lib pcm.c: (snd_pcm_open_noupdate) Unknown PCM gsm_out
# et le mobile du conteneur est MUET, sans qu aucun log ne parle d audio.
# start-docker.sh l.469 monte bien celui de l hote (-v ...:/etc/asound.conf), mais un
# conteneur lance autrement - docker run a la main, diagnostic - n a rien.
COPY configs/asound.conf /etc/asound.conf

WORKDIR ${ROOT}

# 2. Création de l'utilisateur osmocom
RUN groupadd osmocom && useradd -r -g osmocom -s /sbin/nologin -d /var/lib/osmocom osmocom && \
    mkdir -p /var/lib/osmocom && chown osmocom:osmocom /var/lib/osmocom

# ═══════════════════════════════════════════════════════════════════════
# STAGE osmo-core — la pile Osmocom : 17 depots, 4 patchs, gapk, SMSC, libosmo-dsp
# ═══════════════════════════════════════════════════════════════════════
FROM base AS osmo-core
# ARG ne traverse pas un FROM : les stages qui appellent osmo-deb ou
# ${ROOT} doivent les redeclarer, sinon ROOT est vide et
# OSMO_DEB_REFRESH=1 (build.sh --no-cache) ne descend pas jusqu ici.
ARG ROOT=/opt/GSM
ARG OSMO_DEB_REFRESH=0

# 3. Compilation de la pile Osmocom (Ordre respecté)
RUN for repo in \
    libosmocore:1.12.1 \
    libosmo-netif:1.7.0 \
    libosmo-abis:2.1.0 \
    libosmo-sigtran:2.2.1 \
    libsmpp34:1.14.5 \
    libgtpnl:1.3.3 \
    osmo-hlr:1.9.2 \
    osmo-mgw:1.15.0 \
    osmo-ggsn:1.14.0 \
    osmo-sgsn:1.13.1 \
    osmo-msc:1.15.0 \
    osmo-bsc:1.14.0 \
    osmo-trx:1.7.2 \
    osmo-bts:1.10.0 \
    osmo-pcu:1.5.2 \
    osmo-sip-connector:1.7.2 \
    libosmo-gprs:0.2.1; \
    do \
    name=$(echo $repo | cut -d: -f1) && \
    version=$(echo $repo | cut -d: -f2) && \
    \
    if [[ "$name" =~ ^libosmo ]]; then \
        GIT_URL="https://gitea.osmocom.org/osmocom/$name"; \
    else \
```

```
        GIT_URL="https://gitea.osmocom.org/cellular-infrastructure/$name"; \
    fi && \
    \
    # Le paquet du cache d abord : s il est la, ni clone ni compilation.
    if osmo-deb install "$name" "$version"; then continue; fi && \
    \
    cd ${ROOT} && \
    git clone "$GIT_URL" && cd "$name" && \
    git checkout "$version" && \
    \
    autoreconf -fi && \
    EXTRA_FLAGS="" && \
    if [ "$name" = "libosmo-abis" ]; then EXTRA_FLAGS="--disable-dahdi"; fi && \
    if [ "$name" = "osmo-msc" ]; then EXTRA_FLAGS="--enable-smpp"; fi && \
    if [ "$name" = "osmo-mgw" ]; then EXTRA_FLAGS="--enable-alsa"; fi && \
    if [ "$name" = "osmo-trx" ]; then EXTRA_FLAGS="--with-ipc"; fi && \
    if [ "$name" = "osmo-bts" ]; then EXTRA_FLAGS="--enable-virtual --enable-trx"; fi && \
    if [ "$name" = "osmo-ggsn" ]; then EXTRA_FLAGS="--enable-gtp-linux"; fi && \
    \
    ./configure $EXTRA_FLAGS && \
    make -j$(nproc) && \
    # make install sous DESTDIR -> .deb dans le cache -> dpkg -i dans la racine
    osmo-deb pack "$name" "$version" make install && \
    ldconfig \
    || { echo "ECHEC build osmocom: $name"; exit 1; }; \
    done

# ── Patch osmo-trx IPC : alignement ts_initial sur la trame TDMA (fix RACH/LU) ──
# Le device IPC (calypso-ipc-device) commit des buffers RX a des timestamps
# multiples de 2500 (CALYPSO_SHM_BUFSIZE) ; sans arrondi, ts_initial%5000 valait
# 0 ou 2500 -> device-TN0 mappe sur osmo-TN4 -> la RACH (UL) ratait le slot
# TS0/RACH -> NOPE/-110 -> Location Update bloquee. Le patch arrondit ts_initial
# a la frontiere de trame (8*625=5000 @ 4 SPS). Patch maintenu dans patches/.
# ── Patch osmo-trx RACH UL : table de modulation per-RA (fix LU, no-hardcode) ──
# La vraie RA du mobile (d_rach@0x0474, plombee via /dev/shm/calypso_rach par QEMU)
# varie a chaque burst ; le device rejouait un RA=3 fixe -> request-reference de
# l'IMM ASSIGN jamais matchee -> LU en boucle. sigProcLib pre-genere la modulation
# Laurent EXACTE d'osmo-trx pour chaque RA 0x00..0x0f (-> /root/rach_ref_RA<nn>.cs16),
# le device selectionne le ref de la VRAIE RA. Ajoute aussi les logs RACH-DET
# (Transceiver.cpp) + le lien libosmocoding dans COMMON_LDADD (Makefile.am, requis
# pour gsm0503_rach_ext_encode). Touche Makefile.am -> autoreconf+configure requis.
COPY patches/osmo-trx-ipc-ts-frame-align.patch /tmp/osmo-trx-ipc-ts-frame-align.patch
COPY patches/osmo-trx-rach-per-ra-table.patch /tmp/osmo-trx-rach-per-ra-table.patch
# Meme nom de paquet que dans la boucle (osmo-trx), version 1.7.2+ipc : dpkg
# fait une mise a jour, les fichiers sont les memes. Si la boucle est sortie du
# cache, l arbre source n existe pas : on le reclone avant d appliquer les patchs.
RUN if ! osmo-deb install osmo-trx 1.7.2+ipc; then \
      { [ -d ${ROOT}/osmo-trx ] || { cd ${ROOT} \
          && git clone https://gitea.osmocom.org/cellular-infrastructure/osmo-trx \
          && git -C osmo-trx checkout 1.7.2; }; } \
      && git -C ${ROOT}/osmo-trx apply /tmp/osmo-trx-ipc-ts-frame-align.patch \
      && git -C ${ROOT}/osmo-trx apply /tmp/osmo-trx-rach-per-ra-table.patch \
      && cd ${ROOT}/osmo-trx \
      && autoreconf -fi \
      && ./configure --with-ipc \
      && make -j$(nproc) \
      && osmo-deb pack osmo-trx 1.7.2+ipc make install \
      && ldconfig; \
    fi

# ── Patch osmo-bts RAND : forcer le defi d'authentification ──────────────
# osmo-bts relaie normalement l'AUTHENTICATION REQUEST du reseau telle quelle.
# Ce patch, dans rsl_rx_rll() (src/common/rsl.c), reecrit le RAND de cette
# trame quand la variable d'environnement RAND est posee sur le processus BTS :
#     RAND non defini -> rien de touche (comportement normal) ;
#     RAND=0          -> RAND nul sur 16 octets (defi fixe COMP128v1,
```

```
#                       defi fixe connu, Ki = prefixe connu
#                       + 2 octets a retrouver) ;
#     RAND=<hexa>     -> jusqu'a 16 octets lus en hexa, completes de zeros.
# « seulement quand precise, mais force quand precise ». Ne touche que rsl.c,
# donc pas de Makefile.am : autoreconf/configure quand meme, l'arbre reclone
# arrive sans ./configure. Meme nom de paquet (osmo-bts), version 1.10.0+rand :
# dpkg fait une mise a jour. Patch maintenu dans patches/.
COPY patches/osmo-bts-force-rand-toy.patch /tmp/osmo-bts-force-rand-toy.patch
RUN if ! osmo-deb install osmo-bts 1.10.0+rand; then \
      { [ -d ${ROOT}/osmo-bts ] || { cd ${ROOT} \
          && git clone https://gitea.osmocom.org/cellular-infrastructure/osmo-bts \
          && git -C osmo-bts checkout 1.10.0; }; } \
      && git -C ${ROOT}/osmo-bts apply /tmp/osmo-bts-force-rand-toy.patch \
      && cd ${ROOT}/osmo-bts \
      && autoreconf -fi \
      && ./configure --enable-virtual --enable-trx \
      && make -j$(nproc) \
      && osmo-deb pack osmo-bts 1.10.0+rand make install \
      && ldconfig; \
    fi

# ── Patch osmo-hlr RAND : forcer le RAND du vecteur d'auth ──────────────────
# auc.c/auc_compute_vectors() : gate minimal. Si la variable d'environnement
# RAND est posee sur le process HLR, le RAND tire pour CHAQUE vecteur est force
# a 0 (16 octets) AVANT osmo_auth_gen_vec2 : SRES et Kc sont alors calcules sur
# RAND=0 (le HLR a le Ki). C'est l'etage indispensable pour une LU ACCEPTEE :
# osmo-msc envoie ce RAND et compare au SRES du meme vecteur -> match. Le seul
# gate osmo-msc (cote tx) donnerait un mismatch -> LU reject. RAND absent ->
# reseau normal. Ne touche pas Makefile.am ; autoreconf/configure quand meme
# (arbre reclone sans ./configure). Meme nom de paquet (osmo-hlr), version
# 1.9.2+rand : dpkg met a jour. Patch maintenu dans patches/.
COPY patches/osmo-hlr-force-rand-toy.patch /tmp/osmo-hlr-force-rand-toy.patch
RUN if ! osmo-deb install osmo-hlr 1.9.2+rand; then \
      { [ -d ${ROOT}/osmo-hlr ] || { cd ${ROOT} \
          && git clone https://gitea.osmocom.org/cellular-infrastructure/osmo-hlr \
          && git -C osmo-hlr checkout 1.9.2; }; } \
      && git -C ${ROOT}/osmo-hlr apply /tmp/osmo-hlr-force-rand-toy.patch \
      && cd ${ROOT}/osmo-hlr \
      && autoreconf -fi \
      && ./configure \
      && make -j$(nproc) \
      && osmo-deb pack osmo-hlr 1.9.2+rand make install \
      && ldconfig; \
    fi

# ── Patch osmo-msc RAND : forcer le defi d'authentification ─────────────────
# gsm48_tx_mm_auth_req() (src/libmsc/gsm_04_08.c) : gate minimal. Si la variable
# d'environnement RAND est posee sur le processus MSC, le RAND de l'AUTH REQUEST
# descendante est force (0 sur 16 octets dans notre cas ; changer l'initialiseur
# forced_rand[] pour un autre RAND). RAND absent -> comportement normal du reseau.
# Ne touche pas Makefile.am ; autoreconf/configure quand meme (arbre reclone sans
# ./configure). Meme nom de paquet (osmo-msc), version 1.15.0+rand : dpkg met a
# jour. Patch maintenu dans patches/. Meme gate cote BSC (openbsc gsm_04_08.c).
COPY patches/osmo-msc-force-rand-toy.patch /tmp/osmo-msc-force-rand-toy.patch
RUN if ! osmo-deb install osmo-msc 1.15.0+rand; then \
      { [ -d ${ROOT}/osmo-msc ] || { cd ${ROOT} \
          && git clone https://gitea.osmocom.org/cellular-infrastructure/osmo-msc \
          && git -C osmo-msc checkout 1.15.0; }; } \
      && git -C ${ROOT}/osmo-msc apply /tmp/osmo-msc-force-rand-toy.patch \
      && cd ${ROOT}/osmo-msc \
      && autoreconf -fi \
      && ./configure --enable-smpp \
      && make -j$(nproc) \
      && osmo-deb pack osmo-msc 1.15.0+rand make install \
      && ldconfig; \
    fi
```

```
# ── Patch gapk : sonde sur la sortie ALSA (GAPK_ALSA_PROBE) ──────────────────
# Diagnostic pur, inerte tant que GAPK_ALSA_PROBE != 1. Tranche une question que
# rien d'autre ne permet de trancher : quand l'ecouteur du MS est muet, le PCM
# remis a snd_pcm_writei est-il deja nul (defaut en amont, dans la chaine gapk)
# ou non nul (defaut en aval, ALSA/PulseAudio) ? osmo_gapk_pq_execute ne
# journalise rien ici, car il ne se plaint que d'un retour negatif et
# pq_cb_alsa_output rend « succes » meme pour une ecriture vide.
# Patch maintenu dans patches/ (regenere si pq_alsa.c change).
#
# [2026-08-12] LE PATCH RESTE DANS LE DEPOT, LE BUILD NE L'APPLIQUE PLUS.
# Demande explicite : garder patches/gapk-pq-alsa-output-probe.patch versionne
# (il tranche encore la question amont/aval quand l'ecouteur est muet) mais ne
# plus le poser sur le gapk construit — un `git apply` dans le build casse
# l'image des que pq_alsa.c bouge en amont, et la sonde n'est plus la question
# du jour (descendant muet resolu par CALYPSO_PULSE_LATENCY_MSEC, 240 depuis
# le 2026-09-09 : cf. start-native.sh, le tampon du greffon ALSA-PulseAudio).
# POUR LE REMETTRE, deux gestes, dans cet ordre :
#   1. decommenter la ligne COPY ci-dessous ;
#   2. reinserer dans le RUN, entre le `git clone` et le `cd osmo-gapk` :
#        git -C ${ROOT}/osmo-gapk apply /tmp/gapk-pq-alsa-output-probe.patch && \
#      (elle ne peut pas rester en commentaire : une ligne `#` au milieu d'une
#       continuation `\` couperait la chaine shell du RUN).
# Verifier d'abord que le patch s'applique toujours :
#   git -C <clone gapk> apply --check patches/gapk-pq-alsa-output-probe.patch
# POUR L'APPLIQUER A CHAUD SANS REBUILD : le poser dans le conteneur sur un
# clone de gapk, puis reconstruire gapk seul.
# COPY patches/gapk-pq-alsa-output-probe.patch /tmp/gapk-pq-alsa-output-probe.patch
RUN if ! osmo-deb install osmo-gapk 0.git; then \
      cd ${ROOT} && \
      git clone https://gitea.osmocom.org/osmocom/gapk osmo-gapk && \
      cd osmo-gapk && \
      autoreconf -fi && \
      ./configure --enable-alsa && \
      make -j$(nproc) && \
      osmo-deb pack osmo-gapk 0.git make install && \
      ldconfig; \
    fi

# ── gsup-smsc-proto : SMSC externe connecté à OsmoHLR via GSUP ──────────────
# Programmes : proto-smsc-daemon (réception MO SMS + relai MT via GSUP)
#              proto-smsc-sendmt (injection MT SMS via socket UNIX local)
# Dépendances build : libosmocore, libosmogsm, libosmo-gsup-client
# [2026-09-03] Son Makefile de tete lance `make DESTDIR= install` dans daemon/
# et sendmt/ - DESTDIR force a vide, le staging d osmo-deb restait vide ("rien
# n a ete installe dans DESTDIR") alors que la compilation (gcc courant) etait
# bonne. On installe donc les deux sous-repertoires nous-memes, avec le DESTDIR
# qu osmo-deb pose : leurs Makefiles, eux, l honorent.
RUN if ! osmo-deb install gsup-smsc-proto 0.git; then \
      cd ${ROOT} && \
      git clone https://gitea.osmocom.org/themwi/gsup-smsc-proto && \
      cd gsup-smsc-proto && \
      ./configure --with-osmo=/usr/local && \
      make -j$(nproc) && \
      osmo-deb pack gsup-smsc-proto 0.git \
        sh -c 'for i in daemon sendmt; do make -C "$i" DESTDIR="$DESTDIR" install || exit 1; done' && \
      ldconfig; \
    fi

# ── sms-coding-utils : encodage/décodage SMS PDU (GSM 03.40) ────────────────
# sms-encode-text, gen-sms-deliver-pdu, sms-pdu-decode, etc.
# Son Makefile de tete pose `DESTDIR=` (vide) et le repasse aux sous-repertoires :
# une variable donnee SUR LA LIGNE DE COMMANDE de make l emporte et descend
# avec, d ou `make install DESTDIR=...` explicite (l ancien INSTDIR=... n existait
# dans aucun de ses Makefiles). bindir vaut /usr/local/bin par configure.
RUN if ! osmo-deb install sms-coding-utils 0.r1; then \
```

```
      cd ${ROOT} && \
      wget -q https://raw.githubusercontent.com/bbaranoff/fc-magnetite/main/freecalypso-env/sms-coding-utils-
       ↪  latest.tar.bz2 &&
       ↪  \
      tar xf sms-coding-utils-latest.tar.bz2 && \
      cd sms-coding-utils-r1 && \
      ./configure && \
      make -j$(nproc) && \
      osmo-deb pack sms-coding-utils 0.r1 \
        sh -c 'mkdir -p "$DESTDIR/usr/local/bin" && make install DESTDIR="$DESTDIR"'; \
    fi

# ── libosmo-dsp (dépendance transceiver/burst_ind) ──────────────────────
RUN if ! osmo-deb install libosmo-dsp 0.git; then \
      cd /opt/GSM \
      && git clone https://gitea.osmocom.org/sdr/libosmo-dsp.git \
      && cd libosmo-dsp \
      && autoreconf -fi \
      && ./configure \
      && make -j$(nproc) \
      && osmo-deb pack libosmo-dsp 0.git make install \
      && ldconfig; \
    fi

# ═══════════════════════════════════════════════════════════════════════
# STAGE bb — osmocom-bb, le firmware Calypso et les deux branches gcc-9
# ═══════════════════════════════════════════════════════════════════════
FROM osmo-core AS bb
# ARG ne traverse pas un FROM : les stages qui appellent osmo-deb ou
# ${ROOT} doivent les redeclarer, sinon ROOT est vide et
# OSMO_DEB_REFRESH=1 (build.sh --no-cache) ne descend pas jusqu ici.
ARG ROOT=/opt/GSM
ARG OSMO_DEB_REFRESH=0

# ── Calypso build ──────────────────────────

# ── Patch osmocon : filtre Kc — RETIRE le 2026-08-30 ────────────────────
# Ce patch faisait ecrire /dev/shm/calypso_kc a osmocon, en espionnant
# L1CTL_CRYPTO_REQ au passage sur le lien serie. Il etait la SEULE source du Kc
# a l'epoque ; il ne l'est plus, et il faisait desormais plus de mal que de bien.
#
# POURQUOI ON LE RETIRE. Un espion voit passer la cle, mais pas sa fin de vie :
# il doit la DEVINER. Le patch la devinait sur DM_EST_REQ et DM_REL_REQ, et
# remettait le fichier a 32 zeros. L'intention est juste pour le RELEASE, fausse
# pour l'ESTABLISH -- le mobile emet aussi un DM_EST_REQ pour ouvrir un lien
# SUPPLEMENTAIRE sur un canal DEJA chiffre (le SAPI 3 du SMS). La cle
# disparaissait donc en pleine session chiffree.
#
# Pire, il y avait TROIS ecrivains sur ce fichier (osmocon, l1ctl_sock.c, le
# shunt DSP) sans le moindre arbitrage : le dernier qui ecrit gagne, et c'est
# l'effaceur. Mesure du 30/08 : fichier a 32 zeros, « A5=non » sur tout le canal
# dedie, premier appel mort.
#
# CE QUI LE REMPLACE. calypso_dsp_shunt.c publie /dev/shm/calypso_kc_l1 depuis
# d_a5mode et a_kc du NDB -- ce que le FIRMWARE a reellement charge dans le DSP
# (calypso/dsp.c:dsp_load_ciph_param). Ce n'est plus un espion qui devine, c'est
# l'etat de la couche 1, remis a zero par le firmware lui-meme quand il repart
# en clair. Et le fichier lui est PROPRE : plus de course a l'ecriture.
#
# Le patch reste dans patches/ a titre documentaire, il n'est plus applique.
# L arbre ENTIER part dans le paquet (snapshot) : trx_toolkit, osmocon et les
# binaires host sont lus dans l arbre au runtime, pas seulement dans /usr/local.
# [2026-09-08] trx_toolkit/clck_gen.py : l horloge TDMA de fake_trx ne se
# remet a zero que pour un retard superieur a une trame (4,615 ms). Avant, un
# retard de 0,3 a 2 ms decalait la base de temps a chaque fois ; osmo-bts-trx
# compensait en boucle et la parole perdait des trames. Patch maintenu dans
```

```
# patches/, applique aussi par install_modules/40-patches.sh en natif.
# mobile : une SIM reinseree sans RPLMN (reconnexion SAP) restait en A6 et la
# recherche de PLMN finissait « unhandled » (patches/osmocom-bb-plmn-a6-reinsert.patch).
# PLMNsel / FPLMN lus avec osmo_plmn_to_bcd (ecriture) au lieu de from_bcd : 000-00
# (patches/osmocom-bb-sim-plmn-from-bcd.patch). Les deux, aussi, par 40-patches.sh.
COPY patches/osmocom-bb-clck-gen-frame-tolerance.patch /tmp/osmocom-bb-clck-gen-frame-tolerance.patch
COPY patches/osmocom-bb-plmn-a6-reinsert.patch /tmp/osmocom-bb-plmn-a6-reinsert.patch
COPY patches/osmocom-bb-sim-plmn-from-bcd.patch /tmp/osmocom-bb-sim-plmn-from-bcd.patch
# [2026-10-23] OSMOCOM_BB_REF : osmocom-bb est EPINGLE. En amont, a0a315e2 « trxcon/trx_if: migrate to
# libosmo-trx » a sorti trxcon de `make nofirmware` et lui fait exiger libosmo-trx : un clone de la tete ne
# produit plus src/host/trxcon/src/trxcon (install plus bas en echec, notebook d'installation du 23/10). eac4eed1
# est l'arbre du banc, celui sur lequel les trois patchs sont faits.
ARG OSMOCOM_BB_REF=eac4eed1
RUN if ! osmo-deb install osmocom-bb 0.git; then \
      cd ${ROOT} && \
      git clone https://gitea.osmocom.org/phone-side/osmocom-bb && \
      git -C ${ROOT}/osmocom-bb checkout -q ${OSMOCOM_BB_REF} && \
      git -C ${ROOT}/osmocom-bb apply /tmp/osmocom-bb-clck-gen-frame-tolerance.patch && \
      git -C ${ROOT}/osmocom-bb apply /tmp/osmocom-bb-plmn-a6-reinsert.patch && \
      git -C ${ROOT}/osmocom-bb apply /tmp/osmocom-bb-sim-plmn-from-bcd.patch && \
      cd osmocom-bb/src && \
      # Build complet : firmware (layer1.bin/.elf pour Calypso) + outils host
      # (mobile, trxcon, virtphy, ccch_scan). Le firmware est nécessaire pour
      # le mode PHY_MODE=qemu où QEMU émule un Calypso et exécute layer1.
      make nofirmware && \
      osmo-deb snapshot osmocom-bb 0.git ${ROOT}/osmocom-bb; \
    fi

# ── Note historique — patch fake_trx TRXD v0 (RETIRE) ──────────────────────
# Conserve verbatim : il documente une panne vecue. Il flottait dans
# Dockerfile.run juste au-dessus du bloc firmware, avec lequel il n'a AUCUN
# rapport — voisinage accidentel, pas lien de cause a effet.
# ── Patch fake_trx : forcer TRXD v0 ────────────────────────────────────────
# osmo-bts-trx 1.10+ négocie TRXD v1 via SETFORMAT, mais trxcon (OsmocomBB)
# ne supporte que v0. Le header v1 a 2 octets de plus → décalage des soft-bits
# → BER constant 55/456 sur chaque frame → FBSB_SEARCH échoue en boucle.
# Fix : forcer ver_req=0 dans la réponse SETFORMAT → le BTS reste en v0.
# (patch TRXD v0 retire : burst_fwd.py:66 re-encode deja par interface,
#  trxcon recoit du v0 quoi qu'il arrive -> le forcage etait inutile)

# ── Firmware Calypso prebuild (layer1.highram.*) — ex-Dockerfile.run ────────
# `make nofirmware` ci-dessus ne construit justement PAS le firmware : on prend
# les binaires prebuild de bbaranoff/firmware et on les depose la ou le mode
# PHY_MODE=qemu les cherche. ORDRE IMPOSE : ce bloc exige A LA FOIS ce clone ET
# l'arbre /opt/GSM/osmocom-bb ci-dessus — le placer plus haut ferait echouer les
# cp et PHY_MODE=qemu partirait sans layer1.
# (Le `rm -rf /opt/GSM/firmware` qui precedait le clone dans Dockerfile.run est
#  supprime : ici le chemin n'existe pas encore, l'instruction etait morte.)
# GIT_TERMINAL_PROMPT=0 : sans terminal (docker build), un clone qui demande un
# login doit echouer NET (message clair) au lieu d attendre un identifiant que
# personne ne tapera. --depth 1 : on ne veut que les binaires prebuild, pas
# l historique. (Le -c http.version=HTTP/1.1 qui etait ici est retire.)
RUN if ! osmo-deb install calypso-firmware 0.git; then \
      GIT_TERMINAL_PROMPT=0 git clone --depth 1 \
        https://github.com/bbaranoff/firmware /opt/GSM/firmware \
      && osmo-deb snapshot calypso-firmware 0.git /opt/GSM/firmware; \
    fi
# [2026-08-28] Les trois `cp` vers /opt/GSM/osmocom-bb/src/target/firmware qui
# suivaient sont SUPPRIMES. Ils entretenaient un deuxieme exemplaire du firmware
# que plus personne ne lit : depuis la normalisation, environnement/paths.env du
# depot qemu ne connait qu'un chemin, $GSM_ROOT/firmware, et c'est celui que
# QEMU (-kernel .elf) comme osmocon (romload .bin) recoivent.
# Verifie sur le banc 192.168.1.7 : les deux exemplaires etaient bit a bit
# identiques (md5 3363105f...), et seul /opt/GSM/firmware etait charge.
# Si vous recompilez le firmware dans osmocom-bb, deposez le resultat dans
# /opt/GSM/firmware : c'est desormais le seul endroit consulte.
```

```
# ── GCC 9 pour osmocom-bb branches expérimentales (jolly/testing, burst_ind) ─
# gcc-9 et gcc-11 sont installés avec le reste, plus haut : ici on ne fait que
# déclarer les alternatives, dont l'ordre compte pour osmocom-bb.
RUN update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-9 90 \
       --slave /usr/bin/g++ g++ /usr/bin/g++-9 \
    && update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-11 110 \
       --slave /usr/bin/g++ g++ /usr/bin/g++-11

RUN update-alternatives --set gcc /usr/bin/gcc-9

# osmocom-bb jolly/testing → transceiver (BTS soft-SDR pour Calypso)
# Le binaire ET l arbre partent dans le paquet : les sources restent dans
# /opt/GSM au rebuild depuis le cache (voir l en-tete, CACHE .deb).
# [2026-10-19] patches/osmocom-bb-transceiver-qemu-bench.patch : l appli hote
# `transceiver` sert le Calypso EMULE (start-native.sh --osmo-trx, c54x_exe/run.sh
# OSMO_TRX=1) : --no-sync (pas de cellule a ecouter, mode BTS des le reset),
# --ts-mask (un seul timeslot), --l1ctl/--base-port, et SETTSC/SETFORMAT/
# NOMTXPOWER acquiesces (osmo-bts-trx >= 1.3 les tient pour critiques).
# Applique aussi par install_modules/40-patches.sh en natif.
COPY patches/osmocom-bb-transceiver-qemu-bench.patch /tmp/osmocom-bb-transceiver-qemu-bench.patch
RUN if ! osmo-deb install osmocom-bb-transceiver 0.git; then \
      git clone --branch jolly/testing --depth 1 \
        https://gitea.osmocom.org/phone-side/osmocom-bb.git \
        /opt/GSM/osmocom-bb-transceiver \
      && git -C /opt/GSM/osmocom-bb-transceiver apply /tmp/osmocom-bb-transceiver-qemu-bench.patch \
      && cd /opt/GSM/osmocom-bb-transceiver/src \
      && make HOST_layer23_CONFARGS=--enable-transceiver nofirmware -j$(nproc) \
      && cp /opt/GSM/osmocom-bb-transceiver/src/host/layer23/src/transceiver/transceiver \
         /usr/local/bin/transceiver \
      && osmo-deb snapshot osmocom-bb-transceiver 0.git /usr/local/bin/transceiver \
            /opt/GSM/osmocom-bb-transceiver; \
    fi

# osmocom-bb fixeria/burst_ind → ccch_scan / bcch_scan / cell_log
RUN if ! osmo-deb install osmocom-bb-burst-ind 0.git; then \
      git clone --branch fixeria/burst_ind --depth 1 \
        https://gitea.osmocom.org/phone-side/osmocom-bb.git \
        /opt/GSM/osmocom-bb-burst_ind \
      && cd /opt/GSM/osmocom-bb-burst_ind/src \
      && make nofirmware -j$(nproc) \
      && cp /opt/GSM/osmocom-bb-burst_ind/src/host/layer23/src/misc/ccch_scan \
         /usr/local/bin/ccch_scan \
      && { cp /opt/GSM/osmocom-bb-burst_ind/src/host/layer23/src/misc/bcch_scan \
         /usr/local/bin/bcch_scan 2>/dev/null || true; } \
      && { cp /opt/GSM/osmocom-bb-burst_ind/src/host/layer23/src/misc/cell_log \
         /usr/local/bin/cell_log 2>/dev/null || true; } \
      && osmo-deb snapshot osmocom-bb-burst-ind 0.git /opt/GSM/osmocom-bb-burst_ind \
           $(ls /usr/local/bin/ccch_scan /usr/local/bin/bcch_scan /usr/local/bin/cell_log 2>/dev/null); \
    fi

# Retour au compilateur par defaut : ce stage ne sert qu a produire des .deb,
# mais le laisser sur gcc-9 piegerait quiconque en derive une etape.
RUN update-alternatives --set gcc /usr/bin/gcc-11

# ═══════════════════════════════════════════════════════════════════════
# STAGE qemu — le seul arbre QEMU : /opt/GSM/qosmo (--enable-l1-grgsm)
# ═══════════════════════════════════════════════════════════════════════
FROM base AS qemu
# ARG ne traverse pas un FROM : les stages qui appellent osmo-deb ou
# ${ROOT} doivent les redeclarer, sinon ROOT est vide et
# OSMO_DEB_REFRESH=1 (build.sh --no-cache) ne descend pas jusqu ici.
ARG ROOT=/opt/GSM
ARG OSMO_DEB_REFRESH=0

# ═══════════════════════════════════════════════════════════════════════
```

```
# QEMU Calypso — RAN virtuel (baseband émulé)
# ──────────────────────────────────────────────────────────────────────────
# Architecture :
#   - QEMU émule un SoC Calypso (ARM7TDMI + DSP TMS320C54x)
#   - L'ARM exécute le vrai firmware osmocom-bb layer1.highram.elf
#   - Le DSP charge le ROM réel (calypso_dsp.txt) au boot
#   - bridge.py relaie les bursts entre osmo-bts-trx (UDP 5700-5702)
#     et la BSP du DSP (UDP 6702), avec QEMU comme maître d'horloge TDMA
#   - Le mobile (layer23) se connecte directement au socket L1CTL
#     publié par le firmware via la PTY série de QEMU
#
# Voir scripts/run.sh PHY_MODE=qemu pour l'orchestration runtime.
# ──────────────────────────────────────────────────────────────────────────

# (Le `apt-get install python3-venv python3-pip python3-numpy python3-scipy
#  libglib2.0-dev libpixman-1-dev libslirp-dev socat ninja-build` qui etait ici
#  a ete fusionne dans la liste apt-fast en tete de fichier : une seule liste,
#  un seul endroit ou la faire evoluer. Aucun paquet perdu.)
# ──────────────────────────────────────────────────────────────────────────
# [2026-09-25] qosmo + c54x_exe + grgsm_exe — remplacent qosmo-grgsm et qosmo-dsp
# ──────────────────────────────────────────────────────────────────────────
# Les deux forks QEMU sont retires (cf. start-native.sh, environment/paths.env) :
#   /opt/GSM/qosmo      UN seul arbre QEMU (bbaranoff/qosm0), --enable-l1-grgsm.
#                       Il porte run.sh, run_modules, cfgs. Le meme binaire sert
#                       au mode --dsp : CALYPSO_DSP_EXTERN=1 y coupe la L1 gr-gsm.
#                       `ninja install` pose aussi le lanceur C `qosmo`.
#   /opt/GSM/c54x_exe   le C54x HORS de QEMU (mask-ROM TI, sa ROM dans rom/).
#                       Il compile les sources de /opt/GSM/qosmo : APRES qosmo.
#   /opt/GSM/grgsm_exe  la couche 1 gr-gsm hors QEMU, memes sources qosmo.
# Snapshot de l arbre qosmo ENTIER, build/ compris : QEMU_BIN pointe sur
# $OQC_ROOT/build/qemu-system-arm, et QEMU lit build/qemu-bundle pour se
# relocaliser (voir Dockerfile.lite).
# L ancien RUN « /opt/GSM/qemu/{build,*.py} » et calypso-ipc-device disparaissent
# avec qosmo-grgsm : plus rien ne les lit.
RUN if ! osmo-deb install qosmo 0.git; then \
      git clone https://github.com/bbaranoff/qosm0 /opt/GSM/qosmo \
      && cd /opt/GSM/qosmo \
      && python3 -m venv /root/.venv-qemu \
      && . /root/.venv-qemu/bin/activate \
      && pip install --no-cache-dir tomli \
      && mkdir -p build && cd build \
      && ../configure --target-list=arm-softmmu --enable-l1-grgsm \
            --prefix=/opt/GSM/qemu-install --disable-werror --disable-docs \
      && make -j$(nproc) \
      && make install \
      && cp /opt/GSM/qemu-install/bin/qemu-system-arm /usr/local/bin/qemu-system-arm \
      && cp /opt/GSM/qemu-install/bin/qosmo /usr/local/bin/qosmo \
      && osmo-deb snapshot qosmo 0.git /opt/GSM/qosmo /opt/GSM/qemu-install \
            /root/.venv-qemu /usr/local/bin/qemu-system-arm /usr/local/bin/qosmo; \
    fi

# ──────────────────────────────────────────────────────────────────────────
# STAGE l1 — les deux couches 1 hors QEMU : c54x_exe et grgsm_exe
# ──────────────────────────────────────────────────────────────────────────
FROM qemu AS l1
# ARG ne traverse pas un FROM : les stages qui appellent osmo-deb ou
# ${ROOT} doivent les redeclarer, sinon ROOT est vide et
# OSMO_DEB_REFRESH=1 (build.sh --no-cache) ne descend pas jusqu ici.
ARG ROOT=/opt/GSM
ARG OSMO_DEB_REFRESH=0

# Elles compilent les sources de /opt/GSM/qosmo (deja la, ce stage en derive)
# et se lient a libosmocoding/libosmocore pour c54x_exe. On rapatrie donc
# /usr/local depuis osmo-core plutot que de faire deriver ce stage de lui :
# copier /usr/local (quelques centaines de Mo) coute bien moins que de copier
# l arbre de build de QEMU (~1,5 Go d objets) dans l autre sens.
```

```
COPY --from=osmo-core /usr/local /usr/local
RUN ldconfig

# c54x_exe : le Makefile met -march=native dans CFLAGS - dans une image qui
# tourne sur d autres CPU que celui du build, c est un SIGILL au demarrage. On
# garde ses drapeaux, sans celui-la. La ROM est aussi posee en
# /opt/GSM/calypso_dsp.*.bin, le --rom-dir par defaut de c54x_exe.
# [2026-10-10] La ROM n est plus dans le depot c54x_exe : le `cp rom/...`
# tombait sur « cannot stat rom/calypso_dsp.txt ». rom/fetch-rom.sh telecharge
# le dump FreeCalypso 3606, le convertit en .bin et verifie SHA256SUMS.3606 ;
# calypso_dsp.txt, c est le dump brut lui-meme, qu il ne garde pas : on le
# telecharge a part.
# [2026-10-24] dsp-rom-3606-dump.txt vient du depot fc-magnetite (freecalypso-env/),
# plus du FTP FreeCalypso (~18 Ko/s).
ARG FREECALYPSO_URL=https://raw.githubusercontent.com/bbaranoff/fc-magnetite/main/freecalypso-env
RUN if ! osmo-deb install c54x-exe 0.git; then \
      git clone https://github.com/bbaranoff/c54x_exe /opt/GSM/c54x_exe \
      && cd /opt/GSM/c54x_exe \
      && make QOSMO=/opt/GSM/qosmo \
             CFLAGS="-O3 -g -Wall -Werror=format -Werror=format-extra-args -Wno-unused-function
 ↪  -Wno-unused-variable -Wno-unused-but-set-variable -Wno-sign-compare" \
      && FREECALYPSO_URL="$FREECALYPSO_URL" bash rom/fetch-rom.sh --dest /opt/GSM --dest /opt/GSM/c54x_exe/rom \
      && curl -fsS --retry 2 -o /opt/GSM/calypso_dsp.txt "$FREECALYPSO_URL/dsp-rom-3606-dump.txt" \
      && cp /opt/GSM/calypso_dsp.txt rom/calypso_dsp.txt \
      && osmo-deb snapshot c54x-exe 0.git /opt/GSM/c54x_exe \
             /opt/GSM/calypso_dsp.PROM0.bin /opt/GSM/calypso_dsp.PROM1.bin /opt/GSM/calypso_dsp.PROM2.bin \
             /opt/GSM/calypso_dsp.PROM3.bin /opt/GSM/calypso_dsp.DROM.bin /opt/GSM/calypso_dsp.PDROM.bin \
             /opt/GSM/calypso_dsp.Registers.bin /opt/GSM/calypso_dsp.txt; \
    fi

RUN if ! osmo-deb install grgsm-exe 0.git; then \
      git clone https://github.com/bbaranoff/grgsm_exE /opt/GSM/grgsm_exe \
      && cd /opt/GSM/grgsm_exe \
      && make QOSMO=/opt/GSM/qosmo \
      && osmo-deb snapshot grgsm-exe 0.git /opt/GSM/grgsm_exe; \
    fi

# ──────────────────────────────────────────────────────────────────────
# STAGE grgsm-venv — GNU Radio 3.10 + gr-osmosdr + gr-gsm dans /root/.env
# ──────────────────────────────────────────────────────────────────────
# gr-gsm se lie a libosmocore/libosmocoding/libosmogsm (son CMakeLists les
# cherche par pkg-config) : ce stage DOIT donc partir d osmo-core, pas de base.
# Il reste parallele a `bb` et a `l1`, qui en derivent aussi.
FROM osmo-core AS grgsm-venv
# ARG ne traverse pas un FROM : les stages qui appellent osmo-deb ou
# ${ROOT} doivent les redeclarer, sinon ROOT est vide et
# OSMO_DEB_REFRESH=1 (build.sh --no-cache) ne descend pas jusqu ici.
ARG ROOT=/opt/GSM
ARG OSMO_DEB_REFRESH=0

# ── gr-gsm : GNU Radio 3.10 + gr-osmosdr + gr-gsm dans le venv /root/.env ────
# (= moteur de démod du SI réel utilisé par si_bridge.py / grgsm_decode).
# Les deps GNU Radio (apt build-dep) sont posees par l unique bloc apt en tete
# de fichier, deb-src compris : plus aucun apt ici.

# ── GNU Radio + gr-osmosdr + gr-gsm, en UN paquet : le venv /root/.env ─────────
# On TÉLÉCHARGE et on exécute CE script (le gist, pinné au commit fcdb409). Il
# cree /root/.env et y installe les trois. Puis :
#
# Patch gr-gsm : le receiver poste le BSIC/FN du SCH (decode_sch) sur le port
# `measurements` ET sur stdout ("SCHBSIC <bsic> <fn>"). Le shunt DSP le recoit
# (si_bridge.py parse le stdout de grgsm_decode -> UDP 4731 -> feed_sb) et encode
# le VRAI BSIC dans dispatch_sb (remplace SHUNT_CANNED_BSIC 63). Applique APRES le
# gist (qui clone+build gr-gsm propre), puis recompile/reinstalle dans le venv.
# Patch maintenu dans patches/ (regenere a chaque changement gr-gsm).
#
```

```
# matplotlib est consomme par tools/fft_global.sh et tools/matrix.sh.
# numpy RESTE en 1.x : le gist pose "numpy<2" (GNU Radio 3.10 compile contre),
# mais un `pip install matplotlib` nu tire la derniere matplotlib/contourpy, qui
# exigent numpy>=2, et pip remplace alors la 1.26 par une 2.x. La contrainte
# sur la meme ligne fait choisir a pip une matplotlib compatible numpy 1.
# Le paquet du cache passe en 1.git : le 0.git embarque un venv en numpy 2.
#
# Le paquet du cache (grgsm-venv) est le venv COMPLET, patch et matplotlib
# compris : c est pour cela que les trois etapes tiennent dans un seul RUN. Les
# arbres /opt/GSM/{gnuradio,gr-osmosdr,gr-gsm} ne sont pas dans le paquet - rien
# ne les lit au runtime (le venv porte ses .so avec un RPATH sur /root/.env).
COPY patches/grgsm-receiver-publish-bsic-fn.patch /tmp/grgsm-receiver-publish-bsic-fn.patch
RUN if ! osmo-deb install grgsm-venv 1.git; then \
      curl -fsSL
       ↪  https://gist.githubusercontent.com/bbaranoff/3683811057933af0954b661821e950d1/raw/fcdb4092483ec383440b67fc002db0c1583
       ↪  | bash \
      && git -C /opt/GSM/gr-gsm apply /tmp/grgsm-receiver-publish-bsic-fn.patch \
      && cd /opt/GSM/gr-gsm/build \
      && make -j"$(nproc)" \
      && make install \
      && . ~/.env/bin/activate && pip install matplotlib "numpy<2" \
      && osmo-deb snapshot grgsm-venv 1.git /root/.env /etc/ld.so.conf.d/gnuradio.conf; \
    fi && ldconfig

# ──────────────────────────────────────────────────────────────────────
# STAGE node — Node 22 et le dashboard web osmo-egprs-web
# ──────────────────────────────────────────────────────────────────────
FROM base AS node
# ARG ne traverse pas un FROM : les stages qui appellent osmo-deb ou
# ${ROOT} doivent les redeclarer, sinon ROOT est vide et
# OSMO_DEB_REFRESH=1 (build.sh --no-cache) ne descend pas jusqu ici.
ARG ROOT=/opt/GSM
ARG OSMO_DEB_REFRESH=0

# ═══════════════════════════════════════════════════════════════════════
# Node.js + dashboard web osmo-egprs-web (ex-Dockerfile.run, chaine complete)
# ═══════════════════════════════════════════════════════════════════════
# Place EN FIN DE FICHIER a dessein : aucune etape du build ne depend de Node,
# et la fin de fichier preserve le cache des couches longues de compilation.
# L'ORDRE INTERNE DE CETTE CHAINE EST CONTRAINT — voir les ⚠ ci-dessous.
# /opt/GSM et pas /opt : c'est /opt/GSM que l'ISO recupere en bloc
# (docker cp "$CID:/opt/GSM"). Le dashboard vivait a cote, dans /opt, et devait
# donc etre reclone une seconde fois a la construction de l'image - deux
# sources pour le meme depot, qui n'avancaient pas ensemble.
RUN if [ ! -d "/opt/GSM/osmo-egprs-web/" ]; then \
      mkdir -p /opt/GSM && \
      git clone -b main https://github.com/bbaranoff/osmo-egprs-web /opt/GSM/osmo-egprs-web; \
    fi

# ── Runtime Node.js + service web osmo-egprs-web ──────────────────────
# Le dashboard /opt/GSM/osmo-egprs-web/server.js tourne en mode NATIF (telnet VTY
# local, pas de docker) et est servi sur :8080. Le DÉMARRAGE est géré par
# start-native.sh (`systemctl restart osmo-egprs-web`) — ici on ne fait
# qu'INSTALLER le runtime + les dépendances + le unit (enable au boot).
#
# Node 22 (LTS « Jod ») — même famille majeure que l'ISO, qui installe
# nodesource setup_22.x (build-iso.sh). Garder les deux alignés : le dashboard
# est le même server.js des deux côtés.
ARG NODE_VERSION=v22.23.2
# Le tarball suit l architecture de l image (x64 sur amd64, arm64 sur un build
# --platform linux/arm64 pour le Raspberry Pi) : un node x64 dans une image
# arm64 ne se lancerait pas, et le dashboard avec lui.
RUN case "$(uname -m)" in x86_64) _na=x64 ;; aarch64) _na=arm64 ;; *) echo "arch node inconnue: $(uname -m)" >&2;
 ↪  exit 1 ;; esac && \
    curl -fsSL "https://nodejs.org/dist/${NODE_VERSION}/node-${NODE_VERSION}-linux-${_na}.tar.xz" \
        -o /tmp/node.tar.xz && \
```

```
    mkdir -p /opt/node && \
    tar -xJf /tmp/node.tar.xz -C /opt/node --strip-components=1 && \
    rm -f /tmp/node.tar.xz && \
    ln -sf /opt/node/bin/node /usr/local/bin/node && \
    ln -sf /opt/node/bin/npm  /usr/local/bin/npm && \
    ln -sf /opt/node/bin/npx  /usr/local/bin/npx && \
    node --version
# Unit systemd (fichier versionné dans le repo osmo-nitb-for-calypso, source unique).
# Le START reste géré par start-native.sh (`systemctl restart osmo-egprs-web`).
COPY services/osmo-egprs-web.service /etc/systemd/system/osmo-egprs-web.service
RUN ln -sf /etc/systemd/system/osmo-egprs-web.service \
        /etc/systemd/system/multi-user.target.wants/osmo-egprs-web.service

# ── Dashboard web : install-web-service.sh joue AU BOOT ──────────────────
# [2026-08-12] Remplace le geste manuel `bash /opt/GSM/osmo-egprs-web/install-web-service.sh`
# qu'il fallait refaire dans chaque conteneur pour armer le HTTPS.
#
# Ce qui reste au BUILD : node et le unit du service (plus haut), le
# `npm install` (juste en dessous). Ce qui passe au BOOT : le certificat TLS
# auto-signe — une cle privee generee au build serait la meme pour tous ceux qui tirent l'image. Le detail
# du raisonnement est dans services/osmo-egprs-web-install.service.
#
# ⚠ ORDRE : ce bloc DOIT rester APRES le clone de /opt/GSM/osmo-egprs-web
# (haut de cette section) — la COPY ci-dessous ecrit dans ce depot.
#
# SOURCE UNIQUE DU UNIT. install-web-service.sh installe le unit en le copiant
# depuis /opt/GSM/osmo-egprs-web/osmo-egprs-web.service — la copie du depot
# osmo-egprs-web, qui avait DIVERGE de celle-ci (elle avait perdu
# `CAP_IFACE=any`, donc la capture du dashboard). On ecrase donc cette copie par
# services/osmo-egprs-web.service : les deux emplacements servent desormais le
# meme fichier, et le script ne peut plus reintroduire la regression.
COPY services/osmo-egprs-web.service /opt/GSM/osmo-egprs-web/osmo-egprs-web.service
# Fail-fast : si l'amont retire le script, on le sait au build, pas au boot par
# un HTTPS muet.
RUN test -s /opt/GSM/osmo-egprs-web/install-web-service.sh
# Dependances JS (ws) — UN SEUL `npm install`, apres le clone.
# Non fatal (`|| true`) : node_modules est versionne dans le depot, un build
# hors-ligne reste valable. Mais la sortie est conservee, pas avalee.
RUN cd /opt/GSM/osmo-egprs-web && npm install --omit=dev --no-audit --no-fund || true
# ── L INSTALLATION EST JOUEE ICI, AU BUILD ──────────────────────────────
# [2026-08-31] On se contentait de VERIFIER que le script existe (le `test -s`
# ci-dessus) et on remettait tout au boot. L unite arrivait donc dans l image
# telle que le gabarit la portait, avec son ExecStart fige — et dans un
# conteneur, ou node vit en /usr/local/bin et non /usr/bin, systemd sortait en
#     status=203/EXEC
# en boucle, sans que rien ne parle d un chemin.
#
# Le script sait resoudre node et poser l unite avec le bon chemin : on le
# joue donc au build, pour que l image contienne une unite deja juste.
#
# `|| true` : dans un build il n y a PAS de systemd. Le script le sait et ne
# s arrete pas dessus (voir son en-tete), mais ses `systemctl` echouent
# forcement ; les avaler ici evite qu un detail d environnement fasse tomber
# une image de 11 Go. Ce qui compte - le unit, le certificat, le runtime - est
# ecrit avant ces appels.
#
# ⚠ ET IL N Y A PLUS DE REJEU AU BOOT. osmo-egprs-web-install.service a ete
# retire des trois chemins de deploiement, pour une raison simple : ce script
# TELECHARGE node quand il manque. Le laisser au demarrage faisait dependre
# d Internet un banc concu pour tourner isole - et un premier boot hors ligne
# repartait sans dashboard. Tout est desormais fige dans l image ; le
# demarrage ne fait plus que lancer osmo-egprs-web.service.
RUN bash /opt/GSM/osmo-egprs-web/install-web-service.sh || true

# ═══════════════════════════════════════════════════════════════════════
# STAGE lte — la 4G du banc : srsRAN_4G (ZeroMQ) et Open5GS dans /opt/LTE
```

```
# ──────────────────────────────────────────────────────────────────────
FROM base AS lte
# ARG ne traverse pas un FROM : les stages qui appellent osmo-deb ou
# ${ROOT} doivent les redeclarer, sinon ROOT est vide et
# OSMO_DEB_REFRESH=1 (build.sh --no-cache) ne descend pas jusqu ici.
ARG ROOT=/opt/GSM
ARG OSMO_DEB_REFRESH=0

# osmo-lte-install --configs lit $OSMO_REPO : ce stage a besoin du depot, et
# il ne derive pas de celui qui le clone. Le clone y est donc refait — c est
# quelques secondes, et l assemblage refait le sien, qui reste LA source pour
# le `git pull` de Dockerfile.run.
RUN git clone https://github.com/bbaranoff/osmo-operator /opt/GSM/osmo-operator

# ── LA 4G DU BANC : srsRAN_4G (ZeroMQ) et Open5GS, dans /opt/LTE ──────────
# [2026-09-08] DEUX CHEMINS VERS LES MEMES PAQUETS (voir packaging/
# snapshot-lte-debs.sh) : le raccourci photographie le natif de la machine de
# reference en osmo-build-libzmq / osmo-build-srsran / osmo-build-open5gs, et
# le cache .deb les rend a `osmo-deb install` ci-dessous - rien n est alors
# compile. Cache vide : on compile depuis les sources, par le MEME script que
# le natif (tools/osmo-lte-install.sh --build, OSMO_DEB=1 -> osmo-deb pack).
# libzmq vient d Ubuntu (libzmq3-dev, pose par `--deps` ci-dessous) : le
# /opt/LTE/libzmq compile a la main n a de raison d etre que sur le natif.
# Open5GS : prefixe /opt/LTE/open5gs/install (bin, etc/open5gs, var/log) - pas
# /root.
#
# LES CONFIGS ET LES LANCEURS AUSSI (`--configs --launchers`, en fin de RUN).
# [2026-09-09] Ils n etaient poses que dans l ISO (88-lte-pmos.sh) et en natif
# (addition.sh), « pas ici : l image docker n a pas de HOME de session a
# servir ». Faux pour le conteneur : il tourne en root, et /root/.config/srsran
# EST le HOME que srsenb/srsue lisent. L image sortait donc avec les binaires
# 4G mais sans mme.yaml (donc sans SGs, donc sans CSFB), sans enb.conf, et
# sans la commande osmo-lte : une 4G qu on ne pouvait pas lancer. Et le .deb
# Open5GS n apporte AUCUN yaml (ninja install sous DESTDIR ne pose pas les
# exemples : 128 fichiers, tous dans bin/ et lib/) - les configs du depot
# sont la seule source. Elles viennent du clone GitHub de /opt/GSM/
# osmo-operator (ligne « git clone » plus haut) : ce qui est pousse est ce que
# le conteneur voit. Les abonnes (configs/open5gs/dump, subscribers.json) ne
# sont pas poses au build : osmo-epc start les restaure dans MongoDB.
#
# ⚠ `--deps` D ABORD, TOUJOURS (pas seulement quand on compile).
# [2026-09-09] Le commentaire ci-dessus affirmait que libzmq3-dev etait « dans
# la liste apt plus haut » : il n y etait pas, ni lui ni libmbedtls-dev,
# libboost-program-options-dev, libconfig++-dev, meson, flex, bison, libmongoc.
# Le build tombait donc net des le cmake de srsRAN :
#     Could NOT find MbedTLS (missing: MBEDTLS_LIBRARIES MBEDTLS_INCLUDE_DIRS)
# La liste apt du haut de ce fichier est celle de la pile Osmocom (et elle est
# RELUE par install_modules/10-deps.sh pour le natif) : la 4G a la sienne,
# tenue dans tools/osmo-lte-install.sh (lte_deps), une seule fois pour les
# trois chemins. On l appelle, on ne la recopie pas.
# Inconditionnel parce que ces paquets sont aussi le RUNTIME : les .deb du
# cache ne portent que les binaires (osmo-deb pack = un tar), pas leurs
# dependances - srsenb sorti du cache reclame quand meme libmbedtls, libzmq5,
# libconfig++ et boost_program_options. `--deps` tire aussi mongodb-org (base
# d abonnes du HSS/PCRF) et n est fatal sur aucun paquet.
COPY tools/osmo-lte-install.sh /usr/local/sbin/osmo-lte-install
RUN --mount=type=cache,id=osmo-apt-archives,target=/var/cache/apt/archives,sharing=locked \
    chmod 755 /usr/local/sbin/osmo-lte-install && \
    OSMO_REPO=/opt/GSM/osmo-operator osmo-lte-install --deps && \
    { osmo-deb install libzmq 4.3.5+git || true; } && \
    { osmo-deb install srsgui 0.1+git || true; } && \
    SRS_DEB_VER="$(osmo-lte-install --srs-deb-version)" && \
    if ! osmo-deb install srsran "$SRS_DEB_VER"; then \
        OSMO_DEB=1 OSMO_REPO=/opt/GSM/osmo-operator osmo-lte-install --build || { echo "ECHEC build srsRAN"; exit
        ↪  1; }; \
    fi && \
```

```
    if ! osmo-deb install open5gs 2.8.0+git; then \
        OSMO_DEB=1 OSMO_REPO=/opt/GSM/osmo-operator osmo-lte-install --build || { echo "ECHEC build Open5GS"; exit
         ↪  1; }; \
    fi && \
    OSMO_REPO=/opt/GSM/osmo-operator osmo-lte-install --configs --launchers && \
    ldconfig && test -x /usr/local/bin/srsenb && test -x /opt/LTE/open5gs/install/bin/open5gs-mmed \
    && test -s /opt/LTE/open5gs/install/etc/open5gs/mme.yaml && grep -q '^  sgsap:'
     ↪  /opt/LTE/open5gs/install/etc/open5gs/mme.yaml \
    && test -s /opt/LTE/open5gs/install/etc/freeDiameter/mme.conf && test -s
     ↪  /opt/LTE/open5gs/install/etc/open5gs/tls/mme.crt \
    && test -s /root/.config/srsran/enb.conf && test -x /usr/local/bin/osmo-lte && test -x /usr/local/bin/osmo-epc

# ── L UI SMARTPHONE : pmbootstrap PATCHE, dans /opt/user_interface/pmos ────────
# Le telephone du banc est une VM postmarketOS lancee par pmbootstrap (patche
# pour le modem serie PCI, le son du banc, la taille d ecran, et le port 5038
# qui n est pas un adb : patches/pmbootstrap-osmo-bench-qemu.patch). Le clone
# et le patch sont faits ici, une fois, et sortent en osmo-build-pmbootstrap :
# l ISO (50-injection-image.sh) le pose tel quel, tools/osmo-pmos-install.sh
# le trouve en place et n a plus rien a cloner. Le NOYAU PPP, lui, ne se
# construit pas ici (pmbootstrap, non root, montages de boucle) : il arrive
# par osmo-build-pmos-kernel (packaging/build-pmos-kernel-deb.sh, ou GitHub
# bbaranoff/pmos_ppp_kernel), pose par 88-lte-pmos.sh.
COPY patches/pmbootstrap-osmo-bench-qemu.patch /tmp/pmbootstrap-osmo-bench-qemu.patch
RUN if ! osmo-deb install pmbootstrap 0.git; then \
        mkdir -p /opt/user_interface/pmos && \
        git clone https://gitlab.postmarketos.org/postmarketOS/pmbootstrap.git
         ↪  /opt/user_interface/pmos/pmbootstrap && \
        cd /opt/user_interface/pmos/pmbootstrap && \
        git apply /tmp/pmbootstrap-osmo-bench-qemu.patch && \
        grep -q osmo_bench_args pmb/commands/qemu.py && \
        osmo-deb snapshot pmbootstrap 0.git /opt/user_interface/pmos/pmbootstrap; \
    fi && test -f /opt/user_interface/pmos/pmbootstrap/pmbootstrap.py

# ============================================================================
# STAGE osmocom-nitb — L ASSEMBLAGE. C est l image publiee.
# ============================================================================
# Rien ne se compile ici : on repose les .deb produits par les stages, puis on
# ajoute ce qu aucun paquet ne porte (Node, le dashboard, la 4G, le depot, les
# configs du contexte). Le nom osmocom-nitb reste sur CETTE etape : build.sh
# (compose_build) et compose.yaml passent --target osmocom-nitb.
FROM base AS osmocom-nitb
ARG ROOT=/opt/GSM

# ── 1. Les paquets de tous les stages ──────────────────────────────────────
# bb DERIVE de osmo-core et l1 DERIVE de qemu : leur cache porte deja celui de
# leur parent, d ou cinq COPY et non sept.
COPY --from=bb         /var/cache/osmo-debs/ /var/cache/osmo-debs/
COPY --from=l1         /var/cache/osmo-debs/ /var/cache/osmo-debs/
COPY --from=grgsm-venv /var/cache/osmo-debs/ /var/cache/osmo-debs/
COPY --from=lte        /var/cache/osmo-debs/ /var/cache/osmo-debs/
# (pas de COPY depuis `node` : ce stage n appelle jamais osmo-deb - Node est un
#  tarball et le dashboard un clone git. Son cache n est que celui de base.)

# ⚠ L ORDRE DE POSE COMPTE POUR LES QUATRE PAQUETS PATCHES.
# osmo-trx, osmo-bts, osmo-hlr et osmo-msc existent en DEUX versions portant le
# MEME nom de paquet : celle de la boucle (osmo-trx 1.7.2) et la patchee
# (1.7.2+ipc). Dans la chaine lineaire d avant, la patchee etait posee apres,
# donc elle gagnait. Ici le glob les pose dans l ordre ASCII, ou « + » (0x2B)
# precede « ~ » (0x7E) : osmo-trx_1.7.2+ipc passe AVANT osmo-trx_1.7.2, et
# c est la version NON patchee qui finissait installee - RACH sans table per-RA,
# BTS/HLR/MSC sans le forcage du RAND, et une LU qui ne passe plus. Silencieux :
# le binaire est la, il est juste le mauvais.
# On repose donc les quatre explicitement, en dernier.
RUN set -eux; \
    dpkg -i --force-overwrite /var/cache/osmo-debs/osmo-build-*.deb; \
    dpkg -i --force-overwrite \
```

```
        /var/cache/osmo-debs/osmo-build-osmo-trx_1.7.2+ipc~*.deb \
        /var/cache/osmo-debs/osmo-build-osmo-bts_1.10.0+rand~*.deb \
        /var/cache/osmo-debs/osmo-build-osmo-hlr_1.9.2+rand~*.deb \
        /var/cache/osmo-debs/osmo-build-osmo-msc_1.15.0+rand~*.deb; \
    ldconfig

# ── 2. Ce qu aucun .deb ne porte ──────────────────────────────────────
# Node et le dashboard : le runtime est un tarball detarre dans /opt/node, le
# dashboard un clone git avec son node_modules. Ni l un ni l autre ne passe par
# osmo-deb, et install-web-service.sh a deja ete joue dans le stage node.
COPY --from=node /opt/node                                  /opt/node
COPY --from=node /opt/GSM/osmo-egprs-web                    /opt/GSM/osmo-egprs-web
COPY --from=node /etc/systemd/system/osmo-egprs-web.service /etc/systemd/system/osmo-egprs-web.service
RUN ln -sf /opt/node/bin/node /usr/local/bin/node \
    && ln -sf /opt/node/bin/npm  /usr/local/bin/npm \
    && ln -sf /opt/node/bin/npx  /usr/local/bin/npx \
    && ln -sf /etc/systemd/system/osmo-egprs-web.service \
        /etc/systemd/system/multi-user.target.wants/osmo-egprs-web.service \
    && node --version

# Chemin EXPLICITE. Sans lui, ce clone heritait du WORKDIR /etc/osmocom (pose
# beaucoup plus haut, jamais remis a ${ROOT}) et atterrissait dans
# /etc/osmocom/osmo-operator — masque au runtime par le montage de start-docker.sh l.505,
# et surtout PAS la ou Dockerfile.run va faire son `git pull`. Deux arbres, deux
# HEAD, un seul utilise. /opt/GSM/osmo-operator est le chemin nominal, teste en
# premier par build-iso.sh l.964 et update.sh l.329.
# On reste sur la branche par defaut du depot (main) : pas de checkout explicite,
# donc pas de ref a maintenir ici, et HEAD est attache — ce dont le `git pull`
# de Dockerfile.run a besoin.
RUN git clone https://github.com/bbaranoff/osmo-operator /opt/GSM/osmo-operator

COPY tools/osmo-lte-install.sh /usr/local/sbin/osmo-lte-install
RUN chmod 755 /usr/local/sbin/osmo-lte-install

# ── La 4G ─────────────────────────────────────────────────────────────
# /opt/LTE en entier, et pas seulement ce que portent les .deb : le prefixe
# d installation d Open5GS vit SOUS l arbre de sources (/opt/LTE/open5gs/install),
# et /opt/LTE n est pas sous /opt/GSM - osmo-deb pack n y ajoute donc aucune
# source (SRC_ROOT). Sans cette copie, l image aurait les binaires 4G sans un
# seul arbre pour les recompiler.
COPY --from=lte /opt/LTE /opt/LTE

# ⚠ `--deps` DOIT ETRE REJOUE ICI. Ce sont AUSSI les paquets de RUNTIME
# (libzmq5, libmbedtls, libconfig++, boost_program_options, mongodb-org...) et
# les .deb d osmo-deb ne declarent aucune dependance - leur en-tete le dit. Les
# poser dans le seul stage `lte` donnait une image ou srsenb et open5gs-mmed
# existent mais ne demarrent pas, faute de .so. Le montage du cache apt fait que
# rien n est retelecharge : les paquets sont deja dans le cache BuildKit, poses
# par le stage lte.
#
# `--configs --launchers` ensuite, dans le meme ordre que le stage lte : les yaml
# d Open5GS, enb.conf et les deux lanceurs s ecrivent HORS de tout DESTDIR, donc
# aucun .deb ne les porte. Ils lisent /opt/GSM/osmo-operator, clone juste au-dessus.
RUN --mount=type=cache,id=osmo-apt-archives,target=/var/cache/apt/archives,sharing=locked \
    OSMO_REPO=/opt/GSM/osmo-operator osmo-lte-install --deps && \
    OSMO_REPO=/opt/GSM/osmo-operator osmo-lte-install --configs --launchers && \
    ldconfig && test -x /usr/local/bin/srsenb \
    && test -x /opt/LTE/open5gs/install/bin/open5gs-mmed \
    && test -s /opt/LTE/open5gs/install/etc/open5gs/mme.yaml && grep -q '^  sgsap:'
     ↪  /opt/LTE/open5gs/install/etc/open5gs/mme.yaml \
    && test -s /opt/LTE/open5gs/install/etc/freeDiameter/mme.conf \
    && test -s /opt/LTE/open5gs/install/etc/open5gs/tls/mme.crt \
    && test -s /root/.config/srsran/enb.conf && test -x /usr/local/bin/osmo-lte && test -x /usr/local/bin/osmo-epc

# Les alternatives gcc : elles ont ete DECLAREES dans le stage bb, qui ne
# transmet que ses .deb. On les redeclare ici pour que l image finale offre le
```

```
# meme choix, et on reste sur gcc-11 - le defaut d avant le decoupage.
RUN update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-9 90 \
       --slave /usr/bin/g++ g++ /usr/bin/g++-9 \
    && update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-11 110 \
       --slave /usr/bin/g++ g++ /usr/bin/g++-11 \
    && update-alternatives --set gcc /usr/bin/gcc-11

# 4. Installation des fichiers du projet
WORKDIR /etc/osmocom
COPY scripts/. /etc/osmocom/
COPY configs/*cfg /etc/osmocom/
RUN mv /etc/osmocom/run.sh /root/run.sh
# Copie des binaires vers /usr/bin pour systemd et installation des .service
# (La `COPY scripts/gapk-start.sh /etc/osmocom/gapk-start.sh` qui etait ici est
#  supprimee : `COPY scripts/. /etc/osmocom/` ci-dessus depose deja le fichier.
#  Le RUN ci-dessous reste : le symlink et /var/lib/gapk sont uniques.)
RUN chmod +x /etc/osmocom/gapk-start.sh && \
    ln -sf /etc/osmocom/gapk-start.sh /usr/local/bin/gapk-start.sh && \
    mkdir -p /var/lib/gapk

RUN cp -f /usr/local/bin/osmo* /usr/bin/ || true && \
    cp -f /usr/local/bin/proto-smsc-* /usr/bin/ || true && \
    cp -f /usr/local/bin/sms-* /usr/bin/ || true && \
    cp -f /usr/local/bin/gen-sms-* /usr/bin/ || true && \
    # Si tu as des fichiers .service dans configs/
    cp /etc/osmocom/configs/*.service /lib/systemd/system/ 2>/dev/null || true

# ── Vérification binaires proto-SMSC ──────────────────────────────
# Ex-Dockerfile.run. Premier point du build ou les deux chemins testes sont
# peuples (produits par gsup-smsc-proto, recopies dans /usr/bin juste au-dessus).
RUN which proto-smsc-daemon && which proto-smsc-sendmt && \
    echo "proto-smsc binaries OK" || \
    echo "WARNING: proto-smsc binaries not found in PATH"

# 5. Fix Permissions & Systemd (Status 214/217)
RUN sed -i 's/^CPUScheduling/#CPUScheduling/g' /lib/systemd/system/osmo-*.service && \
    sed -i 's/User=osmocom/User=root/g' /lib/systemd/system/osmo-ggsn.service && \
    sed -i 's/User=osmocom/User=root/g' /lib/systemd/system/osmo-sgsn.service && \
    chmod +x /etc/osmocom/*.sh

# Activation du service et nettoyage
# Les unites systemd vivent dans services/ ; elles doivent atterrir dans
# /etc/systemd/system/, seul repertoire ou systemd les cherche.
COPY services/osmo-bts-trx.service /etc/systemd/system/osmo-bts-trx.service
RUN systemctl enable osmo-bts-trx.service && \
    passwd -d root && \
    systemctl mask getty@tty1.service serial-getty@tty1.service

# Binaires cote hote produits par osmocom-bb, deposes en une seule couche.
# `install -m755` plutot que `cp` : les droits sont explicites, pas herites.
RUN mkdir -p /root/.osmocom/bb/ \
    && install -m755 ${ROOT}/osmocom-bb/src/host/trxcon/src/trxcon          /usr/local/bin/trxcon \
    && install -m755 ${ROOT}/osmocom-bb/src/host/layer23/src/mobile/mobile  /usr/local/bin/mobile \
    && install -m755 ${ROOT}/osmocom-bb/src/host/virt_phy/src/virtphy       /usr/local/bin/virtphy
# ccch_scan n'est plus installe ici : la branche fixeria/burst_ind, en fin de
# fichier, ecrase de toute facon /usr/local/bin/ccch_scan par SA version (cp sans
# `|| true`, donc obligatoire). Le binaire de master ne survivait jamais — c'est
# la version burst_ind qui fait foi.

# Confort du shell interactif. Pas de `source` ici : chaque RUN est un shell neuf,
# ce serait sans effet — le fichier est lu a l'ouverture d'une session.
RUN echo "alias faketrx='python3 ${ROOT}/osmocom-bb/src/target/trx_toolkit/fake_trx.py'" >> ~/.bashrc
# Prompt : root(rouge)@(bleu)<nom-container=\h>(jaune):<dossier courant>(vert)#
# \h = nom du container grâce à `docker run --hostname "$container_name"` (start-docker.sh).
# [refactor] LE PS1 ETAIT DEFINI DEUX FOIS, avec deux formats DIFFERENTS : ici
# et dans Dockerfile.run. Bash lit ~/.bashrc de haut en bas -> c'est celui de
```

```
# Dockerfile.run qui gagnait, et le prompt reellement affiche etait
# `user@host:cwd☎#`, pas celui decrit deux lignes plus haut. On ne garde QUE le
# gagnant, a l'identique, pour ne rien changer au prompt visible ; l'explication
# du `\h` ci-dessus reste vraie pour les deux formats.
# ── Prompt interactif (root rouge / hostname bleu / cwd vert / ☎#) ──────────
RUN printf '%s\n' '' \
    '### calypso-prompt ###' \
    "export PS1='\[\033[1;31m\]\u\[\033[0m\]@\[\033[1;34m\]\h\[\033[0m\]:\[\033[1;32m\]\w\[\033[0m\]☎# '" \
    '### end calypso-prompt ###' >> /root/.bashrc
COPY configs/mobile.cfg /root/.osmocom/bb/mobile.cfg
RUN chmod +x /root/run.sh

# Répertoires pour le proto-SMSC + arborescence Asterisk (ex-Dockerfile.run,
# ou ce mkdir faisait doublon avec celui-ci ; seul le chmod 755 etait unique).
# Position : apres l'installation d'asterisk (liste apt) et apres WORKDIR /etc/osmocom.
RUN mkdir -p /var/log/osmocom /var/run/smsc \
        /etc/asterisk /var/lib/asterisk /var/log/asterisk /var/run/asterisk \
        /root/.osmocom/bb && \
    chmod 755 /etc/asterisk /var/lib/asterisk /var/log/asterisk \
        /var/run/asterisk /var/log/osmocom

# ── PulseAudio system-mode : config bakée au build ──────────────────────────
# (ex-Dockerfile.run — depend du paquet pulseaudio de la liste apt en tete ;
#  place ici, tard, pour ne pas invalider le cache des ~200 lignes de compilation.)
# Cause racine du warning "[audio] PulseAudio injoignable — audio dégradé" :
# le démon system-mode lancé au runtime tournait SANS auth-anonymous (le patch
# de start-native.sh arrivait après son démarrage) → 'pactl' renvoyait
# "Access denied", et un 2e démon ne pouvait pas démarrer ("Daemon already
# running"). On bake ici la config pour que TOUT démon soit joignable dès son
# lancement : socket anonyme + null-sinks gsm_audio/gsm_mic + root dans
# pulse-access.
#
# [2026-08-12] gsm_mic AJOUTÉ ICI. Il n'était créé que par
# scripts/pulse-gsm-setup.sh, appelé tard dans run.sh (après osmo-start.sh) :
# un HLR qui rate coupe run.sh (set -e) et le sink n'existe jamais. Or
# configs/asound.conf fait pointer la CAPTURE gsm_in sur gsm_mic.monitor, et
# gapk_io abandonne lecture ET capture si la capture échoue → appel muet des
# deux côtés. Les deux sinks doivent naître avec le démon, pas avec un script.
RUN set -eux; \
    usermod -aG pulse-access root 2>/dev/null || true; \
    test -f /etc/pulse/system.pa; \
    sed -i 's|^#\?load-module module-native-protocol-unix.*|load-module module-native-protocol-unix
     ↪  auth-anonymous=1 socket=/var/run/pulse/native|' /etc/pulse/system.pa; \
    grep -q 'auth-anonymous=1' /etc/pulse/system.pa \
      || echo 'load-module module-native-protocol-unix auth-anonymous=1 socket=/var/run/pulse/native' >>
       ↪  /etc/pulse/system.pa; \
    grep -q 'sink_name=gsm_audio' /etc/pulse/system.pa \
      || echo 'load-module module-null-sink sink_name=gsm_audio format=s16le rate=8000 channels=1
       ↪  sink_properties=device.description=GSM_Audio' >> /etc/pulse/system.pa; \
    grep -q 'sink_name=gsm_mic' /etc/pulse/system.pa \
      || echo 'load-module module-null-sink sink_name=gsm_mic format=s16le rate=8000 channels=1
       ↪  sink_properties=device.description=GSM_Mic' >> /etc/pulse/system.pa; \
    grep -q 'sink_name=osmo_tts_off' /etc/pulse/system.pa \
      || echo 'load-module module-null-sink sink_name=osmo_tts_off format=s16le rate=8000 channels=1
       ↪  sink_properties=device.description=TTS_off' >> /etc/pulse/system.pa; \
    sed -i 's|^load-module module-suspend-on-idle|#load-module module-suspend-on-idle|' /etc/pulse/system.pa; \
    mkdir -p /var/run/pulse && chown -R pulse:pulse /var/run/pulse

# Dernier maillon de la chaine venv/gr-gsm (ex-Dockerfile.run) : le profil de
# root active le venv.
RUN echo 'source ~/.env/bin/activate' >> ~/.bashrc

# ── scripts bridge camping -> /opt/GSM (sinon /opt/GSM/qosmo/run.sh casse) :
# si_bridge.py (full SI set -> 4730 -> shunt feed_si), si_bridge_loop.sh,
# record_drain.py (iq_record.fifo -> record.cfile), grgsm_fft_live.py.
COPY opt-gsm/. /opt/GSM/
```

```
# --- Metadonnees de l'image ------------------------------------------------
# Regroupees a la fin : elles decrivent le conteneur qui tournera, pas une etape
# de construction. SIGRTMIN+3 est le signal d'arret propre de systemd.
STOPSIGNAL SIGRTMIN+3
ENTRYPOINT ["/etc/osmocom/entrypoint.sh"]
CMD ["/bin/bash"]

# ── L'ETAPE `debs` : le cache .deb seul, rien d'autre ──────────────────────
# ATTENTION - `osmocom-nitb` N'EST PLUS LA DERNIERE ETAPE de ce fichier. Tout
# ce qui construit l'image doit donc dire `--target osmocom-nitb` : build.sh
# (compose_build), compose.yaml (build.target du service nitb),
# iso_modules/21-docker-build.sh, .github/workflows/docker.yml. Sans ce
# --target, docker batit `debs` et sort une image VIDE de 2 Go de paquets.
#
# Pourquoi cette etape existe. En CI, l'image de base n'est plus chargee dans
# le docker du runner : elle part directement vers GHCR (buildx --push), parce
# que la materialiser localement demandait de tenir DEUX fois 11 Go et tuait le
# job en « exporting layers » (exit 143). Mais le workflow doit quand meme
# recuperer les .deb que le build vient de compiler, pour les rendre a
# actions/cache - et sans image locale, plus de `docker create` ni de
# `docker cp` (l'equivalent de debs_from_image, build.sh:306).
#
# D'ou cette etape : une seconde passe `--target debs --output type=local`
# reutilise INTEGRALEMENT le cache de la premiere (meme builder, memes couches,
# tout en hit) et n'ecrit sur le disque du runner que /var/cache/osmo-debs -
# quelques Go de .deb au lieu de l'image entiere. Sur l'hote, build.sh continue
# de passer par debs_from_image : il a l'image sous la main, lui.
FROM scratch AS debs
COPY --from=osmocom-nitb /var/cache/osmo-debs/ /
```

3.3 /opt/GSM/osmo-operator/Dockerfile.lite

8376 octets, 143 lignes → 143 lignes

```
# ============================================================================
# Dockerfile.lite - l'image d'EXECUTION : les binaires, sans les sources
# ============================================================================
#
# osmocom-nitb pese ~11 Go, dont 4 Go de /opt/GSM : les arbres de construction
# de gnuradio, gr-gsm, libosmocore, qemu, osmo-* ... Ces arbres ont servi a
# COMPILER ; ce qui tourne ensuite vit dans /usr/local/bin (119 Mo) et
# /usr/local/lib (17 Mo). Les garder revient a trimballer l'atelier avec la
# machine.
#
# POURQUOI DEUX ETAPES ET UN APLATISSEMENT
# Effacer dans une couche supplementaire ne rend RIEN : une image est la somme
# de ses couches, et les fichiers effaces continuent d'occuper celles d'avant.
# On elague donc dans une etape (prune), puis on recopie la racine obtenue dans
# une image vierge : seule la version elaguee est ecrite, et l'espace est
# reellement rendu.
#
# On repart de la racine COMPLETE plutot que de reconstruire une base a partir
# d'une liste de paquets : les binaires de /usr/local dependent de dizaines de
# bibliotheques posees par apt (boost, volk, fftw, sofia-sip...). Une liste
# reecrite a la main s'en approche, ne les couvre jamais toutes, et la panne
# n'apparait qu'a l'execution - un binaire qui ne demarre pas, faute d'un .so.
# Ici, tout ce qui n'est pas une source reste EXACTEMENT ce qu'il etait.
#
# Ce qui est GARDE dans /opt/GSM : ce que la pile lance vraiment.
#   qosmo/                          l'arbre ENTIER, moins ses objets de compilation.
#                                [2026-08-27] Avant, on n'en gardait que quatre
#                                morceaux (le binaire, pc-bios, tools, le
#                                firmware compal_e88). Il manquait ce qui LANCE :
```

```
#                               run.sh, run_modules/, environnement/, cfgs/,
#                               scripts/. start-native.sh delegue a
#                               $OQC_ROOT/run.sh - absent, l'image lite savait
#                               emuler un Calypso mais pas le demarrer.
#                               build/ ne garde que qemu-system-arm : 1,5 Go de
#                               .o et de .a en moins, et .git reste coherent
#                               avec son arbre (build/ est dans son .gitignore).
#   osmocom-bb/src/host/osmocon/osmocon       le chargeur
#   osmocom-bb/src/target/trx_toolkit         fake_trx.py et compagnie
#   c54x_exe/, grgsm_exe/                     les couches 1 hors QEMU
#   calypso_dsp.*.bin                         les ROM du DSP
#   firmware/, qemu-install/, osmo-operator/     images, qemu installe, ce depot
#
# LES DEPOTS GARDENT LEUR .git - qosmo comme osmo-operator. Ce n'est pas un
# oubli : c'est lui qui permet "osmo-update" (git fetch en place) sur la machine
# qui tourne, au lieu d'un effacement suivi d'un reclone. Les 96 Mo qu'il pese
# ne sont rien a cote des 4 Go d'ateliers que cette image retire.
#
# Construction :  ./build.sh --lite      (ou docker build -f Dockerfile.lite .)
# L'ISO correspondante :  ./build-iso.sh --role=operator --lite
# ============================================================================

ARG BASE=osmocom-nitb

# ── Etape 1 : elagage ──────────────────────────────────────────────────
FROM ${BASE} AS prune

# set -e implicite (SHELL par defaut) : si l'elagage echoue, l'image ne sort pas
# a moitie taillee. Tout se fait dans UNE couche - c'est son etat final qui sera
# recopie, pas les etapes intermediaires.
RUN set -eux; \
    cd /opt/GSM; \
    mkdir -p /keep/qosmo/build /keep/osmocom-bb/src/host/osmocon \
             /keep/osmocom-bb/src/target; \
    # qosmo : l'arbre entier, sauf build/. Le binaire est mis de cote AVANT
    # la copie, et --exclude=./build empeche le tar de le recouvrir ensuite.
    cp -a qosmo/build/qemu-system-arm  /keep/qosmo/build/ 2>/dev/null || true; \
    # [2026-08-28] qemu-bundle SUIT le binaire, et ce n'est pas un detail de
    # confort : c'est par LUI que QEMU trouve ses donnees quand il est lance
    # depuis un arbre de compilation. QEMU 8 n'utilise pas le prefix compile
    # tel quel - il le RELOCALISE par rapport a l'endroit ou le binaire se
    # trouve reellement, et dans un arbre de build cette relocalisation passe
    # par build/qemu-bundle/, qui rejoue l'arborescence d'installation.
    #
    # En le supprimant avec les 1,5 Go d'objets, l'image lite laissait un
    # qemu-system-arm incapable de lire son propre keymap :
    #
    #     qemu-system-arm: could not read keymap file: 'en-us'
    #
    # QEMU mourait AVANT d'allouer ses PTY, et start-native.sh s'arretait sur
    #     [FAIL] Emulator serial PTY (... s'est arrete avant d'allouer son PTY)
    # c'est-a-dire une image, et une ISO, sans MS. Le fichier existait pourtant
    # en trois exemplaires sur la machine - a un chemin que QEMU ne consulte
    # jamais. Diagnostique sur banc le 2026-08-28 : sans qemu-bundle rc=1 et
    # QEMU meurt, avec lui il tourne.
    #
    # Ce que ca coute : 146 Ko, 175 entrees dont 140 liens symboliques. C'est
    # la difference entre l'image complete, ou le Calypso demarre, et la lite,
    # ou il ne demarrait pas.
    cp -a qosmo/build/qemu-bundle      /keep/qosmo/build/ 2>/dev/null || true; \
    tar -C qosmo --exclude=./build -cf - . | tar -C /keep/qosmo -xf -; \
    # [2026-09-25] c54x_exe (--dsp) et grgsm_exe : petits arbres (binaire +
    # run.sh + rom/), gardes ENTIERS, .git compris (osmo-update). Leur source
    # C vit dans qosmo, garde ci-dessus. qosmo-grgsm/qosmo-dsp sont retires.
    for d in c54x_exe grgsm_exe; do \
        [ -d "$d" ] && cp -a "$d" /keep/ || true; \
```

```
    done; \
    cp -a osmocom-bb/src/host/osmocon/osmocon \
          /keep/osmocom-bb/src/host/osmocon/ 2>/dev/null || true; \
    cp -a osmocom-bb/src/target/trx_toolkit \
          /keep/osmocom-bb/src/target/ 2>/dev/null || true; \
    cp -a *.bin /keep/ 2>/dev/null || true; \
    for d in firmware qemu-install osmo-operator pont; do \
        [ -e "$d" ] && cp -a "$d" /keep/ || true; \
    done; \
    # La coupe : tout /opt/GSM disparait, seuls les artefacts reviennent.
    cd /; rm -rf /opt/GSM; mkdir -p /opt/GSM; \
    cp -a /keep/. /opt/GSM/; rm -rf /keep; \
    # Reste de l'atelier : en-tetes, bibliotheques statiques, objets, caches.
    # /var/cache/osmo-debs : le cache des paquets compiles (packaging/osmo-deb.sh)
    # est un outil de CONSTRUCTION - l arbre QEMU en zstd, le venv gr-gsm... Il
    # vit sur l hote ; dans l image d execution elaguee il n a pas sa place.
    rm -rf /usr/local/include /root/.cache /root/.cargo /root/.rustup \
           /usr/share/doc /usr/share/man /usr/share/locale \
           /var/lib/apt/lists/* /var/cache/apt/archives/*.deb \
           /var/cache/osmo-debs /var/cache/apt/apt-fast \
           /tmp/* /var/tmp/*; \
    find /usr/local/lib -name '*.a' -delete 2>/dev/null || true; \
    # Le balayage des .o EVITE /opt/GSM. [2026-08-27] Il y passait, et
    # l ancien qosmo-grgsm/tools/calypso-ipc-device/*.o etaient SUIVIS PAR GIT :
    # les effacer laissait un arbre que git voit comme modifie, et "osmo-update"
    # refuse alors de l'avancer - a raison, il ne sait pas ce qu'on y a fait.
    # Ce qui restait a couper est de toute facon parti avec l'atelier.
    find / -xdev -path /opt/GSM -prune -o -name '*.o' -exec rm -f {} + 2>/dev/null || true; \
    du -sh /opt/GSM /usr/local/bin /usr/local/lib

# ── Etape 2 : aplatissement ──────────────────────────────────────
# FROM scratch + COPY de la racine : une seule couche, qui ne contient que
# l'etat elague. C'est ce qui fait reellement tomber la taille.
FROM scratch
COPY --from=prune / /

# La configuration ne traverse pas un FROM scratch : on la redeclare a
# l'identique de l'image d'origine, sinon l'image lite ne demarre pas pareil.
ENV PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin \
    container=docker \
    OSMO_IMAGE=1 \
    PKG_CONFIG_PATH=/usr/local/lib/pkgconfig \
    LD_LIBRARY_PATH=/usr/local/lib \
    DEBIAN_FRONTEND=noninteractive

WORKDIR /etc/osmocom
ENTRYPOINT ["/etc/osmocom/entrypoint.sh"]
CMD ["/bin/bash"]
```

3.4 /opt/GSM/osmo-operator/Dockerfile.run

13302 octets, 194 lignes → 194 lignes

```
# Dockerfile.run — COUCHE DE MISE A JOUR (image d'execution osmocom-run)
# ──────────────────────────────────────────────────────────────────
# Ce fichier ne CONSTRUIT rien. Il repart de l'image de base osmocom-nitb
# (Dockerfile : ~11 Go, ~40 min de compilation) et n'y rafraichit que ce qui
# bouge a chaque iteration du lab :
#   - les scripts et les configs du depot (COPY, quelques secondes) ;
#   - le pont TRX pont/pont.py, source iteree a chaud au meme titre ;
#   - les arbres git qosmo, c54x_exe, grgsm_exe et osmo-operator (git pull + rebuild).
#
# Tout le reste vit desormais dans Dockerfile : paquets apt, Node/dashboard web,
# firmware Calypso, config PulseAudio, prompt, repertoires, units systemd,
```

```
# verification proto-smsc, dsp_txt2bin, pip matplotlib. IL N'Y A PLUS UN SEUL
# `apt` ICI — l'image d'execution n'a plus a connaitre apt du tout.
#
# Le recouvrement des COPY avec Dockerfile est VOULU, ce n'est pas un doublon a
# supprimer : c'est exactement ce qui permet de rebatir une config en secondes
# au lieu de rebatir l'image de base. Cote Dockerfile, les memes COPY restent en
# place parce que l'image osmocom-nitb doit rester autonome (start-nitb.sh la
# lance seule, avec son propre ENTRYPOINT).
# [2026-09-13] BASE parametrable. L'image de base ne se rebatit plus a chaque
# push : la CI la publie une fois sur GHCR (osmocom-nitb:base-<empreinte du
# contexte>) et passe sa reference ici. En local, le defaut reste l'image du
# docker de l'hote, donc `docker build -f Dockerfile.run .` et build.sh ne
# changent pas d'un iota. Meme mecanique que Dockerfile.lite, qui a son ARG
# BASE depuis toujours.
ARG BASE=osmocom-nitb
FROM ${BASE} AS osmocom-run

# ── Scripts ──────────────────────────────────────────────────────────────
# Deux destinations, deux usages :
#   /scripts      -> entrypoint.sh (l'ENTRYPOINT ci-dessous) et les outils que
#                    scripts/run.sh appelle par ce chemin (pulse-gsm-setup.sh) ;
#   /etc/osmocom  -> ce que consomment les modules ($OSMOCOM_CFG/...). L'image de
#                    base les y depose deja, mais a l'etat de SON build : sans
#                    cette COPY, osmo-start.sh / status.sh / check.sh /
#                    audio-chain.sh n'etaient JAMAIS rafraichis par l'image
#                    legere — trou dans la « mise a jour des scripts ».
# Cette COPY groupee remplace les 7 COPY unitaires d'avant (sms-interop-relay.py,
# gapk-start.sh, smsc-start.sh, send-mt-sms.sh, pulse-gsm-setup.sh...), qui
# recopiaient un a un des fichiers deja couverts.
# ⚠ Le depot versionne ces fichiers en 644 (cf. start-nitb.sh l.50-55) : le bit
# +x se pose APRES les COPY, dans le RUN plus bas — une COPY posterieure au
# chmod reposerait le mode 644 et annulerait le chmod.
COPY scripts/* /etc/osmocom/
COPY scripts/entrypoint.sh /scripts/

# ── Configs Osmocom (templates avec placeholders __XXX__) ────────────────
# Toutes les configs sont copiées ; les valeurs réelles sont injectées
# au démarrage via start-docker.sh (apply_config_templates) et montées en volume.
COPY configs/*.conf /etc/asterisk/
COPY configs/*.cfg configs/sms-routing.conf /etc/osmocom/
COPY configs/mobile.cfg /root/.osmocom/bb/mobile.cfg
COPY configs/asound.conf /etc/

# ── Pont TRX burst-level (CALYPSO_BRIDGE=pont) ───────────────────────────
# AVANT : le pont n'etait NULLE PART dans cette image. Il n'existait dans le
# conteneur que parce qu'on l'y avait depose a la main ; une reconstruction de
# l'image le faisait disparaitre, et start-native.sh tombait sur sa branche
# « pont introuvable (...) » en degradant silencieusement le mode.
# Le chemin est desormais EXPLICITE et unique : /opt/GSM/pont/pont.py, expose
# par PONT_PY que start-native.sh consomme (`_PONT="${PONT_PY:-...}"`).
# ⚠ La source qui part dans l'image est pont/pont.py DU CONTEXTE DE BUILD
# (/opt/GSM/osmo-operator/pont/). Apres avoir modifie /opt/GSM/pont/pont.py a
# chaud, la recopier ici avant de reconstruire, sinon l'image reste en arriere.
#
# EXCEPTION ASSUMEE au « tout dans Dockerfile » : pont.py est une source iteree
# a chaud, de la meme famille que scripts/ et configs/ — le garder ici, c'est
# une reconstruction en secondes au lieu de l'image de base. build-iso.sh
# l.590-592 nomme d'ailleurs explicitement Dockerfile.run comme la raison pour
# laquelle pont/ doit figurer dans la charge de l'ISO.
# (Le `mkdir -p /opt/GSM/pont` qui precedait est supprime : une COPY cree
#  toujours ses repertoires parents.)
# [2026-08-27] COPY RETIRE. Il fabriquait /opt/GSM/pont/pont.py, une copie
# HORS DEPOT que rien ne gardait en phase avec pont/pont.py -- et l ENV
# PONT_PY la rendait souveraine. Le 27/08 cette copie etait restee a une
# version sans plan SDCCH/8 : deux heures de LU en echec, SMS rejete et
# appel impossible, sans un message. Le depot est desormais la seule source ;
```

```
# start-native.sh resout $NITB_ROOT/pont/pont.py.
# [2026-10-24] ENV PONT_PY RETIRE : il forcait pont.py (grgsm) jusque dans
# c54x_exe/run.sh en MODE=dsp, qui veut pont_dsp.py. Chaque lanceur a son defaut.

# Bits d'execution + chemins uniques. DOIT rester APRES toutes les COPY ci-dessus.
# /scripts/run.sh (laissé par "COPY scripts/*") et /etc/osmocom/run.sh
# pointent vers la même source que /root/run.sh — on symlink pour éviter
# trois copies divergentes. Idem /root/osmo-start.sh, chemin de memoire
# musculaire que rien ne reference dans le depot : un symlink plutot qu'une
# deuxieme copie qui divergerait.
RUN chmod +x /etc/osmocom/*.sh

# ── Disclaimer au login du conteneur (pendant du 01-osmo-disclaimer.sh de l'ISO)
# On veut le MEME rappel a l'entree d'un conteneur qu'au boot de l'ISO. Le
# conteneur se rejoint par `docker exec -ti <ct> bash` : un shell interactif
# NON-login, qui lit ~/.bashrc (et pas /etc/profile.d) - c'est la qu'est deja
# pose le prompt calypso. On depose donc le texte dans un fichier a part et on
# le source depuis .bashrc, garde par [ -t 1 ] et OSMO_DISCLAIMER_SHOWN :
#   - [ -t 1 ]                : uniquement quand un humain a un terminal ;
#   - OSMO_DISCLAIMER_SHOWN   : une seule fois par session, pas a chaque sous-shell.
# Les `bash -c "..."` de start-docker.sh (lancement des piles) sont NON-interactifs :
# bash n'y lit pas .bashrc, le disclaimer ne pollue donc pas le defilement des
# [ OK ]. Le fichier n'est pas interpole au build : $(hostname) n'est evalue
# qu'a l'ouverture du shell, il nomme donc le conteneur courant.
# Le texte lui-meme vit dans scripts/osmo-disclaimer.sh (deja depose en
# /etc/osmocom/ par la COPY plus haut) : PAS de heredoc ici. Le builder docker
# classique (`Step N/M`, sans BuildKit) ne connait pas les heredocs — il joignait
# les lignes de continuation et lancait le RUN sans le corps :
#   /bin/bash: warning: here-document at line 1 delimited by end-of-file
# laissant /etc/osmo-disclaimer.sh VIDE et le disclaimer muet. Un fichier du
# depot + un symlink marchent avec les deux builders.
# Le `grep -q` garde l'ajout idempotent si l'image de base a deja pose le bloc.
RUN ln -sf /etc/osmocom/osmo-disclaimer.sh /etc/osmo-disclaimer.sh && \
    { grep -q '### osmo-disclaimer ###' /root/.bashrc || \
      printf '%s\n' '' \
        '### osmo-disclaimer ###' \
        '[ -z "${OSMO_DISCLAIMER_SHOWN:-}" ] && [ -t 1 ] && . /etc/osmo-disclaimer.sh' \
        '### end osmo-disclaimer ###' >> /root/.bashrc; }

# ── Mise a jour des arbres git ──────────────────────────────────────────
# [refactor] start-docker.sh l.323 et l.326 lancent
#   `docker build --build-arg QEMU_CACHE_BUST=$(date +%s) -f Dockerfile.run`
# mais AUCUN `ARG QEMU_CACHE_BUST` n'existait : docker repondait « unused build
# argument » et servait les deux `git pull` ci-dessous DEPUIS LE CACHE — les
# deux RUN qu'on garde justement pour la fraicheur ne rafraichissaient rien, et
# l'image run pouvait embarquer un qosmo et un osmo-operator perimes sans que
# rien ne le signale. L'ARG est declare ici, JUSTE AVANT les pulls (un ARG
# invalide le cache a son PREMIER USAGE, pas a sa declaration : d'ou le
# `: "${QEMU_CACHE_BUST:-}"` en tete de chaque RUN, qui le consomme reellement).
ARG QEMU_CACHE_BUST

# [2026-08-12] Le `cd /opt/GSM/osmo-operator && git pull` qui était ici a été
# supprimé : il faisait doublon avec celui de la ligne ~192, et rien entre les
# deux ne consomme /opt/GSM/osmo-operator. On garde le plus tardif, au plus près
# de l'entrypoint.
# Le clone de l'image de base laisse HEAD sur la branche par defaut : un simple
# pull suffit, aucune ref a nommer ici.
# [2026-08-31] LE PULL NE PEUT PLUS TUER LE BUILD.
# Constate : `git pull` ici sort en 1 avec
#     fatal: could not read Username for 'https://github.com': No such device or address
#     fatal: expected flush after ref listing
# et emporte une image de 11 Go et 40 minutes de compilation - pour un
# rafraichissement, pas pour une dependance.
#
# CE N EST PAS UN PROBLEME D IDENTIFIANTS. Le depot est PUBLIC : depuis l hote
# comme depuis un conteneur LANCE (docker run), `git ls-remote origin HEAD`
```

```
# repond a0a7888 en anonyme, DNS compris. Seul le reseau du BUILD echoue. Git
# lit alors une reponse HTTP qu il n attend pas ("expected flush after ref
# listing"), la prend pour une demande d authentification, et va chercher un
# terminal qui n existe pas dans un build - d ou le message trompeur sur le nom
# d utilisateur, qui envoie chercher des identifiants la ou il n en faut aucun.
#
# GIT_TERMINAL_PROMPT=0 : echouer TOUT DE SUITE et clairement plutot que de
# bloquer sur une invite invisible.
# `|| echo` : l arbre du clone de base reste utilisable tel quel. Une image qui
# se construit avec un depot de la veille vaut mieux qu une image qui ne se
# construit pas - et le message dit lequel des deux on a.
RUN : "${QEMU_CACHE_BUST:-}" && cd /opt/GSM/osmo-operator \
    && (GIT_TERMINAL_PROMPT=0 git pull --ff-only \
        || echo "[warn] git pull osmo-operator impossible (reseau du build) - on garde l arbre du clone de base")

# ── Rebuild qosmo + c54x_exe + grgsm_exe (pull + rebuild) ──────────────────
# [2026-09-25] qosmo-grgsm/qosmo-dsp sont retires : un seul arbre QEMU
# (/opt/GSM/qosmo) et deux couches 1 hors QEMU, c54x_exe et grgsm_exe, qui
# compilent les sources de /opt/GSM/qosmo (hw/arm/calypso, contrib/hors-qemu).
# Les trois sont donc rafraichis DANS LE MEME RUN : un c54x_exe recompile sur
# un qosmo perime (ou l inverse) donnerait un DSP et un ARM qui ne parlent pas
# du meme protocole de pont, sans rien qui le signale. Ne pas decouper.
# Le pull rafraichit, il ne conditionne pas (cf. osmo-operator ci-dessus).
# [2026-10-01] fetch + reset --hard, PAS `pull --ff-only`. Les arbres de la base
# portent des fichiers suivis modifies (__pycache__/*.pyc dans qosmo, le binaire
# c54x_exe versionne) : le pull refusait de fusionner, le `|| echo` avalait
# l echec et l image se reconstruisait sur du code perime. Ces arbres sont des
# sources a jour, pas un espace de travail : on les aligne sur origin.
# gcc-13 pour ninja : build/ de qosmo vient d un snapshot configure avec gcc-13
# (meson y a memorise -ftrivial-auto-var-init, absent de gcc-11, le `cc` par
# defaut de l image). Avec un arbre inchange ninja ne recompilait rien et la
# difference ne se voyait pas ; des sources a jour imposent de recompiler, et
# gcc-11 echouait des la premiere unite. Un PATH local suffit (update-alternatives ne connait pas gcc-13) ; le
 ↪  defaut reste gcc-11.
# c54x_exe : CFLAGS sans -march=native, comme dans le Dockerfile de base.
RUN : "${QEMU_CACHE_BUST:-}" && \
    for d in qosmo c54x_exe grgsm_exe; do \
        (cd /opt/GSM/$d && GIT_TERMINAL_PROMPT=0 git fetch origin \
            && git reset --hard "origin/$(git rev-parse --abbrev-ref HEAD)") \
            || echo "[warn] mise a jour de $d impossible (reseau ?) - on rebatit sur l arbre present"; \
    done && \
    mkdir -p /tmp/cc13 && ln -sf /usr/bin/gcc-13 /tmp/cc13/cc && ln -sf /usr/bin/gcc-13 /tmp/cc13/gcc \
    && ln -sf /usr/bin/g++-13 /tmp/cc13/c++ && ln -sf /usr/bin/g++-13 /tmp/cc13/g++ && \
    (cd /opt/GSM/qosmo/build && PATH=/tmp/cc13:$PATH ninja) && rm -rf /tmp/cc13 && \
    make -C /opt/GSM/c54x_exe QOSMO=/opt/GSM/qosmo \
        CFLAGS="-O3 -g -Wall -Werror=format -Werror=format-extra-args -Wno-unused-function -Wno-unused-variable
 ↪  -Wno-unused-but-set-variable -Wno-sign-compare" && \
    cp /opt/GSM/c54x_exe/rom/calypso_dsp.*.bin /opt/GSM/ && \
    make -C /opt/GSM/grgsm_exe QOSMO=/opt/GSM/qosmo clean all

# ── Entrypoint ─────────────────────────────────────────────────────────────
ENTRYPOINT ["/scripts/entrypoint.sh"]
CMD ["bash"]
```

3.5 /opt/GSM/osmo-operator/Dockerfile.stp

9335 octets, 174 lignes → 174 lignes

```
# ============================================================================
# Dockerfile.stp — l'inter-STP : le hub SS7 du WAN, et rien d'autre
# ============================================================================
#
# osmocom-run porte toute la maquette (HLR, MSC, BSC, PCU, BTS, Asterisk, QEMU,
# gr-gsm…). Le hub n'en utilise rien : il reçoit du M3UA, consulte sa table de
```

```
# routage, renvoie vers le bon linkset. Cette image ne contient que ça.
#
# osmo-stp n'a pas de dépôt à lui — il est produit par libosmo-sigtran. D'où ces
# trois sources et pas quatre.
#
# Versions alignées sur Dockerfile : un hub compilé contre une libosmo-sigtran
# plus récente que les nœuds négocierait des extensions M3UA qu'ils ignorent, ce
# qui ne se lit que « ASP DOWN ».
#
# DEUX FAÇONS DE LA CONSTRUIRE, ET POURQUOI
#
#   autonome (défaut) — compile les trois bibliothèques depuis ubuntu:24.04
#   (noble, la meme base que Dockerfile), en passant par le cache .deb de
#   packaging/osmo-deb.sh : la seconde construction ne compile plus rien.
#     docker build -f Dockerfile.stp -t osmocom-stp .
#     docker compose build stp          (build.sh / build-iso.sh --role=interstp)
#
#   sur la pile existante — si osmocom-nitb est déjà là (c'est la base de
#   Dockerfile.run), elle porte DÉJÀ ces bibliothèques et osmo-stp. On les
#   reprend au lieu de les recompiler : quelques secondes, et surtout la
#   certitude que le hub et les nœuds partagent le MÊME code. Deux compilations
#   séparées peuvent diverger sans que rien ne le signale.
#     docker build -f Dockerfile.stp --build-arg BASE_IMAGE=osmocom-nitb -t osmocom-stp .
#
#   docker run --rm -p 2908:2908 -p 4239:4239 osmocom-stp
#   docker run --rm -v ./hub.cfg:/etc/osmocom/osmo-stp-interop.cfg osmocom-stp
# ============================================================================

# ── Base commune aux deux étages ──────────────────────────────────────────
# La liste est celle de Dockerfile, reprise telle quelle. Volontairement plus
# large que ce que ces trois bibliothèques paraissent exiger : le configure de
# libosmocore active ou refuse ses composants selon ce qu'il trouve, et une
# liste taillée au plus juste l'a déjà fait échouer (« no suitable Python
# interpreter found » — python3 sert à ses générateurs de code, pas à
# l'exécution). La déclarer une seule fois évite qu'elle diverge entre l'étage
# qui compile et celui qui exécute.
ARG BASE_IMAGE=ubuntu:24.04
FROM ${BASE_IMAGE} AS base
ARG DEBIAN_FRONTEND=noninteractive
# apt-fast : le meme installeur que Dockerfile (packaging/apt-fast-install.sh).
# Cache apt BuildKit, le meme que Dockerfile (id osmo-apt-archives).
ARG OSMO_UBUNTU_MIRROR=
COPY packaging/apt-fast-install.sh /usr/local/sbin/apt-fast-install
RUN --mount=type=cache,id=osmo-apt-archives,target=/var/cache/apt/archives,sharing=locked \
    chmod 755 /usr/local/sbin/apt-fast-install && apt-fast-install
# apt est idempotent : sur osmocom-nitb, où tout est déjà installé, cette étape
# ne fait que constater. On ne duplique donc pas la liste pour ce cas.
# libasound2t64 : nom noble de libasound2 (transition time_t 64 bits).
RUN --mount=type=cache,id=osmo-apt-archives,target=/var/cache/apt/archives,sharing=locked \
    apt-fast update && apt-fast install -y --no-install-recommends \
    build-essential git gcc g++ make cmake autoconf automake libtool pkg-config wget curl \
    libtalloc-dev libpcsclite-dev libsctp-dev libmnl-dev liburing-dev \
    libdbi-dev libdbd-sqlite3 libsqlite3-dev sqlite3 libc-ares-dev libgnutls28-dev \
    libortp-dev libfftw3-dev libusb-1.0-0-dev libsofia-sip-ua-dev libsofia-sip-ua-glib-dev \
    python3 python3-dev python3-scapy ca-certificates \
    libasound2-dev libasound2t64 libgsm1-dev libgsm1 libpcap0.8-dev libsystemd-dev \
    && rm -rf /var/lib/apt/lists/*
ENV PKG_CONFIG_PATH=/usr/local/lib/pkgconfig \
    LD_LIBRARY_PATH=/usr/local/lib

# ── Compilation ───────────────────────────────────────────────────────────
FROM base AS builder
ARG LIBOSMOCORE_VERSION=1.12.1
ARG LIBOSMO_NETIF_VERSION=1.7.0
ARG LIBOSMO_SIGTRAN_VERSION=2.2.1
```

```
WORKDIR /build

# ── Cache .deb (packaging/osmo-deb.sh) : les memes paquets que l image nitb ──
# .deb-cache/ est le miroir du cache de l hote (/var/cache/osmo-debs). Un
# paquet osmo-build-libosmocore_1.12.1~noble deja la = pas de compilation.
ARG OSMO_DEB_REFRESH=0
COPY packaging/osmo-deb.sh /usr/local/sbin/osmo-deb
COPY .deb-cache/ /var/cache/osmo-debs/
RUN chmod 755 /usr/local/sbin/osmo-deb

# Ordre imposé : chaque bibliothèque se compile contre la précédente.
# libosmocore depuis github (miroir officiel), les deux autres depuis gitea —
# même provenance que Dockerfile, pour que les deux images voient le même code.
#
# Si la base fournit déjà osmo-stp (cas BASE_IMAGE=osmocom-nitb), on ne
# recompile rien : ce serait long, et surtout ce serait un SECOND exemplaire des
# mêmes bibliothèques, susceptible de diverger de celui des nœuds.
RUN set -eux; \
    if command -v osmo-stp >/dev/null 2>&1; then \
        echo "osmo-stp fourni par l'image de base — compilation inutile"; \
        osmo-stp --version; \
        exit 0; \
    fi; \
    for repo in \
        "libosmocore:${LIBOSMOCORE_VERSION}:https://github.com/osmocom/libosmocore" \
        "libosmo-netif:${LIBOSMO_NETIF_VERSION}:https://gitea.osmocom.org/osmocom/libosmo-netif" \
        "libosmo-sigtran:${LIBOSMO_SIGTRAN_VERSION}:https://gitea.osmocom.org/osmocom/libosmo-sigtran" \
    ; do \
        name="${repo%%:*}"; rest="${repo#*:}"; \
        version="${rest%%:*}"; url="${rest#*:}"; \
        if osmo-deb install "$name" "$version"; then continue; fi; \
        git clone "$url" "$name"; \
        cd "/build/$name"; \
        git checkout "$version"; \
        autoreconf -fi; \
        ./configure --prefix=/usr/local --disable-static; \
        make -j"$(nproc)"; \
        osmo-deb pack "$name" "$version" make install; \
        ldconfig; \
        cd /build; \
    done

# Config par défaut du hub, produite par le script du dépôt : une seule source
# de vérité pour le plan de point codes 1.<nœud><op>.<rôle>, partagée avec l'ISO
# et avec start-docker.sh. Remplaçable par un montage au lancement.
ARG HUB_NODES=3
ARG HUB_OPS=1
COPY helpers/create_interop.sh /build/create_interop.sh
RUN mkdir -p /etc/osmocom \
    && bash /build/create_interop.sh --wan "${HUB_NODES}" "${HUB_OPS}" \
         /etc/osmocom/osmo-stp-interop.cfg

# ── Image finale ─────────────────────────────────────────────────────
# Même base, donc mêmes bibliothèques : rien de ce que la compilation a pu
# activer ne peut manquer. Ce qui reste dehors, ce sont les arbres de sources
# et les objets de compilation — l'essentiel du volume.
FROM base
LABEL org.opencontainers.image.title="osmocom-stp" \
      org.opencontainers.image.description="Inter-STP osmo-operator — hub SS7/M3UA, PC 0.0.0"

# groupadd et useradd échouent (code 9) si le compte existe déjà — et il existe
# déjà quand BASE_IMAGE=osmocom-nitb, où le Dockerfile principal le crée. Sans
# ces gardes, le chemin « construire sur la pile existante » ne construit pas du
# tout, alors que c est justement le plus rapide des deux.
RUN getent group osmocom >/dev/null || groupadd -r osmocom \
    && id -u osmocom >/dev/null 2>&1 \
```

```
       || useradd -r -g osmocom -s /usr/sbin/nologin -d /var/lib/osmocom osmocom \
    && mkdir -p /var/lib/osmocom /var/log/osmocom \
    && chown osmocom:osmocom /var/lib/osmocom /var/log/osmocom

# /usr/local est l'endroit où les deux chemins déposent leur résultat : celui
# qui compile (--prefix=/usr/local) comme osmocom-nitb (même préfixe dans
# Dockerfile). Une seule copie couvre donc les deux cas.
COPY --from=builder /usr/local/bin/osmo-stp /usr/local/bin/osmo-stp
COPY --from=builder /usr/local/lib/         /usr/local/lib/
COPY --from=builder /etc/osmocom/osmo-stp-interop.cfg /etc/osmocom/osmo-stp-interop.cfg
# Les paquets produits voyagent dans l image finale : build.sh les ramene dans
# le cache de l hote (docker cp), et build-iso.sh --role=interstp les pose
# dans le rootfs du hub par dpkg.
COPY --from=builder /var/cache/osmo-debs/ /var/cache/osmo-debs/

# La clôture des dépendances vérifiée sur l'image LIVRÉE, binaire et
# bibliothèques : une .so manquante ne se verrait sinon qu'au premier démarrage,
# dans une VM, loin d'ici.
RUN set -eux; \
    ldconfig; \
    fail=0; \
    for f in /usr/local/bin/osmo-stp /usr/local/lib/*.so*; do \
        [ -f "$f" ] || continue; \
        if ldd "$f" 2>/dev/null | grep 'not found'; then \
            echo "DÉPENDANCE MANQUANTE : $f" >&2; fail=1; \
        fi; \
    done; \
    [ "$fail" = "0" ]; \
    osmo-stp --version

# 2908 : M3UA — en SCTP, que EXPOSE ne sait pas déclarer (tcp/udp seulement).
# 4239 : VTY.
EXPOSE 2908/tcp 4239/tcp

ENTRYPOINT ["/usr/local/bin/osmo-stp"]
CMD ["-c", "/etc/osmocom/osmo-stp-interop.cfg"]
```

3.6 /opt/GSM/osmo-operator/addition.sh

76801 octets, 1275 lignes → 1274 lignes (1 groupes compactés)

```
#!/bin/bash
# ============================================================================
# addition.sh - LES SUPPLEMENTS QUI NE SONT PAS DANS L ISO
#
# Jumelle d update.sh, et lancee comme elle : une icone GTK sur le bureau
# (osmo-addition-anim), ou a la main.
#
# POURQUOI DOCKER N EST PAS DANS L IMAGE
# L ISO est un noeud NATIF : elle porte le banc complet sans conteneur, et
# n a donc besoin de docker pour rien. L y embarquer couterait ~500 Mo et le
# demon tournerait en permanence sur un banc qui ne s en sert pas - avec son
# bridge, ses regles iptables et sa MASQUERADE, au milieu d une machine dont
# tout l interet est de router du GSM a la main.
# Docker ne sert qu a UN scenario : le multi-operateur (start-multi.sh), ou les
# operateurs 2 et 3 et l inter-STP tournent en conteneurs a cote du natif.
# C est un supplement, il s installe comme tel - ici.
#
# Usage :
#   sudo ./addition.sh            fenetre GTK : "Docker container and SS7
#                                 multioperator", puis une seconde fenetre en
#                                 boutons radio : image TELECHARGEE ou COMPILEE
#                                 (exclusif). osmocom-run est derivee dans la
#                                 foulee, puis osmo-multi.service DEMARRE : en
```

```
#                               sortant, osmo-operator-2, -3 et le hub
#                               tournent (OSMO_MULTI_START=0 : poser sans
#                               lancer).
#   sudo ./addition.sh --multi    le meme, sans la fenetre : image TELECHARGEE
#                               (docker pull ghcr.io/<depot>/osmocom-nitb,
#                               taguee osmocom-nitb)
#   sudo ./addition.sh --multi-build
#                               le meme. [2026-09-13] Ce drapeau ne COMPILE
#                               plus : il tire lui aussi l image du depot,
#                               que la CI publie a partir du meme Dockerfile.
#                               Pour compiler vraiment (Dockerfile modifie,
#                               rien de publie) :
#                                 OSMO_ADDITION_BUILD_LOCAL=1 ... --multi-build
#   sudo ./addition.sh --docker   le moteur de conteneurs seul   (depannage)
#   sudo ./addition.sh --image    l image operateur seule, telechargee (depannage)
#   sudo ./addition.sh --build    l image operateur seule, du depot   (depannage)
#   sudo ./addition.sh --opencl   la pile OpenCL seule (calcul GPU)
#   sudo ./addition.sh --claude   Claude Code (CLI de l assistant) seul
#   sudo ./addition.sh --extras   Jeux + media (Doom, Quake III, Kodi,
#                               YouTube+uBlock, Wireshark root, Linphone)
#   sudo ./addition.sh --gnuarm   la chaine historique arm-elf-gcc 4.5.2 (/root/gnuarm),
#                               copiee de l image Docker osmocom-bb-trx
#   sudo ./addition.sh --magnetite Wine + TI TMS470, FreeCalypso Magnetite C139
#                               (sources, patchs du banc, image flash)
#   sudo ./addition.sh --trx      osmocom-bb-trx : image Docker publiee + firmware
#                               trx (TX) compile avec gnuarm
#   sudo ./addition.sh --calypso-fw le firmware Calypso du banc (layer1, rssi)
#                               recompile avec gnuarm (OSMO_FW_INSTALLER=1 : le poser)
#   sudo ./addition.sh --fernvale fernly, fernvale-nuttx, layer1 MTK (MT6260)
#   sudo ./addition.sh --simcom   documentation et SDK SIMCom (~3,4 Go)
#   sudo ./addition.sh --firmwares les six precedents
#   sudo ./addition.sh --status   dit seulement ce qui est present
# ============================================================================
set -uo pipefail

RED='\033[0;31m'; GREEN='\033[0;32m'; YELLOW='\033[1;33m'
CYAN='\033[0;36m'; BOLD='\033[1m'; NC='\033[0m'

DIR="$(cd "$(dirname "$(readlink -f "$0")")" && pwd)"
MULTI_CONF="${MULTI_CONF:-/etc/osmocom/osmo-multi.conf}"
# NOM DE L IMAGE : osmocom-run, PAS "osmo-operator".
# build.sh produit osmocom-nitb (Dockerfile, ~11 Go) puis Dockerfile.run en
# derive osmocom-run, et c est CELLE-LA que start-docker.sh lance - verifie sur le
# banc : `docker inspect osmo-operator-1 --format {{.Config.Image}}` rend
# osmocom-run. Chercher "osmo-operator" (le nom du DEPOT, pas de l image)
# rendait la sonde toujours negative : "image absente" en permanence, et une
# recompilation Osmocom de 40 minutes relancee pour rien a chaque passage.
MULTI_IMAGE="${MULTI_IMAGE:-osmocom-run}"
# IMAGE DE BASE ET SON ORIGINE. osmocom-nitb est ce que build.sh COMPILE
# (~40 min sur une bonne machine, 1h23 sur un runner). On la TIRE et on la tague
# osmocom-nitb : start-docker.sh en derive ensuite osmocom-run via Dockerfile.run
# exactement comme apres un build local.
#
# [2026-09-13] LA SOURCE EST GHCR, PLUS DOCKER HUB. .github/workflows/docker.yml
# publie l'image a chaque push sur ghcr.io/<depot>/osmocom-nitb - c'est la MEME
# pile, construite par la CI a partir du MEME Dockerfile, et elle suit le depot
# au lieu de dependre d'un compte Hub tenu a la main. bastienbaranoff/norf_gsm
# reste joignable en posant HUB_IMAGE.
#   :latest      la derniere publiee (defaut)
#   :base-<sha>  une empreinte de contexte precise, si on veut figer
BASE_IMAGE="${BASE_IMAGE:-osmocom-nitb}"
HUB_IMAGE="${HUB_IMAGE:-ghcr.io/bbaranoff/osmo-operator/osmocom-nitb:latest}"

# ── _trust_desktop : rendre un raccourci du bureau VISIBLE par DING ──────────
# DING n affiche un .desktop avec son nom et son icone que s il est executable,
# possede par l utilisateur du bureau, ET porteur de metadata::trusted. Cet
```

```
# attribut ne vit PAS dans le fichier : il est ecrit dans les metadonnees gvfs
# de la SESSION, donc gio doit parler au bus de CET utilisateur.
#
# LE PIEGE QUI RENDAIT L ICONE INVISIBLE. addition.sh est lance par
# `pkexec env DISPLAY=... XAUTHORITY=... ` (osmo-addition-anim) : pkexec NETTOIE
# l environnement et ne transmet ni XDG_RUNTIME_DIR ni DBUS_SESSION_BUS_ADDRESS.
# `gio set metadata::trusted` n avait alors aucun bus a joindre, echouait en
# silence (2>/dev/null), et le fichier restait pose mais jamais approuve - donc
# absent du bureau. On reconstruit l env a partir de l UID proprietaire du
# bureau et on lance gio SOUS cet utilisateur.
_trust_desktop() {
    local f="$1" home="$2" pos="${3:-}" owner uid bus
    [ -f "$f" ] || return 1
    owner="$(stat -c '%U' "$home" 2>/dev/null)"; [ -n "$owner" ] || owner=root
    uid="$(id -u "$owner" 2>/dev/null)" || return 1
    chown "$owner" "$f" 2>/dev/null || true
    chmod +x "$f" 2>/dev/null || true
    bus="/run/user/$uid/bus"
    # Session active pour ce proprietaire : on pose les attributs TOUT DE SUITE.
    # Sinon (pas de bus), osmo-trust-desktop (autostart) posera trusted au
    # prochain login - l icone apparait alors sans intervention.
    # pos "x,y" (optionnel) : place l icone sur le bureau via DING.
    if [ -S "$bus" ] && command -v runuser >/dev/null 2>&1; then
        runuser -u "$owner" -- env XDG_RUNTIME_DIR="/run/user/$uid" \
            DBUS_SESSION_BUS_ADDRESS="unix:path=$bus" \
            gio set -t string "$f" metadata::trusted true 2>/dev/null || true
        [ -n "$pos" ] && runuser -u "$owner" -- env XDG_RUNTIME_DIR="/run/user/$uid" \
            DBUS_SESSION_BUS_ADDRESS="unix:path=$bus" \
            gio set -t string "$f" metadata::nautilus-icon-position "$pos" 2>/dev/null || true
    fi
    return 0
}

# ── LES BUREAUX A SERVIR : /root ET CELUI DE LA SESSION GRAPHIQUE ──────────
# [2026-09-05] LE BUREAU DE L UTILISATEUR ETAIT MANQUE, ET TOUJOURS. Les deux
# blocs qui posent des icones construisaient leur liste ainsi :
#     _homes=(/root)
#     [ -n "$SUDO_USER" ] && _homes+=("$(getent passwd "$SUDO_USER" ...)")
# Or SUDO_USER n existe QUE sous sudo. Ces scripts sont lances par une icone,
# donc par PKEXEC (qui pose PKEXEC_UID, pas SUDO_USER), ou par une unite
# systemd, ou depuis un shell root - trois cas ou la variable est vide. La
# liste restait donc (/root), tout atterrissait dans /root/Desktop, et le
# compte cree par l installeur ouvrait sa session sur un bureau NU. C est la
# meme confusion racine/session qui rendait les boutons de l encart inertes.
#
# On cherche donc le compte de la SESSION GRAPHIQUE, dans cet ordre :
#   1. SUDO_USER   (sudo)
#   2. PKEXEC_UID  (l icone, pkexec)
#   3. le proprietaire d un bus de session actif (/run/user/*/bus) - systemd,
#      shell root, autologin : c est la seule preuve qu une session existe.
# [2026-10-24] 0. OSMO_COMPTE : le compte cree par l installeur, passe par
#    osmo-supplements-firstboot.sh. Au premier demarrage personne n est encore
#    connecte : sans lui, ni groupe docker ni icone pour ce compte.
# /root reste TOUJOURS servi : la cle vive ouvre sa session en root.
_comptes_session() {
    local u uid bus
    for u in "${OSMO_COMPTE:-}" "${SUDO_USER:-}" \
             "$([ -n "${PKEXEC_UID:-}" ] && getent passwd "$PKEXEC_UID" | cut -d: -f1)"; do
        [ -n "$u" ] && [ "$u" != root ] || continue
        getent passwd "$u" >/dev/null 2>&1 && { printf '%s\n' "$u"; return 0; }
    done
    for bus in /run/user/*/bus; do
        [ -S "$bus" ] || continue
        uid="${bus#/run/user/}"; uid="${uid%/bus}"
        [ "$uid" = 0 ] && continue
        u="$(getent passwd "$uid" 2>/dev/null | cut -d: -f1)"
```

```
        [ -n "$u" ] && printf '%s\n' "$u"
    done
}

_bureau_homes() {
    local homes=(/root) u uh
    while read -r u; do
        [ -n "$u" ] || continue
        uh="$(getent passwd "$u" 2>/dev/null | cut -d: -f6)"
        [ -n "$uh" ] && [ -d "$uh" ] && homes+=("$uh")
    done < <(_comptes_session)
    printf '%s\n' "${homes[@]}"
}

# ── LE COMPTE DE LA SESSION DOIT POUVOIR PARLER A DOCKER ──────────────────
# [2026-09-05] MESURE SUR LE BANC : le Conky affichait « banc a l arret » et ses
# sections Coeur GSM / Radio / Abonnes VIDES pendant que les trois operateurs
# tournaient. Cause : le Conky et l encart tournent sous le compte de la
# SESSION, qui n etait dans aucun groupe docker ; chaque sonde
# (`docker exec ... pgrep`) rendait « permission denied while trying to connect
# to the docker API », donc « arrete ». tools/osmo-op.sh decrit deja ce piege -
# une sonde qui confond « absent » et « invisible » est une sonde qui ment - et
# son commentaire affirmait que le compte etait « ajoute au groupe docker par
# ailleurs (users.conf, addition.sh) ». Ce code n existait nulle part.
# Le groupe ne prend qu au prochain login : on le dit plutot que de laisser
# croire a une panne.
_docker_groupe_session() {
    getent group docker >/dev/null 2>&1 || return 0
    local u
    while read -r u; do
        [ -n "$u" ] || continue
        id -nG "$u" 2>/dev/null | tr ' ' '\n' | grep -qx docker && continue
        usermod -aG docker "$u" 2>/dev/null || continue
        echo -e "  ${GREEN}✓${NC} ${BOLD}$u${NC} ajoute au groupe ${BOLD}docker${NC} (le Conky et l encart voient
        ↪  les conteneurs)"
        echo -e "    ${YELLOW}!${NC} effectif au prochain login de $u"
    done < <(_comptes_session)
}

DO_DOCKER=0; DO_IMAGE=0; DO_MULTI=0; DO_OPENCL=0; DO_CLAUDE=0; STATUS_ONLY=0; ANY_FLAG=0
DO_BUILD=0; DO_EXTRAS=0; DO_SIM=0
DO_GNUARM=0; DO_MAGNETITE=0; DO_TRX=0; DO_FERNVALE=0; DO_SIMCOM=0; DO_CALFW=0
for a in "$@"; do
    case "$a" in
        --docker) DO_DOCKER=1; ANY_FLAG=1 ;;
        --image)  DO_IMAGE=1;  ANY_FLAG=1 ;;
        --build)  DO_IMAGE=1;  DO_BUILD=1; ANY_FLAG=1 ;;
        --multi)  DO_MULTI=1;  ANY_FLAG=1 ;;
        --multi-build) DO_MULTI=1; DO_BUILD=1; ANY_FLAG=1 ;;
        --opencl) DO_OPENCL=1; ANY_FLAG=1 ;;
        --claude) DO_CLAUDE=1; ANY_FLAG=1 ;;
        --extras) DO_EXTRAS=1; ANY_FLAG=1 ;;
        --sim)    DO_SIM=1;    ANY_FLAG=1 ;;
        --gnuarm)    DO_GNUARM=1;    ANY_FLAG=1 ;;
        --magnetite) DO_MAGNETITE=1; ANY_FLAG=1 ;;
        --trx)       DO_TRX=1;       ANY_FLAG=1 ;;
        --calypso-fw) DO_CALFW=1;    ANY_FLAG=1 ;;
        --fernvale)  DO_FERNVALE=1;  ANY_FLAG=1 ;;
        --simcom)    DO_SIMCOM=1;    ANY_FLAG=1 ;;
        --firmwares) DO_GNUARM=1; DO_MAGNETITE=1; DO_TRX=1; DO_CALFW=1; DO_FERNVALE=1; DO_SIMCOM=1; ANY_FLAG=1 ;;
        --all)    DO_DOCKER=1; DO_IMAGE=1; DO_MULTI=1; DO_OPENCL=1; DO_CLAUDE=1; DO_EXTRAS=1; DO_SIM=1; ANY_FLAG=1
        ↪  ;;
        --status) STATUS_ONLY=1; ANY_FLAG=1 ;;
        -h|--help) sed -n '2,54p' "$0"; exit 0 ;;
    esac
done
```

```
echo -e "${CYAN}${BOLD}"
echo "╔═══════════════════════════════════════════════════════╗"
echo "║   osmo-operator - supplements (hors ISO)              ║"
echo "╚═══════════════════════════════════════════════════════╝"
echo -e "${NC}"

etat() {
    local ok_d=0 ok_r=0 ok_i=0 ok_m=0 ok_u=0 ok_k=0
    command -v docker >/dev/null 2>&1 && ok_d=1
    [ "$ok_d" = "1" ] && docker info >/dev/null 2>&1 && ok_r=1
    [ "$ok_r" = "1" ] && docker image inspect "$MULTI_IMAGE" >/dev/null 2>&1 && ok_i=1
    [ -f "$MULTI_CONF" ] && ok_m=1
    # [2026-10-01] CE QUE L ICONE EXIGE, L ETAT LE MONTRE. launch.sh --multi
    # refuse de demarrer sans l unite osmo-multi.service, sans start-multi.sh
    # et sans la topologie - et le dit par notification. Avant, cet etat ne
    # regardait que docker, l image et la topologie : « tout est vert » ici et
    # « unite absente » au clic, sur la meme machine (depot deroule a la main,
    # unite jamais posee). Les memes conditions, lues au meme endroit.
    systemctl cat osmo-multi.service >/dev/null 2>&1 && ok_u=1
    [ -f /usr/share/applications/osmo-multi.desktop ] && ok_k=1
    local ok_cl=0
    # clinfo qui rend au moins une plateforme : c est la seule preuve qu une
    # pile OpenCL est UTILISABLE. Les paquets poses ne prouvent rien - un ICD
    # sans pilote derriere laisse "Number of platforms 0".
    command -v clinfo >/dev/null 2>&1 \
        && [ "$(clinfo -l 2>/dev/null | grep -c Platform)" -gt 0 ] && ok_cl=1
    local ok_cc=0
    # claude qui repond --version : la seule preuve qu il est UTILISABLE. Le
    # binaire pose sans node derriere (installation npm cassee) sort en erreur.
    command -v claude >/dev/null 2>&1 \
        && claude --version >/dev/null 2>&1 && ok_cc=1
    [ "$ok_d" = "1" ] && echo -e "  docker installe      : ${GREEN}oui${NC}"      || echo -e "  docker installe
     ↪  : ${YELLOW}non${NC}"
    [ "$ok_r" = "1" ] && echo -e "  demon actif          : ${GREEN}oui${NC}"      || echo -e "  demon actif
     ↪  : ${YELLOW}non${NC}"
    [ "$ok_i" = "1" ] && echo -e "  image operateur      : ${GREEN}presente${NC}" || echo -e "  image operateur
     ↪  : ${YELLOW}absente${NC}"
    [ "$ok_m" = "1" ] && echo -e "  topologie SS7        : ${GREEN}posee${NC}"    || echo -e "  topologie SS7
     ↪  : ${YELLOW}absente${NC}"
    [ "$ok_u" = "1" ] && echo -e "  osmo-multi.service   : ${GREEN}pose${NC} ($(systemctl is-active
     ↪  osmo-multi.service 2>/dev/null || echo inactive))" || echo -e "  osmo-multi.service   :
     ↪  ${YELLOW}absent${NC}"
    [ "$ok_k" = "1" ] && echo -e "  icone multi-operator : ${GREEN}posee${NC}"    || echo -e "  icone
     ↪  multi-operator : ${YELLOW}absente${NC}"
    [ -x "$DIR/launch/start-multi.sh" ] || echo -e "  start-multi.sh       : ${YELLOW}absent de $DIR${NC}"
    [ "$ok_cl" = "1" ] && echo -e "  OpenCL               : ${GREEN}operationnel${NC}" || echo -e "  OpenCL
     ↪  : ${YELLOW}absent${NC}"
    [ "$ok_cc" = "1" ] && echo -e "  Claude Code          : ${GREEN}operationnel${NC}" || echo -e "  Claude Code
     ↪  : ${YELLOW}absent${NC}"
    # [2026-10-23] les firmwares telephone (tools/osmo-firmwares-install.sh)
    _fw_etat() { [ -e "$2" ] && echo -e "  $(printf '%-20s' "$1") : ${GREEN}present${NC}" || echo -e "  $(printf
     ↪  '%-20s' "$1") : ${YELLOW}absent${NC}"; }
    _fw_etat "gnuarm"            /root/gnuarm/install/bin/arm-elf-gcc
    _fw_etat "Magnetite C139"    /opt/GSM/firmware/magnetite/c139-flash.img
    _fw_etat "firmware trx (TX)" /opt/GSM/osmocom-bb-trx/src/target/firmware/board/compal_e88/trx.highram.bin
    _fw_etat "Fernvale fernly"   /opt/GSM/fernly/build/firmware.bin
    _fw_etat "Fernvale NuttX"    /opt/GSM/fernvale-nuttx/nuttx/nuttx.bin
    _fw_etat "layer1 MTK"
     ↪  /opt/GSM/osmocom-bb-mtk-layer1/src/target/firmware/board/fernvale/layer1.mt6260.bin
    _fw_etat "SIMCom"            /opt/GSM/simcom/.git
    { command -v docker >/dev/null 2>&1 && docker image inspect
     ↪  "${OSMOCOM_BB_TRX_IMAGE:-ghcr.io/bbaranoff/osmocom-bb-trx:main}" >/dev/null 2>&1; } \
        && echo -e "  image osmocom-bb-trx : ${GREEN}presente${NC}" || echo -e "  image osmocom-bb-trx :
         ↪  ${YELLOW}absente${NC}"
}
```

```
[ "$STATUS_ONLY" = "1" ] && { etat; exit 0; }

# ── UNITES osmo-* : User=osmocom -> compte de la SESSION ──────────────────
# Les .service amont (osmo-bsc/bts-trx/bts-virtual/msc) posent User=osmocom /
# Group=osmocom. En natif post-live le compte osmocom n'existe pas (l'image le
# supprime, cf iso_modules/52-qemu.sh) : les demons meurent en 217/USER avec
# Restart=always -> crash-loop bloquant et MUET (rien dans leur propre journal),
# et run.sh abandonne la sequence ("OsmoMSC started but never ready"). On rend
# ces unites au compte de la SESSION (celui de l'installeur, meme racine/session
# que le reste d'addition.sh) via un drop-in /etc/systemd/system/<unit>.d/ - on
# NE touche PAS au .service amont qu'un update de paquet reecrirait. A defaut de
# session (build en chroot) : root, comme 52-qemu.sh. Miroir : update.sh
# (rattrapage machines deja installees) et 52-qemu.sh (image live -> root).
_osmo_unites_user_session() {
    [ "$(id -u)" -eq 0 ] || return 0
    command -v systemctl >/dev/null 2>&1 || return 0
    local u grp unit base d n=0
    u="$(_comptes_session | head -1)"
    [ -n "$u" ] && getent passwd "$u" >/dev/null 2>&1 || u=root
    grp="$(id -gn "$u" 2>/dev/null || echo "$u")"
    for unit in /lib/systemd/system/osmo-*.service /etc/systemd/system/osmo-*.service; do
        [ -f "$unit" ] || continue
        grep -qE '^(User|Group)=osmocom$' "$unit" || continue
        base="$(basename "$unit")"; d="/etc/systemd/system/$base.d"
        mkdir -p "$d"
        printf '[Service]\nUser=%s\nGroup=%s\n' "$u" "$grp" > "$d/00-session-user.conf"
        n=$((n+1))
    done
    [ "$n" -gt 0 ] || return 0
    systemctl daemon-reload 2>/dev/null || true
    # les repertoires d'etat/log osmocom appartenaient a root/osmocom : sans ce
    # chown le demon (desormais User=$u) ne peut pas creer /var/log/osmocom/*.log
    # -> "Unable to create file" -> parse du .cfg en echec -> status=1.
    for _d in /var/log/osmocom /var/lib/osmocom /run/osmocom; do
        [ -e "$_d" ] && chown -R "$u:$grp" "$_d" 2>/dev/null || true
    done
    # couper un crash-loop en cours : relancer les unites qui tournaient/bouclaient
    for base in osmo-msc osmo-bsc osmo-bts-trx osmo-bts-virtual; do
        systemctl cat "$base" >/dev/null 2>&1 || continue
        systemctl reset-failed "$base" 2>/dev/null || true
        systemctl try-restart "$base" 2>/dev/null || true
    done
    echo -e "  ${GREEN}\u2713${NC} unites osmo-* rendues au compte de session ${BOLD}$u${NC} (sinon User=osmocom
     ↪  introuvable -> 217/USER)"
    return 0
}
_osmo_unites_user_session

# ── LE CHOIX ──────────────────────────────────────────────────────────────
# Lancee par l icone, cette fenetre est la SEULE interface : sans elle, un
# double-clic partait droit sur un apt-get de plusieurs centaines de Mo et une
# compilation Osmocom, sans rien demander.
#
# [2026-08-31] UN SEUL CHOIX, ET C EST VOULU.
# La liste a cocher en offrait trois - "multi-operateur SS7", "docker", "image
# operateur" - alors qu il n y a jamais eu qu un scenario : le multi-operateur,
# qui ENTRAINE les deux autres (l implication, juste en dessous, remettait
# DO_DOCKER=1 et DO_IMAGE=1 des que la premiere ligne etait cochee). Les deux
# dernieres cases n etaient donc decochables qu en apparence : les decocher ne
# changeait rien tant que la premiere restait cochee, et les cocher seules
# donnait un demi-supplement dont rien ne se sert - docker sans image, ou une
# image de 11 Go sans la topologie qui l emploie.
# Une fenetre qui propose des choix sans effet ment sur ce que fait le script.
#
# [2026-09-02] DEUX CHOIX MAINTENANT - ET LA REGLE N A PAS CHANGE.
```

```
# Ce qui etait reproche aux trois cases d avant, ce n est pas leur nombre :
# c est qu elles etaient FAUSSES (cocher "docker" seul ne donnait rien
# d utilisable, et le decocher ne l empechait pas d etre installe). OpenCL,
# lui, est un supplement REELLEMENT independant : il n entraine rien, rien ne
# l entraine, on peut le vouloir sans le multi-operateur et l inverse. Une case
# qui commande quelque chose a le droit d exister ; une case decorative, non.
# Les drapeaux --docker / --image / --opencl restent pour le depannage console.
if [ "$ANY_FLAG" = "0" ]; then
    if command -v zenity >/dev/null 2>&1 && [ -n "${DISPLAY:-}" ]; then
        # --print-column=2 N EST PAS FACULTATIF : le defaut de zenity est la
        # colonne 1, qui pour un --checklist est la CASE A COCHER - on
        # recupererait une liste de "TRUE" au lieu des choix. On imprime donc
        # la colonne 2, cachee, qui porte la CLE ("multi", "opencl") : le
        # libelle affiche peut alors changer sans casser les tests.
        # --separator : le defaut "|" traverse mal les tests ci-dessous ; on
        # prend un caractere qui n apparait dans aucun libelle.
        _choix=$(zenity --list --checklist --width=680 --height=340 \
            --title="osmo-operator - supplements" \
            --text="Supplements qui ne sont PAS dans l ISO.\nUne connexion Internet est necessaire." \
            --ok-label="Installer" --cancel-label="Fermer" \
            --separator="~" \
            --column="" --column="cle" --column="Supplement" \
            --hide-column=2 --print-column=2 \
            TRUE  multi  "Docker container and SS7 multioperator - docker.io, l image operateur (telechargee du
↪  hub OU compilee sur place : le choix vient ensuite), et la topologie op1 natif + op2/op3 docker + inter-STP" \
            FALSE opencl "OpenCL (calcul GPU) - runtime ICD, clinfo, le pilote de la carte detectee (Intel /
↪  Mesa-AMD, pocl en repli), et les outils deka / a51_tools / dst80_reversing / tea1-cracker clones dans /root" \
            FALSE claude "Claude Code (CLI de l assistant IA) - installeur natif claude.ai/install.sh (binaire
↪  autonome, sans npm) ; lance ensuite avec la commande claude" \
            FALSE firmwares "Firmwares telephone - gnuarm (arm-elf-gcc 4.5.2), FreeCalypso Magnetite C139 (Wine +
↪  TI TMS470), osmocom-bb-trx (image Docker + firmware trx TX), Fernvale MT6260 (fernly, NuttX, layer1 MTK),
↪  documentation SIMCom ; long (plusieurs Go, compilations)" \
            FALSE extras "Jeux + media - Doom (freedoom + chocolate-doom), Quake III (OpenArena), Kodi, YouTube
↪  (Firefox + uBlock Origin), Wireshark (root), Linphone ; ranges dans les dossiers Jeux / Media / Telephone /
↪  Outils et lancables depuis l encart du bureau" \
            2>/dev/null) || { echo "Annule."; exit 0; }
        [ -n "$_choix" ] || { echo "Rien de selectionne."; exit 0; }
        case "$_choix" in *multi*)  DO_MULTI=1  ;; esac
        # L ORIGINE DE L IMAGE : une seconde fenetre, en BOUTONS RADIO.
        # Deux sources exclusives par construction - un radiolist ne rend qu une
        # valeur, il n y a pas de "les deux" a rattraper apres coup comme avec
        # des cases a cocher.
        if [ "$DO_MULTI" = "1" ]; then
            _orig=$(zenity --list --radiolist --width=680 --height=260 \
                --title="osmo-operator - image operateur" \
                --text="D ou vient l image operateur (osmocom-nitb, ~11 Go) ?" \
                --ok-label="Continuer" --cancel-label="Fermer" \
                --column="" --column="cle" --column="Origine" \
                --hide-column=2 --print-column=2 \
                TRUE  dl    "LE DEPOT (GHCR) - ${HUB_IMAGE}, publiee par la CI a chaque push" \
                FALSE hub   "DOCKER HUB - bastienbaranoff/norf_gsm, la pile publiee a la main" \
                2>/dev/null) || { echo "Annule."; exit 0; }
            # [2026-09-13] LES DEUX CHOIX SONT DEUX SOURCES, PLUS « TIRER OU
            # COMPILER ». Compiler sur place refaisait, en dizaines de minutes,
            # exactement ce que la CI publie a partir du meme Dockerfile : ce
            # n'etait pas une origine, c'etait une attente. Le build local
            # existe toujours, mais il se demande explicitement
            # (OSMO_ADDITION_BUILD_LOCAL=1), pas par un bouton radio.
            case "$_orig" in
                hub)   DO_BUILD=0; HUB_IMAGE="bastienbaranoff/norf_gsm"; HUB_IS_RUN=1 ;;
                dl)    DO_BUILD=0 ;;
                *)     echo "Aucune origine choisie."; exit 0 ;;
            esac
        fi
        case "$_choix" in *opencl*) DO_OPENCL=1 ;; esac
        case "$_choix" in *claude*) DO_CLAUDE=1 ;; esac
```

```
        case "$_choix" in *extras*) DO_EXTRAS=1 ;; esac
        case "$_choix" in *firmwares*) DO_GNUARM=1; DO_MAGNETITE=1; DO_TRX=1; DO_CALFW=1; DO_FERNVALE=1;
        ↪  DO_SIMCOM=1 ;; esac
    else
        # Console sans zenity : le supplement historique, celui de l icone.
        DO_MULTI=1
    fi
fi

# Le multi-operateur ENTRAINE ses dependances : sans moteur ni image, la
# topologie posee ne servirait a rien et start-multi.sh renverrait ici.
if [ "$DO_MULTI" = "1" ]; then DO_DOCKER=1; DO_IMAGE=1; fi
# osmocom-bb-trx : la partie hote est une image Docker, il lui faut le moteur.
# [2026-10-24] gnuarm aussi : la chaine arm-elf-gcc n est plus recompilee, elle
# est copiee de cette meme image (tools/osmo-firmwares-install.sh). Le firmware
# Calypso du banc (--calypso-fw) la demande.
if [ "$DO_TRX$DO_GNUARM$DO_CALFW" != "000" ]; then DO_DOCKER=1; fi

[ "$(id -u)" -eq 0 ] || { echo -e "${RED}Root requis : sudo $0${NC}"; exit 1; }

# ── OPENCL ──────────────────────────────────────────────────────────────
# Supplement independant : rien ici n entraine docker, et docker n entraine pas
# ceci. Il sert au calcul GPU - les traitements lourds du cote SDR (correlation,
# FFT larges) et tout ce qu un utilisateur voudra faire tourner a cote.
#
# UNE PILE OPENCL, C EST DEUX MOITIES, ET L ERREUR CLASSIQUE EST DE N EN POSER
# QU UNE. Le "runtime" (ocl-icd) n est qu un aiguilleur : il lit
# /etc/OpenCL/vendors/*.icd et charge la bibliotheque du VENDEUR qui y est
# nommee. Sans ICD de vendeur, tout s installe sans une erreur et clinfo rend
#     Number of platforms   0
# ce qui ressemble a un materiel non supporte alors qu il ne manque qu un
# paquet. On pose donc l aiguilleur ET le pilote de la carte reellement
# presente, lue dans lspci.
#
# pocl est pose dans tous les cas : c est une implementation PROCESSEUR. Elle
# ne remplace pas un GPU, mais elle garantit qu un programme OpenCL trouve
# toujours une plateforme - y compris en machine virtuelle, ou dans un banc
# sans carte graphique dediee, ou l absence totale de plateforme fait echouer
# le code avec un CL_PLATFORM_NOT_FOUND_KHR que personne ne sait lire.
#
# NVIDIA est le cas a part : son ICD OpenCL n est pas un paquet separe
# installable seul, il voyage avec le pilote proprietaire (le paquet
# libnvidia-compute-xxx, tire par nvidia-driver-xxx). On installe donc le
# pilote, et l ICD vient avec. C est un pilote graphique proprietaire, sur une
# machine dont l ecran depend : il faut le plus souvent REDEMARRER pour qu il
# prenne - on le dit clairement.
#
# [2026-09-04] LA VERSION EST FIGEE : nvidia-driver-610, et plus
# `ubuntu-drivers install`. L outil interrogeait les depots et decidait seul :
# d une machine a l autre on n installait pas le meme pilote, et sans
# ubuntu-drivers-common il ne decidait rien. Un paquet nomme s installe, se
# verifie et se raconte : sudo apt install nvidia-driver-610.
# OSMO_NVIDIA_DRIVER permet d en choisir un autre.
if [ "$DO_OPENCL" = "1" ]; then
    echo -e "  ${CYAN}→${NC} installation de la pile OpenCL ..."
    export DEBIAN_FRONTEND=noninteractive

    # lvm2 : PAS de l OpenCL, mais deka (installe avec ce supplement) monte ses
    # tables sur des volumes LVM - sans vgchange, deka-start.sh ne monte rien.
    # On le pose donc ici, avec deka.
    # swig4.1 + python3-dev : compilation des modules natifs de deka
    # (_delta.so, _libvankus.so) plus bas. build-essential est deja tire
    # ailleurs.
    # [2026-09-04] swig4.1 ET PAS swig : le metapaquet `swig` de noble tire
    # swig 4.2, sous lequel les interfaces de deka ne passent plus. La 4.1 est
    # dans la distribution (noble/universe, paquet swig4.1) : on la nomme
```

```
# explicitement plutot que de dependre de ce que `swig` designera demain.
_cl_pkgs="ocl-icd-libopencl1 ocl-icd-opencl-dev opencl-headers clinfo pocl-opencl-icd lvm2 swig4.1
 ↪  python3-dev"
_vga="$(lspci 2>/dev/null | grep -iE 'vga|3d controller|display' || true)"
_nvidia=0
case "$_vga" in
    *[Ii]ntel*)   _cl_pkgs="$_cl_pkgs intel-opencl-icd" ;;
esac
case "$_vga" in
    *AMD*|*ATI*|*[Rr]adeon*) _cl_pkgs="$_cl_pkgs mesa-opencl-icd" ;;
esac
case "$_vga" in
    *NVIDIA*|*nVidia*) _nvidia=1 ;;
esac
echo -e "      paquets : ${BOLD}${_cl_pkgs}${NC}"

apt-get update || { echo -e "  ${RED}✗ apt-get update a echoue - pas de reseau ?${NC}"
                    echo -e "    Ce supplement a BESOIN d Internet : rien n est pre-telecharge dans l ISO."
                    exit 1; }
# Paquet par paquet, et non fatal : un ICD de vendeur absent du miroir
# (cela arrive) ne doit pas emporter le runtime avec lui - la pile
# processeur reste utilisable, et c est mieux que rien du tout.
for _p in $_cl_pkgs; do
    apt-get install -y "$_p" >/dev/null 2>&1 \
        && echo -e "      ${GREEN}✓${NC} $_p" \
        || echo -e "      ${YELLOW}!${NC} $_p non installe (absent du miroir ?)"
done

# pyopencl : le binding Python d OpenCL, POSE DANS LE VENV /root/.env - le
# meme que start-clean.sh (qosmo-grgsm) et le profil de root activent, et ou
# vit deja gr-gsm. On l installe avec le pip DU VENV (/root/.env/bin/python3)
# et pas le pip systeme : sinon le module ne serait pas visible du code qui
# tourne sous ce venv. opencl-headers + ocl-icd-opencl-dev (poses au-dessus)
# fournissent les en-tetes et libOpenCL.so dont pyopencl a besoin. Non fatal.
if [ -x /root/.env/bin/python3 ]; then
    echo -e "  ${CYAN}→${NC} pyopencl dans le venv ${BOLD}/root/.env${NC} ..."
    if /root/.env/bin/python3 -m pip install --no-cache-dir --disable-pip-version-check pyopencl >/dev/null
     ↪  2>&1; then
        echo -e "      ${GREEN}✓${NC} pyopencl"
    else
        echo -e "      ${YELLOW}!${NC} pyopencl non installe (voir : /root/.env/bin/python3 -m pip install
         ↪  pyopencl)"
    fi
else
    echo -e "  ${YELLOW}!${NC} venv /root/.env absent - pyopencl non pose (attendu sur l ISO)"
fi

# LA SEULE VERIFICATION QUI VAUT : est-ce qu une plateforme repond ?
if command -v clinfo >/dev/null 2>&1; then
    _np="$(clinfo -l 2>/dev/null | grep -c Platform || true)"
    if [ "${_np:-0}" -gt 0 ]; then
        echo -e "  ${GREEN}✓${NC} OpenCL operationnel - ${BOLD}${_np}${NC} plateforme(s) :"
        clinfo -l 2>/dev/null | sed 's/^/      /'
    else
        echo -e "  ${YELLOW}!${NC} paquets poses, mais aucune plateforme OpenCL ne repond."
        echo -e "      Voir : ${BOLD}ls /etc/OpenCL/vendors/${NC} puis ${BOLD}clinfo${NC}"
    fi
fi

if [ "$_nvidia" = "1" ]; then
   echo -e "  ${CYAN}→${NC} carte ${BOLD}NVIDIA${NC} detectee : installation du pilote (ICD OpenCL inclus) ..."
    # LES HEADERS DU NOYAU D ABORD. Le pilote NVIDIA compile son module par
    # DKMS contre le noyau EN COURS : sans linux-headers-$(uname -r), dkms
    # et build-essential, le build du module echoue et l installation part
    # en erreur (souvent muette : le paquet s installe, mais nvidia.ko
    # n existe pas, et OpenCL reste absent). On les pose avant.
```

```
_kver="$(uname -r)"
echo -e "      ${CYAN}→${NC} pre-requis DKMS : linux-headers-${_kver}, dkms, build-essential"
apt-get install -y "linux-headers-${_kver}" dkms build-essential >/dev/null 2>&1 \
    || echo -e "      ${YELLOW}!${NC} headers/dkms : installation partielle (headers absents du miroir ?)"
# linux-headers-generic en filet : garde les headers alignes sur le
# meta-noyau pour les mises a jour de noyau suivantes.
apt-get install -y linux-headers-generic >/dev/null 2>&1 || true
_nvdrv="${OSMO_NVIDIA_DRIVER:-nvidia-driver-610}"
echo -e "      ${CYAN}→${NC} ${BOLD}apt install ${_nvdrv}${NC}"
if apt-get install -y "$_nvdrv"; then
    echo -e "  ${GREEN}✓${NC} pilote NVIDIA installe : ${BOLD}$(dpkg-query -W -f='${Version}' "$_nvdrv"
     ↪  2>/dev/null)${NC}"
else
    echo -e "  ${YELLOW}!${NC} ${BOLD}${_nvdrv}${NC} non installe - il est dans multiverse :"
    echo -e "      ${BOLD}sudo add-apt-repository multiverse && sudo apt install ${_nvdrv}${NC}"
fi
# ── SECURE BOOT : SIGNER LES MODULES NVIDIA (MOK) ──────────────────
# Sous Secure Boot, le noyau REFUSE un module non signe par une cle de
# confiance : nvidia.ko compile par DKMS n est signe par personne, et le
# pilote reste muet (nvidia-smi : "NVIDIA-SMI has failed..."). On genere
# une cle MOK (une fois), on SIGNE les modules nvidia*.ko du noyau courant,
# puis on PROPOSE la cle a l enrolement (mokutil) - a VALIDER au prochain
# reboot dans l ecran bleu MOK Manager. Hors Secure Boot ou sans module :
# on saute. Emplacement standard Ubuntu : /var/lib/shim-signed/mok.
if command -v mokutil >/dev/null 2>&1 && mokutil --sb-state 2>/dev/null | grep -qi enabled; then
    echo -e "  ${CYAN}→${NC} Secure Boot actif : signature MOK des modules NVIDIA ..."
    apt-get install -y --no-install-recommends openssl mokutil zstd xz-utils >/dev/null 2>&1 || true
    _mokdir=/var/lib/shim-signed/mok
    _mokkey="$_mokdir/MOK.priv"; _mokder="$_mokdir/MOK.der"
    _mokpw="${OSMO_MOK_PASSWORD:-osmo-operator}"
    mkdir -p "$_mokdir"
    if [ ! -s "$_mokkey" ] || [ ! -s "$_mokder" ]; then
        if openssl req -new -x509 -newkey rsa:2048 -nodes -days 36500 \
                -keyout "$_mokkey" -outform DER -out "$_mokder" \
                -subj "/CN=osmo-operator NVIDIA MOK/" >/dev/null 2>&1; then
            chmod 600 "$_mokkey"
            echo -e "      ${GREEN}✓${NC} cle MOK generee : $_mokder"
        else
            echo -e "      ${YELLOW}!${NC} generation cle MOK echouee (openssl ?)"
        fi
    else
        echo -e "      ${CYAN}→${NC} cle MOK existante reutilisee : $_mokder"
    fi
    _kver="$(uname -r)"
    _signfile="/usr/src/linux-headers-${_kver}/scripts/sign-file"
    [ -x "$_signfile" ] || _signfile="/lib/modules/${_kver}/build/scripts/sign-file"
    if [ -x "$_signfile" ] && [ -s "$_mokkey" ] && [ -s "$_mokder" ]; then
        _signed=0
        while IFS= read -r _ko; do
            case "$_ko" in
                *.ko)     "$_signfile" sha256 "$_mokkey" "$_mokder" "$_ko" 2>/dev/null &&
                 ↪  _signed=$((_signed+1)) ;;
                *.ko.xz)  xz -d "$_ko"        && "$_signfile" sha256 "$_mokkey" "$_mokder" "${_ko%.xz}"
                 ↪  2>/dev/null && xz -q "${_ko%.xz}"          && _signed=$((_signed+1)) ;;
               *.ko.zst) zstd -q -d --rm "$_ko" && "$_signfile" sha256 "$_mokkey" "$_mokder" "${_ko%.zst}"
                 ↪  2>/dev/null && zstd -q --rm "${_ko%.zst}" && _signed=$((_signed+1)) ;;
                *.ko.gz)  gunzip "$_ko"      && "$_signfile" sha256 "$_mokkey" "$_mokder" "${_ko%.gz}"
                 ↪  2>/dev/null && gzip "${_ko%.gz}"           && _signed=$((_signed+1)) ;;
            esac
        done < <(find "/lib/modules/${_kver}" -name 'nvidia*.ko*' 2>/dev/null)
        depmod -a "${_kver}" 2>/dev/null || true
        echo -e "      ${GREEN}✓${NC} ${_signed} module(s) NVIDIA signe(s) (noyau ${_kver})"
    else
        echo -e "      ${YELLOW}!${NC} sign-file absent (linux-headers-${_kver} ?) - modules NON signes"
    fi
    if mokutil --test-key "$_mokder" >/dev/null 2>&1; then
```

```
            echo -e "      ${GREEN}✓${NC} cle MOK deja enrolee dans le firmware"
        elif printf '%s\n%s\n' "$_mokpw" "$_mokpw" | mokutil --import "$_mokder" >/dev/null 2>&1; then
            echo -e "  ${YELLOW}!${NC} MOK proposee a l enrolement. AU PROCHAIN REDEMARRAGE, l ecran bleu"
            echo -e "      ${BOLD}MOK Manager${NC} : Enroll MOK -> Continue -> mot de passe =
             ↪  ${BOLD}${_mokpw}${NC}"
        else
            echo -e "      ${YELLOW}!${NC} mokutil --import echoue (deja en attente ? droits root ?)"
        fi
    fi
    echo -e "  ${YELLOW}!${NC} un ${BOLD}redemarrage${NC} est generalement necessaire pour que le pilote"
    echo -e "      NVIDIA (et donc son OpenCL) soit actif - ${BOLD}clinfo${NC} le confirmera ensuite."
fi

# ── LES OUTILS QUI TOURNENT SUR OPENCL ──────────────────────────────
# deka, a51_tools, dst80_reversing, tea1-cracker : quatre depots de calcul
# qui se servent du GPU. Ils VONT AVEC OpenCL - c est pour eux qu on le
# pose - donc ils s installent ici, avec lui, et pas dans l ISO : l image
# reste un noeud GSM, ces outils sont un supplement de calcul.
# Clones dans /root. Idempotent : si le depot est deja la, on met a jour
# (git pull) au lieu de recloner. Non fatal - un depot injoignable (prive,
# reseau) n arrete pas le reste.
echo -e "  ${CYAN}→${NC} outils OpenCL (deka, a51_tools, dst80_reversing, tea1-cracker) dans /root ..."
# NON-INTERACTIF, SINON LA FENETRE FIGE. Sur un depot prive ou absent, git
# reclame un identifiant sur le terminal et attend INDEFINIMENT - lance par
# l icone, il n y a personne pour repondre. GIT_TERMINAL_PROMPT=0 le fait
# echouer net (pas de prompt), et le message plus bas dit quoi faire.
export GIT_TERMINAL_PROMPT=0
for _rt in "deka=https://github.com/bbaranoff/deka" \
           "a51_tools=https://github.com/bbaranoff/a51_tools" \
           "dst80_reversing=https://github.com/bbaranoff/dst80_reversing" \
           "tea1-cracker=https://github.com/bbaranoff/tea1-cracker"; do
    _name="${_rt%%=*}"; _url="${_rt#*=}"; _dst="/root/$_name"
    # [2026-09-03] Plus de -c http.version=HTTP/1.1 (retire partout dans le
    # depot). On CAPTURE la sortie de git et on la MONTRE en cas d echec.
    GIT="git"
    if [ -d "$_dst/.git" ]; then
        if _err=$($GIT -C "$_dst" pull --ff-only 2>&1); then
            echo -e "      ${GREEN}✓${NC} $_name a jour ($(git -C "$_dst" log -1 --format='%h'))"
        else
            echo -e "      ${YELLOW}!${NC} $_name : git pull a echoue - $(echo "$_err" | tail -1)"
        fi
    elif _err=$($GIT clone --depth 1 "$_url" "$_dst" 2>&1); then
        echo -e "      ${GREEN}✓${NC} $_name clone ($(git -C "$_dst" log -1 --format='%h'))"
    else
        echo -e "      ${YELLOW}!${NC} $_name : clone impossible - $_url"
        echo -e "        ${YELLOW}git:${NC} $(echo "$_err" | tail -1)"
    fi
done

# ── deka : compilation des modules natifs (make) ────────────────────
# deka importe _delta.so et _libvankus.so, produits par SWIG + gcc depuis
# delta.c / libvankus.c (voir son Makefile). Sans ce make, les workers
# importent des modules absents et deka ne calcule rien. Pre-requis : swig4.1
# et les entetes python3. Non fatal : un echec le dit, sans stopper le reste.
if [ -f /root/deka/Makefile ]; then
    echo -e "  ${CYAN}→${NC} deka : compilation des modules natifs (make) ..."
    apt-get install -y --no-install-recommends swig4.1 python3-dev build-essential >/dev/null 2>&1 || true
    if ( cd /root/deka && make >/dev/null 2>&1 ) \
       && [ -e /root/deka/_delta.so ] && [ -e /root/deka/_libvankus.so ]; then
        echo -e "      ${GREEN}✓${NC} _delta.so + _libvankus.so construits"
    else
        echo -e "      ${YELLOW}!${NC} make deka a echoue (swig4.1 / python3-dev ?) - voir : cd /root/deka &&
         ↪  make"
    fi
fi
```

```
    # ── deka : une ICONE d appli (pas un service) ──────────────────────────
    # deka monte des tables et lance des workers - ca ne doit PAS partir a
    # chaque boot dans le dos de l utilisateur (le montage suppose le groupe de
    # volumes present). On pose donc une icone : l utilisateur la clique quand
    # il veut demarrer deka. deka-start.sh reste le moteur ; l icone l appelle
    # avec les droits root (pkexec) dans un terminal, pour voir la sortie.
    if [ -d /root/deka ]; then
        echo -e "  ${CYAN}→${NC} deka : pose de l icone d application ..."
        # deka-start.sh vit dans le depot : s il est la (clone a jour), on n y
        # touche pas. On ne l ecrit qu en SECOURS, pour un clone qui ne l aurait
        # pas encore.
        if [ ! -f /root/deka/deka-start.sh ]; then
        cat > /root/deka/deka-start.sh <<'DEKASTART'
#!/bin/bash
# deka-start.sh - montage des tables deka et lancement des workers.
# Pose par addition.sh (supplement OpenCL). Lance par l icone deka (pkexec), ou
# a la main : sudo /root/deka/deka-start.sh
#   1. vgchange -ay : active le groupe de volumes (sinon /dev/tables/* absent).
#   2. les 4 points de montage, puis les 4 volumes (1_10..31_40).
#   3. les 3 workers python, depuis /root/deka (delta.pyc y vit).
set -u
DIR="$(cd "$(dirname "$(readlink -f "$0")")" && pwd)"
LOG=/var/log/deka.log
# Sortie a l ECRAN + copie dans le log (tee) : on veut VOIR les montages et les
# PID des workers, pas les chercher dans un fichier.
exec > >(tee -a "$LOG") 2>&1
echo "=== $(date -Is) deka-start ==="
[ "$(id -u)" -eq 0 ] || { echo "root requis : sudo $0"; exit 1; }
PY="$(command -v python3.7 || command -v python3)"
[ -n "$PY" ] || { echo "aucun python3 trouve"; exit 1; }
vgchange -ay || echo "ATTENTION: vgchange -ay a echoue (groupe absent ?)"
mkdir -p /mnt1 /mnt2 /mnt3
monter() {
    local dev="$1" pt="$2"
    if mountpoint -q "$pt"; then echo "deja monte: $pt"; return 0; fi
    if [ ! -e "$dev" ]; then echo "ABSENT: $dev - non monte sur $pt"; return 1; fi
    mount "$dev" "$pt" && echo "monte: $dev -> $pt" || echo "ECHEC montage: $dev -> $pt"
}
monter /dev/tables/1_10  /mnt
monter /dev/tables/⟨1⟩_⟨2⟩ /mnt⟨3⟩  ×3
    ⟨⟩ = (11,20,1) (21,30,2) (31,40,3)
cd "$DIR" || exit 1
for w in paplon.py oclvankus.py delta_client.py; do
    [ -f "$w" ] || { echo "worker absent: $w"; continue; }
    echo "lancement: $PY $DIR/$w"
    "$PY" "$DIR/$w" >>"/var/log/deka-${w%.py}.log" 2>&1 &
    echo "  pid $!"
done
echo "deka-start termine."
DEKASTART
        chmod 755 /root/deka/deka-start.sh
        fi

        # LE LANCEUR : deka-start.sh a besoin de root (vgchange, mount). Depuis
        # une icone on n est pas root : pkexec eleve, et un terminal reste
        # ouvert pour lire la sortie (montages, PID des workers). Meme facture
        # que osmo-addition-anim.
        cat > /usr/local/bin/osmo-deka-anim <<'DEKAGUI'
#!/bin/bash
set -u
# osmo-deka-anim - deka dans une fenetre GTK du banc (terminal VTE dedans),
# en root par pkexec. [2026-10-01] Plus de gnome-terminal.
SCRIPT=/root/deka/deka-start.sh
if [ ! -x "$SCRIPT" ]; then
    command -v zenity >/dev/null 2>&1 && \
        zenity --error --text="deka-start.sh introuvable : $SCRIPT" 2>/dev/null
```

```
    exit 1
fi
exec python3 /opt/GSM/osmo-operator/tools/osmo-gtk-run.py --progress --root --title "deka" -- "$SCRIPT" "$@"
DEKAGUI
        chmod 755 /usr/local/bin/osmo-deka-anim

        # ICONE DEDIEE : la mandala de connecteurs (data/deka.svg, facon BRMLAB),
        # posee en CHEMIN ABSOLU dans Icon= - un nom de theme arrive apres
        # icon-theme.cache s afficherait en page blanche (cf. update.sh).
        install -d /usr/share/osmo-operator/icons /usr/share/icons/hicolor/scalable/apps
        if [ -f "$DIR/data/deka.svg" ]; then
            install -m644 "$DIR/data/deka.svg" /usr/share/osmo-operator/icons/deka.svg
            install -m644 "$DIR/data/deka.svg" /usr/share/icons/hicolor/scalable/apps/deka.svg
        fi
        _deka_desktop=/usr/share/applications/deka.desktop
        cat > "$_deka_desktop" <<'DEKADSK'
[Desktop Entry]
Type=Application
Name=deka
Name[fr]=deka
Comment=Monte les tables et lance les workers deka
Comment[fr]=Monte les tables et lance les workers deka
Exec=/usr/local/bin/osmo-deka-anim
Icon=/usr/share/osmo-operator/icons/deka.svg
Terminal=false
Categories=System;Utility;
Keywords=deka;tables;workers;opencl;
Actions=Arreter;

[Desktop Action Arreter]
Name=Arreter deka
Name[fr]=Arreter deka
Exec=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py --root --title "Arret deka" -- /root/deka/deka-start.sh --stop
DEKADSK
        chmod 644 "$_deka_desktop"
        command -v gtk-update-icon-cache >/dev/null 2>&1 && gtk-update-icon-cache -f -q /usr/share/icons/hicolor
        ↪  2>/dev/null || true
        update-desktop-database /usr/share/applications 2>/dev/null || true

        # ── PAS D ICONE DEKA SUR LE BUREAU ──────────────────────────────
        # [2026-09-04] Ce bloc posait deka.desktop sur CHAQUE bureau (/root et
        # tous les /home), en "trusted" pour que DING l affiche, et en 0,0 -
        # donc par-dessus le coin ou le fond GSM LAB attend ses icones. Sur une
        # machine fraichement installee, le supplement OpenCL ajoutait ainsi une
        # icone que personne n avait demandee, et qu il fallait retirer a la
        # main de deux repertoires par compte (Bureau ET Desktop).
        # deka reste lancable : l entree /usr/share/applications/deka.desktop
        # ci-dessus le met dans le menu des applications, et deka-start.sh
        # s appelle directement. Ce qui disparait, c est l icone imposee.
        #
        # On efface aussi celles qu une execution precedente avait posees :
        # sans ca, mettre a jour ne suffisait pas a s en debarrasser.
        for _h in /root /home/*; do
            for _dir in Bureau Desktop; do
                rm -f "$_h/$_dir/deka.desktop" 2>/dev/null || true
            done
        done
        echo -e "  ${GREEN}✓${NC} deka : dans le ${BOLD}menu des applications${NC} (aucune icone posee sur le
        ↪  bureau)"
    fi
fi

# ── CLAUDE CODE ──────────────────────────────────────────────────────────
# Supplement independant : Claude Code, la CLI de l assistant. Rien ici
# n entraine docker ni OpenCL, et rien ne l entraine - on peut le vouloir seul.
#
```

```
# ON PREND L INSTALLEUR NATIF (claude.ai/install.sh), PAS npm, ET C EST VOULU.
# @anthropic-ai/claude-code par npm exige node >= 18 ; or le node des depots
# Ubuntu 22.04 est une v12, trop vieille - l installation npm casserait, ou
# reclamerait un depot tiers (NodeSource) qu on evite ici comme on evite
# get.docker.com plus bas. L installeur natif pose un binaire autonome (node
# embarque), sans toucher au systeme : c est le chemin le plus sur.
#
# Il pose dans ~/.local/bin (donc /root/.local/bin, lance en sudo). On lie ce
# binaire dans /usr/local/bin pour qu il soit sur le PATH de tout le monde et
# que la sonde `command -v claude` le voie.
if [ "$DO_CLAUDE" = "1" ]; then
    echo -e "  ${CYAN}→${NC} installation de Claude Code ..."
    export DEBIAN_FRONTEND=noninteractive

    # curl + certificats : l installeur les exige. Non fatal en soi, mais on
    # previent tot si le reseau manque - rien n est pre-telecharge dans l ISO.
    command -v curl >/dev/null 2>&1 || apt-get install -y curl ca-certificates >/dev/null 2>&1 || true
    if ! command -v curl >/dev/null 2>&1; then
        echo -e "  ${RED}✗ curl absent et non installable - pas de reseau ?${NC}"
        echo -e "    Ce supplement a BESOIN d Internet : rien n est pre-telecharge dans l ISO."
        exit 1
    fi

    # HOME explicite : lance par pkexec/sudo, $HOME peut manquer ou pointer
    # ailleurs ; l installeur pose dans $HOME/.local/bin, on veut savoir ou.
    _cc_home="${HOME:-/root}"
    if _out=$(HOME="$_cc_home" bash -c 'curl -fsSL https://claude.ai/install.sh | bash' 2>&1); then
        echo -e "      ${GREEN}✓${NC} installeur natif termine"
    else
        echo -e "      ${YELLOW}!${NC} installeur natif en echec - $(echo "$_out" | tail -1)"
    fi

    # Trouver le binaire pose et le lier sur le PATH global. On regarde les
    # emplacements connus de l installeur (~/.local/bin, puis repli /root).
    _cc_bin=""
    for _c in "$_cc_home/.local/bin/claude" /root/.local/bin/claude "$_cc_home/.claude/bin/claude"; do
        [ -x "$_c" ] && { _cc_bin="$_c"; break; }
    done
    if [ -n "$_cc_bin" ] && [ ! -e /usr/local/bin/claude ]; then
        ln -sf "$_cc_bin" /usr/local/bin/claude
        echo -e "      ${GREEN}✓${NC} lien /usr/local/bin/claude -> $_cc_bin"
    fi

    # LA SEULE VERIFICATION QUI VAUT : est-ce que claude repond ?
    if command -v claude >/dev/null 2>&1 && _ver=$(claude --version 2>/dev/null); then
        echo -e "  ${GREEN}✓${NC} Claude Code operationnel - ${BOLD}${_ver}${NC}"
        echo -e "      lancez-le avec la commande ${BOLD}claude${NC}"
    else
        echo -e "  ${YELLOW}!${NC} Claude Code pose, mais ne repond pas encore."
        echo -e "      Ouvrez un nouveau terminal (PATH), ou lancez ${BOLD}$_cc_home/.local/bin/claude${NC}"
    fi

    # ── ICONE CLAUDE : tuile clay + sunburst Anthropic (data/claude.svg) ──────
    # Lanceur osmo-claude-anim : ouvre un terminal qui lance claude ; s il n est
    # pas installe, il enchaine ce meme supplement (--claude) puis le lance.
    install -d /usr/share/osmo-operator/icons /usr/share/icons/hicolor/scalable/apps
    if [ -f "$DIR/data/claude.svg" ]; then
        install -m644 "$DIR/data/claude.svg" /usr/share/osmo-operator/icons/claude.svg
        install -m644 "$DIR/data/claude.svg" /usr/share/icons/hicolor/scalable/apps/claude.svg
    fi
    cat > /usr/local/bin/osmo-claude-anim <<'CLA'
#!/bin/bash
# osmo-claude-anim - Claude Code dans une fenetre GTK du banc (terminal VTE :
# claude est interactif de bout en bout). Pas dans l ISO : le premier clic
# l installe (addition.sh --claude, en root) puis l ouvre, dans la meme
# fenetre. [2026-10-01] Plus de gnome-terminal.
```

```
set -u
exec python3 /opt/GSM/osmo-operator/tools/osmo-gtk-run.py --tty --title "Claude" -- bash -lc '
if ! command -v claude >/dev/null 2>&1; then
    ADD=/opt/GSM/osmo-operator/addition.sh
    if [ -x "$ADD" ]; then
        if [ "$(id -u)" -ne 0 ] && command -v pkexec >/dev/null 2>&1; then
            pkexec env DISPLAY="${DISPLAY:-}" XAUTHORITY="${XAUTHORITY:-}" "$ADD" --claude
        else
            "$ADD" --claude
        fi
    fi
fi
if command -v claude >/dev/null 2>&1; then exec claude; fi
echo "Claude non installe - lancez le supplement (addition.sh --claude)."
exit 1'


CLA
    chmod 755 /usr/local/bin/osmo-claude-anim
    if [ -f "$DIR/data/desktop/claude.desktop" ]; then
        install -m644 "$DIR/data/desktop/claude.desktop" /usr/share/applications/claude.desktop
       sed -i "s|^Icon=.*|Icon=/usr/share/osmo-operator/icons/claude.svg|" /usr/share/applications/claude.desktop
        command -v gtk-update-icon-cache >/dev/null 2>&1 && gtk-update-icon-cache -f -q /usr/share/icons/hicolor
         ↪  2>/dev/null || true
        update-desktop-database /usr/share/applications 2>/dev/null || true
        mapfile -t _homes < <(_bureau_homes)
        _cposee=0
        for _h in "${_homes[@]}"; do
            for _dir in Bureau Desktop; do
                [ -d "$_h/$_dir" ] || continue
                cp -f /usr/share/applications/claude.desktop "$_h/$_dir/claude.desktop" 2>/dev/null || continue
                _trust_desktop "$_h/$_dir/claude.desktop" "$_h" "110,0"
                touch "$_h/$_dir" 2>/dev/null || true
                _cposee=1
            done
        done
        [ "$_cposee" = "1" ] \
            && echo -e "  ${GREEN}✓${NC} icone ${BOLD}Claude${NC} posee sur le bureau (en haut a gauche)" \
            || echo -e "  ${GREEN}✓${NC} icone ${BOLD}Claude${NC} dans le menu des applications"
    fi
fi

# ── DOCKER ──────────────────────────────────────────────────────────
# docker.io des depots Ubuntu, PAS le script get.docker.com : celui-ci ajoute un
# depot tiers et remplace containerd, ce qui sur une image figee se paie au
# premier apt-get upgrade. La version des depots suffit : on ne fait tourner
# que nos propres conteneurs.
if [ "$DO_DOCKER" = "1" ]; then
    if command -v docker >/dev/null 2>&1; then
        echo -e "  ${GREEN}✓${NC} docker deja installe ($(docker --version 2>/dev/null | head -1))"
    else
        echo -e "  ${CYAN}→${NC} installation de docker.io ..."
        export DEBIAN_FRONTEND=noninteractive
        apt-get update || { echo -e "  ${RED}✗ apt-get update a echoue - pas de reseau ?${NC}"
                            echo -e "    Ce supplement a BESOIN d Internet : rien n est pre-telecharge dans l ISO."
                            exit 1; }
        apt-get install -y docker.io || { echo -e "  ${RED}✗ installation de docker.io echouee${NC}"; exit 1; }
        echo -e "  ${GREEN}✓${NC} docker.io installe"
    fi
    # docker compose build passe par Bake et avertit si buildx manque. Non fatal.
    if ! docker buildx version >/dev/null 2>&1; then
        apt-get install -y docker-buildx >/dev/null 2>&1 \
            || apt-get install -y docker-buildx-plugin >/dev/null 2>&1 || true
    fi
    systemctl enable --now docker 2>/dev/null || true
    # Le socket met un instant a repondre apres un premier demarrage : sans
```

```
    # cette attente, le `docker info` suivant echoue et l on croit
    # l installation ratee.
    for _i in $(seq 1 30); do docker info >/dev/null 2>&1 && break; sleep 1; done
    docker info >/dev/null 2>&1 \
        && echo -e "  ${GREEN}✓${NC} demon docker actif" \
        || { echo -e "  ${RED}✗ le demon docker ne repond pas${NC} - voir : systemctl status docker"; exit 1; }
    _docker_groupe_session
fi

# ── IMAGE OPERATEUR ──────────────────────────────────────────────────
# Deux chemins pour obtenir osmocom-nitb :
#   1. docker pull ${HUB_IMAGE} && docker tag ... osmocom-nitb  (defaut)
#      La pile Osmocom deja compilee, publiee sur le hub : le telechargement
#      remplace 40 minutes de compilation. Le tag donne au resultat le nom que
#      start-docker.sh (Dockerfile.run : FROM osmocom-nitb) et start-nitb.sh attendent.
#   2. build.sh --no-cache  (--build / --multi-build / case "compiler")
#      --no-cache : le supplement rebatit l image PROPRE. Le cache docker
#      gardait des couches d une pile Osmocom a moitie compilee (ex. le clone
#      github casse de libosmocore) et les rejouait a l identique - un build
#      deja casse restait casse a chaque passage. On force la reconstruction.
# [2026-09-13] COMPILER, C EST DESORMAIS TIRER DE GHCR.
# --build / --multi-build compilaient sur place : plusieurs dizaines de minutes
# de pile Osmocom, sur la machine de l operateur, pour obtenir exactement ce
# que la CI vient de publier a partir du meme Dockerfile. Le resultat est le
# meme a l octet pres quand le depot n a pas bouge ; le temps, non. Ces deux
# drapeaux font donc le pull, et gardent leur sens : « prends l image de CE
# depot », par opposition a une image du Hub.
# Le vrai build local reste accessible - il le faut, c est le seul chemin quand
# on a MODIFIE le Dockerfile et qu'aucune CI n a encore publie :
#     sudo OSMO_ADDITION_BUILD_LOCAL=1 ./addition.sh --multi-build
_build_image() {
    if [ "${OSMO_ADDITION_BUILD_LOCAL:-0}" != "1" ]; then
        echo -e "  ${CYAN}→${NC} image du depot : ${BOLD}docker pull ${HUB_IMAGE}${NC}"
        echo -e "      (build local : ${BOLD}OSMO_ADDITION_BUILD_LOCAL=1${NC}, plusieurs dizaines de minutes)"
        _pull_image
        return $?
    fi
    [ -x "$DIR/build.sh" ] || { echo -e "  ${RED}✗ build.sh introuvable dans $DIR${NC}"; return 1; }
    echo -e "  ${CYAN}→${NC} construction de l image : ${BOLD}${DIR}/build.sh --no-cache${NC}"
    echo -e "      compilation Osmocom - comptez plusieurs dizaines de minutes."
    "$DIR/build.sh" --no-cache || { echo -e "  ${RED}✗ ${DIR}/build.sh a echoue${NC}"; return 1; }
    echo -e "  ${GREEN}✓${NC} image operateur construite (${BASE_IMAGE})"
}
_pull_image() {
    echo -e "  ${CYAN}→${NC} telechargement de l image : ${BOLD}docker pull ${HUB_IMAGE}${NC}"
    echo -e "      ~11 Go - selon le debit, quelques minutes a une heure."
    # GHCR sert les images PUBLIQUES sans authentification. Si le paquet du
    # depot est prive, il faut un jeton : docker login ghcr.io -u <compte>
    # (jeton classique avec read:packages). Le message le dit plutot que de
    # laisser l operateur devant un "denied" sans explication.
    docker pull "$HUB_IMAGE" || {
        echo -e "  ${RED}✗ docker pull ${HUB_IMAGE} a echoue${NC} - reseau, image absente, ou paquet prive."
        case "$HUB_IMAGE" in ghcr.io/*)
            echo -e "    Paquet prive ? ${BOLD}docker login ghcr.io -u <compte>${NC} (jeton avec read:packages)" ;;
        esac
        echo -e "    Autre image : ${BOLD}HUB_IMAGE=<ref> $0 --multi${NC}"
        echo -e "    Compiler sur place : ${BOLD}OSMO_ADDITION_BUILD_LOCAL=1 $0 --multi-build${NC}"
        return 1; }
    # [2026-10-24] norf_gsm est publiee depuis osmocom-run (Dockerfile.run deja
    # joue) : on la tague osmocom-run directement, sans derivation.
    local _tag="$BASE_IMAGE"; [ "${HUB_IS_RUN:-0}" = 1 ] && _tag="$MULTI_IMAGE"
    docker tag "$HUB_IMAGE" "$_tag" || { echo -e "  ${RED}✗ docker tag ${HUB_IMAGE} ${_tag} a echoue${NC}"; return
     ↪  1; }
    echo -e "  ${GREEN}✓${NC} image operateur tiree et taguee ${_tag}"
}
# L IMAGE QUE LE MULTI-OPERATEUR LANCE EST osmocom-run, PAS osmocom-nitb.
```

```
# Ni le pull ni build.sh ne la produisent : c est Dockerfile.run (FROM
# osmocom-nitb) qui la derive, en quelques secondes. start-multi.sh sonde
# osmocom-run et renvoie ici si elle manque : sans cette derivation, un
# supplement "reussi" laissait le multi-operateur incapable de demarrer.
# Meme commande que start-docker.sh (build_run_image, mode quick).
_derive_run_image() {
    echo -e "  ${CYAN}→${NC} derivation de l image d execution : ${BOLD}docker build -f Dockerfile.run -t
     ↪  ${MULTI_IMAGE}${NC}"
    ( cd "$DIR" && docker build --build-arg QEMU_CACHE_BUST=$(date +%s) \
                      -f Dockerfile.run -t "$MULTI_IMAGE" . ) \
        || { echo -e "  ${RED}✗ derivation de ${MULTI_IMAGE} echouee${NC}"; return 1; }
    echo -e "  ${GREEN}✓${NC} image d execution prete (${MULTI_IMAGE})"
}
# norf_gsm (choix « hub », ou HUB_IMAGE pose par l installeur) EST osmocom-run.
case "$HUB_IMAGE" in *norf_gsm*) HUB_IS_RUN=1 ;; esac
if [ "$DO_IMAGE" = "1" ]; then
    if [ "$DO_BUILD" = "1" ]; then
        _build_image || exit 1
        _derive_run_image || exit 1
    elif docker image inspect "$MULTI_IMAGE" >/dev/null 2>&1; then
        echo -e "  ${GREEN}✓${NC} image operateur deja presente (${MULTI_IMAGE})"
    elif [ "${HUB_IS_RUN:-0}" != 1 ] && docker image inspect "$BASE_IMAGE" >/dev/null 2>&1; then
        echo -e "  ${GREEN}✓${NC} image de base deja presente (${BASE_IMAGE})"
        _derive_run_image || exit 1
    else
        _pull_image || exit 1
        [ "${HUB_IS_RUN:-0}" = 1 ] || _derive_run_image || exit 1
    fi
fi

# ── TOPOLOGIE MULTI-OPERATEUR ──────────────────────────────────────────
# Ecrite ICI, une fois, et relue par start-multi.sh. Elle n est pas recalculee
# des deux cotes : deux formules jumelles finissent toujours par diverger, et le
# depot en porte deja la trace (start-docker.sh et lib/gabarits.sh tiennent les MEMES
# fonctions en double).
#
# LE PLAN SS7, tel que start-interstp.sh le definit :
#     PC   = 1.<noeud><op>.<role>      role 1=MSC 2=STP 3=BSC
#     RCTX = noeud*1000 + op*100 + 50
# Sur une seule machine le noeud vaut 1, donc PC = 1.<op>.<role>.
#
#   op 1  NATIF   - PC 1.1.2, celui de start-native.sh, laisse tel quel
#   op 2  DOCKER  - PC 1.2.2, backbone 172.20.0.12
#   op 3  DOCKER  - PC 1.3.2, backbone 172.20.0.13
#
# ⚠ LES CONTENEURS COMMENCENT A 2, ET CELA SE PILOTE.
# start-docker.sh numerotait ses conteneurs 1..N en dur : avec --operators 2 il creait
# osmo-operator-1 en point code 1.1.2, celui du natif. Deux equipements a la
# meme adresse SS7 - "pas un conflit de nom, du routage faux"
# (checks/wan_ss7_check.sh). OP_ID_BASE=2 (start-docker.sh) fait demarrer les
# conteneurs au rang 2 et laisse le 1 au natif ; nom, backbone, point code,
# RCTX, segment prive et ports publies en decoulent tous.
# Le natif joint le hub par la passerelle du bridge : l hote atteint
# 172.20.0.10 directement, sans NAT ni route a ajouter.
if [ "$DO_MULTI" = "1" ]; then
    install -d "$(dirname "$MULTI_CONF")"
    cat > "$MULTI_CONF" <<CONF
# osmo-multi.conf - topologie multi-operateur, ecrite par addition.sh.
# Relue par start-multi.sh. Ne pas editer a la main sans relire les deux.
MULTI_NODE=1
MULTI_HUB_NAME=osmo-inter-stp
MULTI_HUB_IP=172.20.0.10
MULTI_HUB_PC=0.0.0
MULTI_M3UA_PORT=2908
MULTI_NET_NAME=osmo-ss7
MULTI_NET_SUBNET=172.20.0.0/24
```

```
MULTI_NET_GW=172.20.0.1
MULTI_IMAGE=osmocom-run
# operateurs : "index:mode:backbone_ip:point_code:rctx"
MULTI_OPS="1:native::1.1.2:150 2:docker:172.20.0.12:1.2.2:250 3:docker:172.20.0.13:1.3.2:350"
CONF
    echo -e "  ${GREEN}✓${NC} topologie SS7 ecrite : ${CYAN}${MULTI_CONF}${NC}"
    echo -e "      op1 ${BOLD}natif${NC} (1.1.2) + op2/op3 ${BOLD}docker${NC} (1.2.2 / 1.3.2) → hub 172.20.0.10"

    # ── L ICONE MULTI-OPERATOR : POSEE ICI, A LA FIN, ET PAS AVANT ──────────
    # L antenne bleue lance start-multi.sh, qui suppose docker, l image et la
    # topologie ci-dessus. La poser au build ISO (comme le telephone ou le
    # tutoriel) la faisait apparaitre sur le bureau des le premier boot, AVANT
    # que ce supplement n existe : un clic ne donnait rien. On la pose donc
    # seulement maintenant, une fois l install SS7 reellement faite - build-iso
    # et install_modules/80-bureau.sh ne la posent plus.
    _ico_dir=/usr/share/osmo-operator/icons
    _svg="$DIR/data/osmo-multi.svg"
    _dsk="$DIR/data/desktop/osmo-multi.desktop"

    # ── L UNITE D ABORD, L ICONE ENSUITE ─────────────────────────────────────
    # [2026-09-04] L icone lance desormais `systemctl start osmo-multi.service`
    # et plus start-multi.sh : elle ne vaut donc que si l unite EXISTE. Sur une
    # ISO elle est posee au build (iso_modules/82-services.sh) ; sur une machine
    # ou ce depot a ete deroule a la main, non. On la pose ici, avec l icone qui
    # en depend - les deux vont ensemble ou ne vont pas du tout.
    #
    # POURQUOI PLUS DE TERMINAL. start-multi.sh se RELANCE lui-meme dans un
    # gnome-terminal quand il detecte qu il n en a pas (OSMO_MULTI_TERM) : un
    # clic sur l icone ouvrait donc une fenetre qui vivait le temps du
    # lancement, et dont la sortie etait perdue a la fermeture. Quand un
    # operateur ne demarrait pas, il n en restait aucune trace. L unite, elle,
    # pose OSMO_NO_TERM=1 : aucune fenetre, tout dans le journal
    # (`journalctl -u osmo-multi`), un etat lisible et un arret propre - comme
    # le mono-operateur, qui est une unite depuis le debut.
    for _u in osmo-banc osmo-multi; do
        [ -f "$DIR/services/$_u.service" ] || continue
        install -m644 "$DIR/services/$_u.service" "/etc/systemd/system/$_u.service"
    done
    systemctl daemon-reload 2>/dev/null || true
    # ── POSEES, PAS ACTIVEES ─────────────────────────────────────────────────
    # [2026-09-05] `systemctl enable osmo-multi` etait fait ICI, et l ISO
    # activait osmo-banc de son cote : le supplement SS7 suffisait donc a ce
    # que la machine monte, a chaque demarrage, la pile radio ET les conteneurs
    # du multi - avant meme l ouverture de session, sur une configuration que
    # personne n avait choisie. Un banc se DEMARRE :
    #   systemctl start osmo-banc | osmo-multi     une session (ou l icone)
    #   systemctl enable --now osmo-multi          a chaque demarrage
    # OSMO_MULTI_ENABLE=1 ./addition.sh retablit l activation automatique.
    if [ -f /etc/systemd/system/osmo-multi.service ]; then
        if [ "${OSMO_MULTI_ENABLE:-0}" = "1" ]; then
            systemctl enable osmo-multi.service >/dev/null 2>&1 \
                && echo -e "  ${GREEN}✓${NC} ${BOLD}osmo-multi.service${NC} installe et active au demarrage
                ↪  (OSMO_MULTI_ENABLE=1)" \
                || echo -e "  ${YELLOW}!${NC} osmo-multi.service installe, activation KO (systemctl enable
                ↪  osmo-multi)"
        else
            systemctl disable osmo-multi.service >/dev/null 2>&1 || true
           echo -e "  ${GREEN}✓${NC} ${BOLD}osmo-multi.service${NC} installe, ${BOLD}non active${NC} au demarrage"
            echo -e "      ${CYAN}systemctl start osmo-multi${NC} pour une session · ${CYAN}systemctl enable --now
             ↪  osmo-multi${NC} a chaque boot"
        fi
    fi

    if [ -f "$_dsk" ]; then
        install -d "$_ico_dir" /usr/share/icons/hicolor/scalable/apps /usr/share/applications
        # SVG en CHEMIN ABSOLU dans Icon= (voir build-iso.sh) : un nom de theme
```

```
    # arrive apres icon-theme.cache s affiche en page blanche.
    if [ -f "$_svg" ]; then
        install -m644 "$_svg" "$_ico_dir/osmo-multi.svg"
        install -m644 "$_svg" /usr/share/icons/hicolor/scalable/apps/osmo-multi.svg
    fi
    install -m644 "$_dsk" /usr/share/applications/osmo-multi.desktop
    # Icon= suit l arbre reel. Exec=, lui, ne nomme plus l arbre : il appelle
    # `systemctl start osmo-multi.service`, et c est l UNITE qui porte le
    # chemin (posee juste au-dessus depuis $DIR/services/). Le sed sur Exec=
    # reste, sans effet sur la ligne actuelle, pour un .desktop ancien qui
    # nommerait encore start-multi.sh en dur.
    sed -i -e "s|^Exec=/opt/GSM/osmo-operator/|Exec=$DIR/|" \
           -e "s|^Icon=.*|Icon=$_ico_dir/osmo-multi.svg|" \
           /usr/share/applications/osmo-multi.desktop

    # Sur les bureaux de root et de l utilisateur sudo, en Bureau/ et
    # Desktop/. metadata::trusted pour que DING l affiche sans pastille ;
    # osmo-trust-desktop (au login) la repositionne et la trust aussi.
    mapfile -t _homes < <(_bureau_homes)
    _posee=0
    for _h in "${_homes[@]}"; do
        for _dir in Bureau Desktop; do
            [ -d "$_h/$_dir" ] || continue
            cp -f /usr/share/applications/osmo-multi.desktop "$_h/$_dir/osmo-multi.desktop" 2>/dev/null ||
             ↪  continue
            _trust_desktop "$_h/$_dir/osmo-multi.desktop" "$_h"
            touch "$_h/$_dir" 2>/dev/null || true
            _posee=1
        done
    done
    command -v gtk-update-icon-cache >/dev/null 2>&1 && gtk-update-icon-cache -f -q /usr/share/icons/hicolor
     ↪  2>/dev/null || true
    command -v update-desktop-database >/dev/null 2>&1 && update-desktop-database /usr/share/applications
     ↪  2>/dev/null || true
    [ "$_posee" = "1" ] \
        && echo -e "  ${GREEN}✓${NC} icone ${BOLD}multi-operator${NC} (antenne) posee sur le bureau" \
        || echo -e "  ${GREEN}✓${NC} icone ${BOLD}multi-operator${NC} dans le menu des applications"
fi

# ── LE BANC MULTI DEMARRE MAINTENANT : LES CONTENEURS EXISTENT EN SORTANT ──
# [2026-10-01] L en-tete promettait « start-multi.sh demarre juste apres »
# et RIEN ne le faisait : le supplement sortait avec une topologie, une
# unite et une icone, mais aucun conteneur - osmo-operator-2 et -3
# n existaient qu au premier clic sur l antenne, et un operateur qui venait
# d « installer le multi » ne voyait rien dans docker ps. On termine donc
# par UNE session du banc multi : `systemctl start`, pas `enable` -
# l activation a chaque demarrage reste le geste de l operateur
# (00-options.sh, 2026-09-14). OSMO_MULTI_START=0 pour poser sans lancer.
# Le journal de l unite defile ici pendant le demarrage : c est lui qui
# montre les conteneurs qui montent, et l erreur s il y en a une.
#
# [2026-10-01] PAR launch.sh --multi, LE MEME CHEMIN QUE L ICONE. Ce bloc
# faisait son propre `systemctl start --no-block` : un deuxieme lanceur,
# avec ses propres tests - et pas ceux de l icone. launch.sh --multi
# verifie l unite, start-multi.sh et la topologie, nomme ce qui manque,
# refuse un banc deja en route, demarre l unite (start, pas restart) et
# annonce le resultat ; c est aussi ce que fait « run multi » de l encart
# du bureau (tools/osmo-launcher.py). Ce que le supplement vient de poser
# est donc teste par ce qui s en servira. Le natif que start-multi.sh
# lance au passage (osmo-banc) part en DSP : c est le defaut de
# start-native.sh depuis ce jour.
if [ "${OSMO_MULTI_START:-1}" = "1" ] && [ -f /etc/systemd/system/osmo-multi.service ]; then
    echo -e "  ${CYAN}→${NC} demarrage du banc multi-operateur (${BOLD}launch.sh --multi${NC}, comme l icone)
     ↪  - comptez plusieurs minutes ..."
    journalctl -u osmo-multi -f -n 0 --no-pager -o cat 2>/dev/null &
    _jpid=$!
```

```
        # OSMO_LAUNCH_NO_SERVICE n a pas de sens ici (--multi est traite en
        # tete de launch.sh) ; on garde le terminal courant pour sa sortie.
        if "$DIR/launch.sh" --multi; then
            sleep 1; kill "$_jpid" 2>/dev/null; wait "$_jpid" 2>/dev/null
            echo -e "  ${GREEN}✓${NC} banc multi-operateur en service - conteneurs :"
            docker ps --format '      {{.Names}}  {{.Status}}' 2>/dev/null | grep -E
             ↪  'osmo-operator-|osmo-inter-stp' \
                || echo -e "      ${YELLOW}!${NC} aucun conteneur osmo-operator-N visible (docker ps)"
        else
            sleep 1; kill "$_jpid" 2>/dev/null; wait "$_jpid" 2>/dev/null
            echo -e "  ${RED}✗${NC} launch.sh --multi n a pas abouti - ${BOLD}journalctl -u osmo-multi -n 80${NC}"
            echo -e "      a la main : ${BOLD}$DIR/launch.sh --multi${NC} (ou l icone multi-operator)"
        fi
    fi
fi

# ── LA SIM DU BANC (supplement independant) ──────────────────────────
# Lecteur PC/SC (pcscd, CCID : HID Omnikey), softSIM (le serveur SAP, Ruby) et
# pySim (lire / programmer une sysmoISIM). Meme code que l ISO (80-chroot.sh) :
# tools/osmo-sim-install.sh, qu on source ici. Ensuite : tools/osmo-sim.sh.
if [ "$DO_SIM" = "1" ]; then
    echo -e "${BOLD}== SIM : lecteur PC/SC, softSIM, pySim ==${NC}"
    if [ -f "$DIR/tools/osmo-sim-install.sh" ]; then
        # shellcheck source=tools/osmo-sim-install.sh
        . "$DIR/tools/osmo-sim-install.sh"
        REPO="$DIR" osmo_sim_install
    else
        echo -e "  ${YELLOW}!${NC} tools/osmo-sim-install.sh introuvable"
    fi
fi

# ── FIRMWARES TELEPHONE (supplement independant) ─────────────────────
# gnuarm, Magnetite C139, osmocom-bb-trx, Fernvale, SIMCom : tools/osmo-firmwares-install.sh.
# osmocom-bb-trx tire une image Docker : sans moteur, --trx ne pose que le firmware.
if [ "$DO_GNUARM$DO_MAGNETITE$DO_TRX$DO_CALFW$DO_FERNVALE$DO_SIMCOM" != "000000" ]; then
    if [ -f "$DIR/tools/osmo-firmwares-install.sh" ]; then
        # shellcheck source=tools/osmo-firmwares-install.sh
        . "$DIR/tools/osmo-firmwares-install.sh"
        export REPO="$DIR"
        [ "$DO_GNUARM" = "1" ]    && { osmo_gnuarm_install    || true; }
        [ "$DO_MAGNETITE" = "1" ] && { osmo_magnetite_install || true; }
        [ "$DO_TRX" = "1" ]       && { osmo_trx_install       || true; }
        [ "$DO_CALFW" = "1" ]     && { osmo_calypso_fw_install || true; }
        [ "$DO_FERNVALE" = "1" ]  && { osmo_fernvale_install  || true; }
        [ "$DO_SIMCOM" = "1" ]    && { osmo_simcom_install    || true; }
        echo -e "  ${CYAN}→${NC} journaux : ${BOLD}${OSMO_FW_LOG:-/var/log/osmo-firmwares}/${NC}"
    else
        echo -e "  ${YELLOW}!${NC} tools/osmo-firmwares-install.sh introuvable"
    fi
fi

# ── JEUX + MEDIA (supplement independant) ────────────────────────────
# Doom / Quake III / Kodi / YouTube+uBlock / Wireshark(root) / Linphone,
# ranges dans les dossiers Jeux / Media / Telephone / Outils et lancables depuis
# l encart du bureau. La logique est partagee avec l ISO (le natif) :
# tools/osmo-extras-install.sh, qu on source ici.
if [ "$DO_EXTRAS" = "1" ]; then
    echo -e "${BOLD}== Jeux + media ==${NC}"
    if [ -f "$DIR/tools/osmo-extras-install.sh" ]; then
        # shellcheck source=tools/osmo-extras-install.sh
        . "$DIR/tools/osmo-extras-install.sh"
        osmo_extras_install
    else
        echo -e "  ${YELLOW}!${NC} tools/osmo-extras-install.sh introuvable"
    fi
fi
```

```
# ── LA 4G (srsRAN ZeroMQ + Open5GS) ET L UI SMARTPHONE (postmarketOS) ──────
# [2026-09-08] Miroir aux trois chemins (addition.sh, update.sh, le build ISO :
# iso_modules/88-lte-pmos.sh). Tout est ecrit une fois dans
# tools/osmo-lte-install.sh (paquets, .deb du cache, configs du depot,
# lanceurs osmo-lte / osmo-epc ; --all compile dans /opt/LTE si les binaires
# manquent et qu aucun .deb n est la) et tools/osmo-pmos-install.sh
# (/opt/user_interface/pmos : scripts, pmbootstrap patche, gabarit ; kernel/ :
# le .deb du noyau PPP, de /var/cache/osmo-debs ou de GitHub). Non fatal.
# [2026-10-24] SAUF QUAND CLAUDE EST SEUL DEMANDE. « addition.sh --claude » (et
# l icone Claude, qui l appelle au premier clic) ne doit installer que Claude :
# ce bloc, lui, passait a chaque appel et lancait la 4G et postmarketOS
# (paquets, eventuelle compilation de /opt/LTE) pour un simple CLI.
if [ "$DO_CLAUDE" = "1" ] && [ "$DO_DOCKER$DO_IMAGE$DO_MULTI$DO_OPENCL$DO_EXTRAS$DO_SIM" = "000000" ] \
   && [ "$DO_GNUARM$DO_MAGNETITE$DO_TRX$DO_CALFW$DO_FERNVALE$DO_SIMCOM" = "000000" ]; then
    :
else
echo -e "${BOLD}== La 4G et le telephone ==${NC}"
if [ -f "$DIR/tools/osmo-lte-install.sh" ]; then
    # shellcheck source=tools/osmo-lte-install.sh
    . "$DIR/tools/osmo-lte-install.sh"
    osmo_lte_install --all || true
else
    echo -e "  ${YELLOW}!${NC} tools/osmo-lte-install.sh introuvable"
fi
if [ -f "$DIR/tools/osmo-pmos-install.sh" ]; then
    # shellcheck source=tools/osmo-pmos-install.sh
    . "$DIR/tools/osmo-pmos-install.sh"
    osmo_pmos_install || true
else
    echo -e "  ${YELLOW}!${NC} tools/osmo-pmos-install.sh introuvable"
fi
fi

echo
etat
echo
[ "$DO_MULTI" = "1" ] && \
    echo -e "  ${CYAN}→${NC} lancer : ${BOLD}sudo $DIR/launch/start-multi.sh${NC}  (ou l antenne bleue du bureau)"
[ "$DO_OPENCL" = "1" ] && \
    echo -e "  ${CYAN}→${NC} verifier OpenCL : ${BOLD}clinfo${NC}"
[ "$DO_EXTRAS" = "1" ] && \
    echo -e "  ${CYAN}→${NC} jeux + media : dossiers ${BOLD}Jeux / Media / Telephone / Outils${NC} et encart du
     ↪  bureau"
exit 0
```

3.7 /opt/GSM/osmo-operator/build-iso.sh

5399 octets, 82 lignes → 82 lignes

```
#!/bin/bash
# build-iso.sh - Genere une ISO bootable en utilisant build.sh et start-docker.sh
# Aucun docker build direct dans ce script, tout passe par les scripts existants.
#
#  Usage : sudo ./build-iso.sh [options]
#    (sans option)        les quatre images amd64 : interstp, operator, lite, desktop
#    --role=operator|interstp   --node=N   --lite   --desktop   --all
#    --arm                images SD arm64 par github.com/armbian/build
#                         (armbian/build-armbian.sh) : Raspberry Pi 4 et toutes
#                         les Orange Pi 5 (5, 5B, 5 Plus, 5 Pro, 5 Max, 5 Ultra) ;
#                         --board=rpi4|opi5|all|<carte>[,...] (defaut all),
#                         --desktop = GNOME + le bureau du banc ; --all et --lite
#                         refuses. Voir iso_modules/00-options.sh.
#    --banc --multi       active osmo-banc.service / osmo-multi.service au boot
```

```
#                         de l'image. Par defaut les deux unites sont POSEES mais
#                         NON activees : la machine demarre sur son bureau, et
#                         c'est l'operateur qui lance son banc (icone, launch.sh,
#                         systemctl start osmo-banc). Equivalents en variables :
#                         OSMO_ISO_BANC=1 / OSMO_ISO_MULTI=1.
#    --version=24.04|22.04   --kb=fr   --output=fichier   --no-cache
#    (defaut, amd64)      L'IMAGE DOCKER EST TIREE, PAS CONSTRUITE, et taguee
#                         osmocom-nitb. Dans l'ordre : bastienbaranoff/norf_gsm:latest
#                         (Docker Hub), puis ghcr.io/<depot>/osmocom-nitb:base-<empreinte>,
#                         puis le :latest de GHCR.
#                         build.sh ne tourne que si aucune ne repond. L'empreinte
#                         est celle des workflows : quand elle correspond, l'image
#                         publiee a ete batie sur exactement cet arbre de travail ;
#                         les deux autres references ne le garantissent pas, et le
#                         script le dit avant de s'en servir. En arm64 rien n'est
#                         publie : la compilation reste le defaut.
#    --skip-build[=image:tag]   rend le pull obligatoire (pas de repli sur
#                         build.sh) et, avec =image, impose la reference au lieu
#                         de la deduire. Une image deja presente localement
#                         n'est pas retiree. Variable : OSMO_ISO_SKIP_BUILD=0
#                         pour l'inverse - ne jamais tirer.
#    --build-docker       l'inverse, et il gagne : l'image est construite par
#                         build.sh meme si un --skip-build traine par ailleurs.
#                         Variable : OSMO_ISO_BUILD_DOCKER=1.
#    --without-debs       aucun .deb ne voyage dans l'image : ni les paquets du
#                         banc (etape 7c), ni ceux du build docker. Ils servent
#                         toujours a poser la pile dans le rootfs, ils n'y
#                         restent plus. Variable : OSMO_ISO_WITHOUT_DEBS=1.
#    --wan --wan-nodes=... --wan-id=N --wan-ops=N --hub-ip=IP
set -euo pipefail

# Couleurs definies AVANT tout : sous `set -u`, la premiere ligne coloree d'un
# script qui les declare plus bas echoue sur "variable sans liaison" - et le
# message ne parle ni de couleurs ni de l'endroit fautif.
RED='\033[0;31m'; GREEN='\033[0;32m'; CYAN='\033[0;36m'
YELLOW='\033[1;33m'; NC='\033[0m'; BOLD='\033[1m'

VERSION="${OSMO_ISO_VERSION:-main}"
OUTPUT="osmo-operator-${VERSION}.iso"
# Repertoire de travail SUR DISQUE (pas /tmp, souvent un tmpfs en RAM -> "No
# space left on device" car le rootfs est volumineux). Overridable via OSMO_ISO_WORK.
WORK="${OSMO_ISO_WORK:-/var/tmp}/iso-build-$$"
ROOTFS="$WORK/rootfs"
ISOROOT="$WORK/isoroot"
LABEL="OSMO_EGPRS_V2"
DIR="$(cd "$(dirname "$0")" && pwd)"
# /usr/local/sbin DANS le PATH : apt-fast-install pose apt-fast la, sur l hote
# comme dans le rootfs. Le chroot plus bas herite de ce PATH (env ... bash -c) :
# sans lui, "[apt-fast] pret : /usr/local/sbin/apt-fast" etait suivi de
# "bash: line 202: apt-fast: command not found" dans le chroot.
export PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:$PATH"


# ── LES ETAPES SONT DANS iso_modules/ ──────────────────────────────────
# [2026-09-04] Ce script faisait 5 100 lignes d un seul tenant. Chaque etape
# vit maintenant dans iso_modules/NN-<slug>.sh, source ICI, dans l ordre des
# numeros, dans CE shell : les variables, les fonctions et le trap restent
# globaux, exactement comme avant le decoupage. Un module qui n a rien a
# faire (une variante, une architecture) fait `return` en tete.
# ATTENTION : `. fichier` rend le statut de la derniere commande du fichier,
# et set -e est actif. Chaque module se termine donc par `true` - sans quoi
# un module finissant par un test faux ("[ ... ] && { ... }", le test root de
# 00-options quand on EST root) arretait le script en silence, statut 1.
MODDIR="$DIR/iso_modules"
[ -d "$MODDIR" ] || { echo -e "${RED}iso_modules/ introuvable a cote de $0${NC}" >&2; exit 2; }
shopt -s nullglob; _ISO_MODS=("$MODDIR"/[0-9][0-9]-*.sh); shopt -u nullglob
```

```
[ "${#_ISO_MODS[@]}" -gt 0 ] || { echo -e "${RED}iso_modules/ est vide${NC}" >&2; exit 2; }
for _m in "${_ISO_MODS[@]}"; do . "$_m"; done
```

3.8 /opt/GSM/osmo-operator/build.sh

21795 octets, 385 lignes → 385 lignes

```
#!/bin/bash
# ============================================================================
#  build.sh - prepare l hote et construit l image osmocom-nitb
# ============================================================================
#
#  [2026-09-03] TROIS CHANGEMENTS, LIES :
#
#  1. apt-fast partout. packaging/apt-fast-install.sh pose apt-fast (aria2) et
#     les reglages apt communs, sur l hote comme dans les images et le rootfs
#     de l ISO. Un seul installeur, une seule facon de parler a apt.
#
#  2. docker compose v2 (+ buildx) est une DEPENDANCE, pas une option : il est
#     installe avec docker, et c est lui qui construit les images
#     (compose.yaml : services nitb, run, lite, stp). `docker build` reste le
#     repli si compose manque.
#
#  3. LE CACHE .deb SUR L HOTE. Chaque dossier compile dans l image sort en
#     paquet (.deb) - packaging/osmo-deb.sh. Ces paquets vivent dans
#         /var/cache/osmo-debs          (OSMO_DEB_CACHE pour deplacer)
#     Avant le build, on les recopie dans .deb-cache/ (le contexte docker) ;
#     l image les pose par dpkg au lieu de recompiler. Apres le build, on
#     ramene ce que l image a produit dans le cache de l hote. Le premier
#     build compile tout ; les suivants ne compilent que ce qui manque.
#     --no-cache : OSMO_DEB_REFRESH=1, tout est recompile et le cache reecrit.
#
#  Usage : sudo ./build.sh [--no-cache] [--lite] [--stp]
# ============================================================================
set -euo pipefail

# Contexte de build : le repertoire de CE script, pas le cwd de l appelant.
# Le lanceur .desktop passe par pkexec, qui demarre dans /root : sans ce cd,
# "docker build ." cherche /root/Dockerfile et echoue.
cd "$(dirname "$(readlink -f "$0")")"
DIR="$(pwd)"
# /usr/local/sbin DANS le PATH : apt-fast-install pose apt-fast la (comme le
# Dockerfile). Un root dont le PATH ne le contient pas (shell perso, sans sudo)
# voyait "[apt-fast] pret : /usr/local/sbin/apt-fast" puis "apt-fast: command
# not found" a la ligne suivante.
export PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:$PATH"

RED='\033[0;31m'; GREEN='\033[0;32m'; YELLOW='\033[0;33m'; CYAN='\033[0;36m'; NC='\033[0m'

# 0. Parsing des arguments
NO_CACHE=""
LITE=0
STP=0
OSMO_ARCH="$(dpkg --print-architecture 2>/dev/null || uname -m)"
for arg in "$@"; do
    case "$arg" in
        --no-cache) NO_CACHE="--no-cache" ;;
        --lite)     LITE=1 ;;
        --stp)      STP=1 ;;
        --arch=*)   OSMO_ARCH="${arg#*=}" ;;
        --arm)      OSMO_ARCH=arm64 ;;
        -h|--help)
            echo "Usage: sudo $0 [--no-cache] [--lite] [--stp] [--arch=arm64|--arm]"
            echo "  --arch=arm64 Construit les images pour arm64 (Raspberry Pi 4) sur cet hote,"
```

```
            echo "               par docker buildx --platform linux/arm64 sous qemu-user-static."
            echo "               Images taguees :arm64 (osmocom-nitb:arm64...). LENT au premier"
            echo "               build (emulation) ; le cache .deb rend les suivants courts."
            echo "               Pose aussi sur l hote tout ce que la chaine arm64 demande :"
            echo "               qemu-user-static, binfmt-support, docker-buildx, debootstrap,"
            echo "               mtools, dosfstools, e2fsprogs, kpartx... (build-iso.sh --arm,"
            echo "               armbian/build-armbian.sh)."
            echo "  --no-cache   Reconstruction complete : cache docker ignore, paquets .deb recompiles"
            echo "               et le cache /var/cache/osmo-debs reecrit."
            echo "  --lite       Construit EN PLUS osmocom-nitb:lite : la meme pile, sans les"
            echo "               arbres de sources de /opt/GSM (~4 Go sur 11). Voir Dockerfile.lite."
            echo "  --stp        Construit EN PLUS osmocom-stp (Dockerfile.stp), le hub SS7 seul."
            echo ""
            echo "  OSMO_DEB_CACHE=/chemin   cache des .deb compiles (defaut /var/cache/osmo-debs)"
            exit 0
            ;;
        *) echo -e "${YELLOW}[WARN] Argument inconnu ignore : $arg${NC}" ;;
    esac
done

# 1. Verification des privileges ROOT
if [[ $EUID -ne 0 ]]; then
   echo -e "${RED}[ERREUR] Ce script doit etre lance en tant que root (sudo).${NC}"
   exit 1
fi

echo "--- Preparation complete de l'hote (SDR & Docker) ---"
export DEBIAN_FRONTEND=noninteractive

# 2. apt-fast sur l hote : le meme installeur que dans les images
echo "[*] apt-fast (packaging/apt-fast-install.sh)..."
bash "$DIR/packaging/apt-fast-install.sh" || echo -e "${YELLOW}[WARN] apt-fast non installe - apt-get${NC}"
command -v apt-fast >/dev/null 2>&1 || apt-fast() { apt-get "$@"; }

# ── Le miroir Ubuntu : mesure, pas suppose - pour l image ET pour l hote ────
# archive.ubuntu.com peut repondre a 3 Ko/s ou pas du tout depuis ici (vu le
# 2026-09-04 : dix minutes muettes a l etape 3/64, et un apt de l hote qui ne
# rend jamais la main). packaging/apt-mirror.sh mesure quelques miroirs :
#   - le plus rapide part dans l image en build-arg (compose le lit dans
#     l environnement) ; OSMO_UBUNTU_MIRROR=http://... force un miroir ;
#   - si archive.ubuntu.com ne repond PAS et que les sources apt de l hote le
#     designent, elles sont basculees sur le plus rapide, avec une sauvegarde
#     (*.osmo-bak) - sinon aucun paquet hote (qemu-user-static, docker...) ne
#     s installe. OSMO_HOST_MIRROR=0 interdit de toucher aux sources de l hote,
#     OSMO_HOST_MIRROR=force les bascule meme si archive repond (lentement).
echo "[*] Miroir Ubuntu (packaging/apt-mirror.sh)..."
_MLIST="$(bash "$DIR/packaging/apt-mirror.sh" --list noble 2>&1 | tee /dev/stderr | grep -E '^[0-9]+' || true)"
OSMO_UBUNTU_MIRROR="${OSMO_UBUNTU_MIRROR:-$(printf '%s\n' "$_MLIST" | head -1 | cut -f2)}"
OSMO_UBUNTU_MIRROR="${OSMO_UBUNTU_MIRROR:-http://archive.ubuntu.com/ubuntu}"
export OSMO_UBUNTU_MIRROR
echo -e "${GREEN}[OK] miroir : ${OSMO_UBUNTU_MIRROR}${NC}"
if { [ "${OSMO_HOST_MIRROR:-1}" = "force" ] || { [ "${OSMO_HOST_MIRROR:-1}" = "1" ] && ! printf '%s\n' "$_MLIST" |
 ↪  grep -q 'archive\.ubuntu\.com'; }; } \
   && [ "$OSMO_UBUNTU_MIRROR" != "http://archive.ubuntu.com/ubuntu" ]; then
    _switched=""
    for _f in /etc/apt/sources.list /etc/apt/sources.list.d/*.list /etc/apt/sources.list.d/*.sources; do
        [ -f "$_f" ] || continue
        grep -qE '^(deb|deb-src|URIs:).*https?://(archive|[a-z]{2}\.archive|security)\.ubuntu\.com/ubuntu' "$_f"
         ↪  || continue
        cp -f "$_f" "$_f.osmo-bak"
        sed -i -E "s#https?://(archive|[a-z]{2}\.archive|security)\.ubuntu\.com/ubuntu/?#${OSMO_UBUNTU_MIRROR}#g"
         ↪  "$_f"
        _switched="$_switched $_f"
    done
    if [ -n "$_switched" ]; then
```

```
        echo -e "${YELLOW}[WARN] archive.ubuntu.com ne repond pas : sources apt de l hote basculees sur
        ↪  ${OSMO_UBUNTU_MIRROR}${NC}"
        echo -e "${YELLOW}       (sauvegarde en *.osmo-bak :${_switched} ; OSMO_HOST_MIRROR=0 pour ne pas y
        ↪  toucher)${NC}"
    fi
fi
apt-fast update -qq || true

# 3. Docker, docker compose v2 et buildx : TOUJOURS dans les dependances.
#    (Avant, compose n etait pose qu avec docker, quand docker manquait : un
#    hote qui avait deja docker.io ne l obtenait jamais.)
#
# [2026-09-13] MAIS JAMAIS docker.io SI DOCKER EST DEJA LA. Sur un runner
# GitHub - et sur tout hote qui suit le depot Docker - docker vient de
# docker-ce + containerd.io ; docker.io entre en conflit avec les deux, apt
# repond « pkgProblemResolver::Resolve generated breaks », les DEUX lignes de
# repli contenaient docker.io, et le job mourait sur « docker /
# docker-compose-v2 non installables » alors que docker ET compose v2
# marchaient parfaitement. iso_modules/20-hote.sh a toujours eu cette garde
# (iso_host_docker, « UNIQUEMENT SI ABSENT ») ; build.sh l avait perdue en
# rendant compose inconditionnel. On ne complete donc que ce qui MANQUE, et
# on accepte les deux nommages : docker-compose-v2 / docker-buildx (Ubuntu)
# et docker-compose-plugin / docker-buildx-plugin (depot Docker).
# OSMO_HOST_DOCKER=0 : ne RIEN demander a apt pour docker. Un runner GitHub
# arrive avec docker, compose v2 et buildx tout faits (paquets moby-*) ; lui
# laisser tenter quoi que ce soit, c est parier sur un nommage de paquets qu on
# ne maitrise pas, pour un gain nul. Le workflow pose donc cette variable.
echo "[*] Docker + docker compose v2 + buildx..."
if [ "${OSMO_HOST_DOCKER:-1}" = 0 ]; then
    echo -e "${GREEN}[OK] docker : installation sautee (OSMO_HOST_DOCKER=0) - $(docker --version 2>/dev/null ||
    ↪  echo 'docker introuvable, la suite dira si c est un probleme')${NC}"
elif command -v docker >/dev/null 2>&1; then
    echo -e "${GREEN}[OK] docker deja installe ($(docker --version 2>/dev/null || echo version inconnue)) - pas
    ↪  touche${NC}"
    docker info >/dev/null 2>&1 || systemctl start docker >/dev/null 2>&1 || true
    docker compose version >/dev/null 2>&1 \
        || apt-fast install -y --no-install-recommends docker-compose-v2 \
        || apt-fast install -y --no-install-recommends docker-compose-plugin \
        || echo -e "${YELLOW}[WARN] docker present mais pas de compose v2 : compose.sh basculera sur docker
        ↪  build${NC}"
    docker buildx version >/dev/null 2>&1 \
        || apt-fast install -y --no-install-recommends docker-buildx \
        || apt-fast install -y --no-install-recommends docker-buildx-plugin \
        || echo -e "${YELLOW}[WARN] buildx absent : --arch=arm64 ne marchera pas sur cet hote${NC}"
else
    apt-fast install -y --no-install-recommends docker.io docker-compose-v2 docker-buildx \
        || apt-fast install -y --no-install-recommends docker.io docker-compose-v2 \
        || apt-fast install -y --no-install-recommends docker.io \
        || command -v docker >/dev/null 2>&1 \
        || { echo -e "${RED}[ERREUR] docker / docker-compose-v2 non installables.${NC}"; \
             echo -e "${RED}        (docker absent du PATH : $PATH)${NC}"; \
            echo -e "${RED}       OSMO_HOST_DOCKER=0 pour sauter cette etape si docker est pose autrement.${NC}";
              ↪  exit 1; }
fi
systemctl enable --now docker >/dev/null 2>&1 || true
HAVE_COMPOSE=0
docker compose version >/dev/null 2>&1 && HAVE_COMPOSE=1
# L affichage BuildKit reste celui par defaut (tty, en bleu) : il replie la
# sortie de chaque RUN sur quelques lignes. Pour lire TOUT ce qu une etape
# ecrit (une compilation d une heure, un apt muet a l etape 3/64) :
#     BUILDKIT_PROGRESS=plain ./build.sh
# Vaut pour docker compose build ET docker build (et buildx).
export BUILDKIT_PROGRESS="${BUILDKIT_PROGRESS:-auto}"

# ── Architecture cible : la meme que l hote, ou une autre par buildx ──────────
# --arch=arm64 sur un hote x86 : docker buildx build --platform linux/arm64,
```

```
# qemu-user-static emulant l aarch64 dans les conteneurs de construction. Il
# faut binfmt_misc avec le drapeau F (fix-binary) - sans lui le conteneur ne
# trouve pas l emulateur. Les images sortent taguees :arm64 pour cohabiter avec
# les images natives. Le cache .deb est deja par architecture (osmo-deb nomme
# les paquets ..._arm64.deb).
HOST_ARCH="$(dpkg --print-architecture)"
BUILD_PLATFORM=""; IMG_TAG=""
if [ "$OSMO_ARCH" != "$HOST_ARCH" ]; then
    case "$OSMO_ARCH" in
        arm64) BUILD_PLATFORM=linux/arm64 ;;
        amd64) BUILD_PLATFORM=linux/amd64 ;;
        *) echo -e "${RED}[ERREUR] --arch : amd64 ou arm64 (recu : $OSMO_ARCH)${NC}"; exit 2 ;;
    esac
    IMG_TAG=":$OSMO_ARCH"
    # [2026-10-20] TOUT ce que la chaine --arm demande a l hote, pose ici une fois pour toutes :
    # l emulation (qemu-user-static, binfmt-support) et buildx pour cette image Docker ; puis les
    # outils de build-iso.sh --arm (iso_modules/20-hote.sh, ISO_HOST_PKGS_COMMON : debootstrap,
    # mke2fs -d, mcopy, sfdisk, kpartx, zstd, python3-pil...) et de armbian/build-armbian.sh (git,
    # docker, qemu-user pour le chroot de customize-image.sh). Avant, build-iso.sh les posait a son
    # etape 0 et build-armbian.sh se contentait de les reclamer : un hote qui n avait fait que
    # ./build.sh --arm partait sans mtools ni dosfstools.
    echo "[*] Paquets hote de la chaine ${OSMO_ARCH} : emulation, buildx, outils d image SD..."
    apt-fast install -y --no-install-recommends qemu-user-static binfmt-support docker-buildx \
        git debootstrap rsync wget curl ca-certificates gnupg zstd xz-utils cpio file gcc make \
        python3 python3-pil python3-yaml e2fsprogs fdisk util-linux dosfstools mtools kpartx \
        || apt-fast install -y --no-install-recommends qemu-user-static binfmt-support \
        || echo -e "${YELLOW}[WARN] qemu-user-static non installe (apt) - binfmt via docker ci-dessous${NC}"
    _bf="/proc/sys/fs/binfmt_misc/qemu-${OSMO_ARCH/arm64/aarch64}"
    _bf="${_bf/amd64/x86_64}"
    [ -e "$_bf" ] || update-binfmts --enable "$(basename "$_bf")" >/dev/null 2>&1 || true
    if ! { [ -e "$_bf" ] && grep -q '^flags:.*F' "$_bf"; }; then
        echo "[*] binfmt $(basename "$_bf") sans drapeau F : enregistrement via tonistiigi/binfmt..."
        docker run --privileged --rm tonistiigi/binfmt --install "${OSMO_ARCH}" >/dev/null 2>&1 || true
    fi
    if [ -e "$_bf" ] && grep -q '^flags:.*F' "$_bf"; then
        echo -e "${GREEN}[OK] emulation ${OSMO_ARCH} (binfmt fix-binary) - images taguees ${IMG_TAG}${NC}"
    else
        echo -e "${RED}[ERREUR] binfmt ${OSMO_ARCH} indisponible (qemu-user-static, binfmt-support)${NC}"; exit 1
    fi
    docker buildx version >/dev/null 2>&1 || { echo -e "${RED}[ERREUR] docker buildx requis pour --arch${NC}";
     ↪  exit 1; }
fi
IMG_NITB="osmocom-nitb${IMG_TAG}"
IMG_LITE="osmocom-nitb:lite${IMG_TAG:+-$OSMO_ARCH}"
IMG_STP="osmocom-stp${IMG_TAG}"
[ "$HAVE_COMPOSE" = 1 ] && echo -e "${GREEN}[OK] $(docker compose version)${NC}" \
    || echo -e "${YELLOW}[WARN] docker compose absent - repli sur docker build${NC}"

# 4. Installation des dependances critiques sur l'hote
# SCTP est vital pour les protocoles de signalisation Osmocom
echo "[*] Installation de SCTP, TUN et D-Bus sur l'hote..."
apt-fast install -y lksctp-tools libsctp-dev dbus uml-utilities libusb-1.0-0-dev \
     wireshark linphone* gnome-terminal whiptail expect netcat-openbsd \
     telnet iproute2 pulseaudio-utils dpkg-dev zstd

# 4b. toast : le codec GSM 06.10 (quut.com), absent du paquet apt
# libgsm1 (apt) fournit la LIB mais pas le binaire toast/untoast/tcat. On le
# compile depuis les sources officielles, SOUS /opt/GSM, et on pose le binaire
# dans /usr/local/bin. Non fatal : sans reseau on continue le build.
echo "[*] Compilation de toast (codec GSM 06.10)..."
_GSM_VER=gsm-1.0.24
_GSM_DIR=gsm-1.0-pl24   # le tarball se decompresse SOUS ce nom
_GSM_URL="https://www.quut.com/gsm/${_GSM_VER}.tar.gz"
if command -v toast >/dev/null 2>&1; then
    echo -e "${GREEN}[OK] toast deja present ($(command -v toast)).${NC}"
else
```

```
    apt-fast install -y --no-install-recommends build-essential wget >/dev/null 2>&1 || true
    mkdir -p /opt/GSM
    if ( cd /opt/GSM \
         && wget -q0 "${_GSM_VER}.tar.gz" "$_GSM_URL" \
         && tar xzf "${_GSM_VER}.tar.gz" \
         && cd "$_GSM_DIR" \
         && { make >/dev/null 2>&1 || true; } \
         && { [ -x bin/toast ] || make toast >/dev/null 2>&1 || true; } \
         && [ -x bin/toast ] ); then
        for _b in toast untoast tcat; do
            [ -e "/opt/GSM/${_GSM_DIR}/bin/$_b" ] \
                && install -m755 "/opt/GSM/${_GSM_DIR}/bin/$_b" "/usr/local/bin/$_b"
        done
        echo -e "${GREEN}[OK] toast installe dans /usr/local/bin (sources : /opt/GSM/${_GSM_DIR}).${NC}"
    else
        echo -e "${YELLOW}[WARN] compilation de toast echouee (reseau ou outils ?) - on continue.${NC}"
    fi
fi

# 5. Chargement des modules noyau
echo "[*] Chargement des modules noyau (SCTP & TUN)..."
modprobe sctp
modprobe tun

# Verification du module SCTP (lecture directe de /proc/modules : pas de tube,
# donc pas de SIGPIPE/pipefail, et ancrage sur le nom de module exact)
if grep -q '^sctp ' /proc/modules; then
    echo -e "${GREEN}[OK] Module SCTP charge sur l'hote.${NC}"
else
    echo -e "${RED}[ERREUR] Impossible de charger SCTP.${NC}"
fi

# ============================================================================
# 6. Le cache .deb : de l hote vers le contexte de build
# ============================================================================
OSMO_DEB_CACHE="${OSMO_DEB_CACHE:-/var/cache/osmo-debs}"
mkdir -p "$OSMO_DEB_CACHE" "$DIR/.deb-cache"
find "$DIR/.deb-cache" -maxdepth 1 -name '*.deb' -delete
OSMO_DEB_REFRESH=0
if [[ -n "$NO_CACHE" ]]; then
    OSMO_DEB_REFRESH=1
    echo "[*] Mode --no-cache actif : reconstruction complete, paquets .deb recompiles."
else
    cp -f "$OSMO_DEB_CACHE"/osmo-build-*.deb "$DIR/.deb-cache/" 2>/dev/null || true
    # find, pas "ls | wc" : sous pipefail, un ls sans resultat sortait en 2 et
    # set -e arretait le script ici, sans un mot, des le premier build.
    _n=$(find "$DIR/.deb-cache" -maxdepth 1 -name '*.deb' | wc -l)
    if [ "$_n" -gt 0 ]; then
        echo -e "${GREEN}[OK] cache .deb : ${_n} paquets repris de ${OSMO_DEB_CACHE} ($(du -sh "$DIR/.deb-cache" |
        ↪  cut -f1))${NC}"
    else
        echo "[*] cache .deb vide (${OSMO_DEB_CACHE}) : premier build, tout sera compile et mis en cache."
    fi
fi
export OSMO_DEB_REFRESH

# Construit un service de compose.yaml (ou son equivalent docker build).
#   $1 = service (nitb|run|lite|stp)   $2 = image   $3 = Dockerfile   $4.. = build-args
compose_build() {
    local svc="$1" image="$2" dockerfile="$3"; shift 3
    # Dockerfile (nitb) ne se termine PLUS par l etape osmocom-nitb : l etape
    # `debs` la suit, qui n exporte que le cache .deb pour la CI. Sans --target
    # docker batirait celle-la. Les autres Dockerfile n ont qu une cible utile,
    # qui est deja la derniere - rien a leur passer. compose.yaml porte le meme
    # target sur son service nitb, pour la branche compose ci-dessous.
    local target=()
```

```
    [ "$svc" = nitb ] && target=(--target osmocom-nitb)
    if [ -n "$BUILD_PLATFORM" ]; then
        # Autre architecture : buildx --platform, --load pour que l image soit
        # dans le docker local (et non seulement dans le cache de buildx).
        local args=(); local a; for a in "$@"; do args+=(--build-arg "$a"); done
        docker buildx build --platform "$BUILD_PLATFORM" --load $NO_CACHE \
            "${target[@]+"${target[@]}"}" \
            "${args[@]+"${args[@]}"}" -f "$DIR/$dockerfile" -t "$image" "$DIR"
    elif [ "$HAVE_COMPOSE" = 1 ]; then
        docker compose -f "$DIR/compose.yaml" build $NO_CACHE "$svc"
    else
        local args=(); local a; for a in "$@"; do args+=(--build-arg "$a"); done
        docker build $NO_CACHE "${target[@]+"${target[@]}"}" \
            "${args[@]+"${args[@]}"}" -f "$DIR/$dockerfile" -t "$image" "$DIR"
    fi
}

# Ramene dans le cache de l hote les paquets que l image vient de produire.
#   $1 = image
debs_from_image() {
    local image="$1" cid n
    cid="$(docker create "$image" /bin/true 2>/dev/null)" || return 0
    mkdir -p "$DIR/.deb-cache/out"
    docker cp "$cid:/var/cache/osmo-debs/." "$DIR/.deb-cache/out/" 2>/dev/null || true
    docker rm "$cid" >/dev/null 2>&1 || true
    n=$(find "$DIR/.deb-cache/out" -maxdepth 1 -name 'osmo-build-*.deb' | wc -l)
    if [ "$n" -gt 0 ]; then
        cp -f "$DIR/.deb-cache/out"/osmo-build-*.deb "$OSMO_DEB_CACHE/"
        echo -e "${GREEN}[OK] cache .deb : ${n} paquets dans ${OSMO_DEB_CACHE} ($(du -sh "$OSMO_DEB_CACHE" | cut
        ↪  -f1))${NC}"
    fi
    rm -rf "$DIR/.deb-cache/out"
}

# 7. Lancement du build Docker
echo "--- Lancement du build de l'image ${IMG_NITB} ---"
if compose_build nitb "$IMG_NITB" Dockerfile "OSMO_DEB_REFRESH=$OSMO_DEB_REFRESH"
↪  "OSMO_UBUNTU_MIRROR=$OSMO_UBUNTU_MIRROR"; then
    echo -e "${GREEN}[OK] Image ${IMG_NITB} construite avec succes.${NC}"
    debs_from_image "$IMG_NITB"
else
    echo -e "${RED}[ERREUR] Le build Docker a echoue.${NC}"
    exit 1
fi

# 8. Image lite : la meme pile, sans les arbres de sources
# osmocom-nitb embarque /opt/GSM - les arbres de construction de gnuradio,
# gr-gsm, qemu, libosmocore... 4 Go sur 11, qui ont servi a COMPILER et ne
# servent plus a rien ensuite : ce qui tourne vit dans /usr/local.
# Dockerfile.lite les retire et aplatit le resultat (voir son entete : effacer
# dans une couche de plus ne rend aucun espace).
if [[ "$LITE" -eq 1 ]]; then
    echo "--- Lancement du build de l'image ${IMG_LITE} ---"
    if LITE_BASE="$IMG_NITB" LITE_IMAGE="$IMG_LITE" \
       compose_build lite "$IMG_LITE" Dockerfile.lite "BASE=$IMG_NITB"; then
        echo -e "${GREEN}[OK] Image ${IMG_LITE} construite.${NC}"
        docker images --format '  {{.Repository}}:{{.Tag}}\t{{.Size}}' \
            | grep -E '^\s+osmocom-nitb:(latest|lite)' || true
    else
        echo -e "${RED}[ERREUR] Le build de l'image lite a echoue.${NC}"
        exit 1
    fi
fi

# 9. Image STP : le hub SS7 seul (Dockerfile.stp), avec le meme cache .deb
if [[ "$STP" -eq 1 ]]; then
```

```
    echo "--- Lancement du build de l'image ${IMG_STP} ---"
    if compose_build stp "$IMG_STP" Dockerfile.stp "OSMO_DEB_REFRESH=$OSMO_DEB_REFRESH"
     ↪  "OSMO_UBUNTU_MIRROR=$OSMO_UBUNTU_MIRROR"; then
        echo -e "${GREEN}[OK] Image ${IMG_STP} construite.${NC}"
        debs_from_image "$IMG_STP"
    else
        echo -e "${RED}[ERREUR] Le build de l'image stp a echoue.${NC}"
        exit 1
    fi
fi

# Le contexte ne garde pas les paquets : ils sont dans le cache de l hote.
find "$DIR/.deb-cache" -maxdepth 1 -name '*.deb' -delete
```

3.9 /opt/GSM/osmo-operator/compose.sh

10072 octets, 211 lignes → 211 lignes

```
#!/bin/bash
# ============================================================================
#  compose.sh - le banc en docker compose v2 : images, hub + un operateur
# ============================================================================
#
#    sudo ./compose.sh build [--no-cache] [--lite] [--stp]   les images (= build.sh)
#    sudo ./compose.sh up [--ms N] [--phy faketrx|virtphy|qemu] [--no-run]
#    sudo ./compose.sh down          arrete et retire hub + operateur (configs gardees)
#    sudo ./compose.sh ps | logs [service] | shell
#    sudo ./compose.sh debs          le cache .deb de l hote (/var/cache/osmo-debs)
#
#  `up` fait ce que start-docker.sh fait pour UN operateur, avec compose.yaml comme
#  description du banc :
#    1. les configs (apply_config_templates de generate_configs.sh, la config
#       du hub par helpers/create_interop.sh) dans data/compose/ ;
#    2. data/compose/.env : les adresses calculees par les memes fonctions que
#       start-docker.sh (op_backbone_ip, op_private_*), les choix (N_MS, PHY_MODE) ;
#    3. data/compose/compose.audio.yaml : le socket PulseAudio de l hote et
#       /dev/snd, quand ils existent (compose ne sait pas les rendre optionnels) ;
#    4. docker compose --profile bench up -d ;
#    5. run.sh dans le conteneur, attente du HLR, abonnes de test dans le HLR
#       (memes IMSI / MSISDN <noeud>00<op><ms> / Ki que start-docker.sh).
#  Pour N operateurs, le WAN, les ports publies : start-docker.sh.
# ============================================================================
set -euo pipefail

DIR="$(cd "$(dirname "$(readlink -f "$0")")" && pwd)"
cd "$DIR"
RED='\033[0;31m'; GREEN='\033[0;32m'; YELLOW='\033[1;33m'; CYAN='\033[0;36m'; NC='\033[0m'

CDIR="${OSMO_COMPOSE_DIR:-$DIR/data/compose}"
FILES=(-f "$DIR/compose.yaml")

usage() { sed -n '2,24p' "$0" | sed 's/^#\{1,2\} \{0,1\}//'; exit "${1:-0}"; }
die() { echo -e "${RED}$*${NC}" >&2; exit 1; }

need_root() { [ "$(id -u)" -eq 0 ] || die "Root requis (sudo)."; }
need_compose() {
    docker compose version >/dev/null 2>&1 \
        || die "docker compose v2 absent : sudo ./build.sh l installe (docker-compose-v2 + docker-buildx)."
}

# La bibliotheque de start-docker.sh : ses fonctions (op_*, apply_config_templates...)
# sans son execution - meme decoupe que build-iso.sh (load_start_lib).
load_start_lib() {
    local lib; lib="$(mktemp)"
```

```
    awk '
        /^banner[[:space:]]*$/                    { exit }
        /^\[ "\$\{1:-\}" = "stop" \]/             { exit }
        /^\[ "\$\(id -u\)" -ne 0 \]/              { exit }
        /^choose_network_mode[[:space:]]*$/       { exit }
        /^\.\//                                   { exit }
        /^case "\$NETWORK_MODE" in[[:space:]]*$/  { exit }
        { print }
    ' "$DIR/launch/start-docker.sh" > "$lib"
    export OSMO_REPO_DIR="$DIR"
    # shellcheck disable=SC1090
    source "$lib"
    rm -f "$lib"
}

compose() {
    local envf=()
    [ -f "$CDIR/.env" ] && envf=(--env-file "$CDIR/.env")
    [ -f "$CDIR/compose.audio.yaml" ] && FILES+=(-f "$CDIR/compose.audio.yaml")
    docker compose "${FILES[@]}" "${envf[@]+"${envf[@]}"}" "$@"
}

cmd_build() { need_root; exec bash "$DIR/build.sh" "$@"; }

cmd_up() {
    need_root; need_compose
    local n_ms=2 phy="${PHY_MODE:-faketrx}" do_run=1
    while [ $# -gt 0 ]; do
        case "$1" in
            --ms)     n_ms="${2:?}"; shift ;;
            --ms=*)   n_ms="${1#*=}" ;;
            --phy)    phy="${2:?}"; shift ;;
            --phy=*)  phy="${1#*=}" ;;
            --no-run) do_run=0 ;;
            -h|--help) usage 0 ;;
            *) die "option inconnue : $1" ;;
        esac; shift
    done
    [[ "$n_ms" =~ ^[1-9][0-9]?$ ]] || die "--ms : 1 a 99"

    for img in "${IMAGE_RUN:-osmocom-run}" "${IMAGE_STP:-osmocom-stp}"; do
        docker image inspect "$img" >/dev/null 2>&1 \
            || die "image $img absente : sudo ./compose.sh build --stp (ou ./build.sh --stp, puis
            ↪  ./launch/start-docker.sh une fois pour osmocom-run)"
    done

    load_start_lib
    export OSMO_PRIV_BY_OP=1      # on provisionne PAR operateur (voir _osmo_priv_index)
    local op=1 node="${WAN_NODE_ID:-1}"
    local bb_ip priv_net priv_gw priv_ip mcc mnc name
    bb_ip="$(op_backbone_ip "$op")"
    priv_net="$(op_private_net "$op")"; priv_gw="$(op_private_gw "$op")"; priv_ip="$(op_private_ip "$op")"
    mcc="$(op_mcc "$op")"; mnc="$(op_mnc "$op")"; name="$(op_default_name "$op")"

    echo -e "${CYAN}── Banc compose : hub 172.20.0.10 + operateur ${op} (${name}, MCC ${mcc} MNC ${mnc}, ${n_ms}
    ↪  MS, PHY ${phy}) ──${NC}"
    echo -e "  backbone ${CYAN}${bb_ip}${NC}   prive ${CYAN}${priv_ip}${NC} (${priv_net})"

    # 1. Les configs
    rm -rf "$CDIR/op1" "$CDIR/hub"
    mkdir -p "$CDIR/op1/osmocom" "$CDIR/op1/asterisk" "$CDIR/op1/bb" "$CDIR/hub"
    apply_config_templates "$CDIR/op1" "$priv_ip" "$priv_gw" "$op" \
        "1.${op}.1" "1.${op}.2" "1.${op}.3" "$mcc" "$mnc" "$name" \
        172.20.0.10 "no shutdown" 1 \
        || die "generation des configs echouee"
    bash "$DIR/helpers/create_interop.sh" 1 "$CDIR/hub/osmo-stp-interop.cfg" >/dev/null \
```

```
        || die "generation de la config du hub echouee"
    for f in extensions.conf pjsip.conf rtp.conf modules.conf annuaire.conf; do
        [ -f "$CDIR/op1/asterisk/$f" ] || { [ -f "$DIR/configs/$f" ] && cp "$DIR/configs/$f"
         ↪  "$CDIR/op1/asterisk/$f" || : > "$CDIR/op1/asterisk/$f"; }
    done
    [ -f "$CDIR/op1/bb/mobile.cfg" ] || cp "$DIR/configs/mobile.cfg" "$CDIR/op1/bb/mobile.cfg"
    cp -f "$DIR/configs/asound.conf" "$CDIR/asound.conf"
    chmod +x "$CDIR/op1/osmocom"/*.sh 2>/dev/null || true
    echo -e "  ${GREEN}✓${NC} configs dans ${CDIR}"

    # 2. .env : ce que compose.yaml lit
    local host_ip; host_ip="$(ip -4 route get 1.1.1.1 2>/dev/null | awk '{for(i=1;i<=NF;i++) if($i=="src") print
     ↪  $(i+1)}' | head -1)"
    cat > "$CDIR/.env" <<ENV
# genere par compose.sh up - ne pas editer : relancer compose.sh up
IMAGE_RUN=${IMAGE_RUN:-osmocom-run}
IMAGE_STP=${IMAGE_STP:-osmocom-stp}
OSMO_COMPOSE_DIR=${CDIR}
OP1_BACKBONE_IP=${bb_ip}
OP1_PRIV_NET=${priv_net}
OP1_PRIV_GW=${priv_gw}
OP1_PRIV_IP=${priv_ip}
N_MS=${n_ms}
PHY_MODE=${phy}
HOST_IP=${host_ip}
WAN_NODE_ID=${node}
ENV

    # 3. L audio de l hote : socket pulse et /dev/snd quand ils existent
    local pa=""
    for s in /var/run/pulse/native /run/pulse/native "/run/user/${SUDO_UID:-$(id -u)}/pulse/native"; do
        [ -S "$s" ] && { pa="$s"; break; }
    done
    {
        echo "# genere par compose.sh up : l audio de l hote, quand il existe"
        echo "services:"
        echo "  operator-1:"
        if [ -n "$pa" ]; then
            echo "    volumes:"; echo "      - ${pa}:/run/pulse/native"
            echo "    environment:"; echo "      PULSE_SERVER: unix:/run/pulse/native"
        fi
        if [ -d /dev/snd ]; then
            echo "    devices:"; echo "      - /dev/snd:/dev/snd"
            getent group audio >/dev/null 2>&1 && { echo "    group_add:"; echo "      - \"$(getent group audio |
             ↪  cut -d: -f3)\""; }
        fi
    } > "$CDIR/compose.audio.yaml"
    [ -n "$pa" ] && echo -e "  ${GREEN}✓${NC} audio : socket pulse ${pa}" \
                 || echo -e "  ${YELLOW}audio : pas de socket pulse sur l hote - repli TCP 172.20.0.1:4713${NC}"

    # 4. Les conteneurs
    modprobe sctp 2>/dev/null || true; modprobe tun 2>/dev/null || true
    mkdir -p /tmp/osmocom-logs/op1
    compose --profile bench up -d --remove-orphans
    echo -e "  ${GREEN}✓${NC} osmo-inter-stp et osmo-operator-1 lances"
    [ "$do_run" = 1 ] || { echo "  (--no-run : run.sh non lance - docker exec -ti osmo-operator-1 bash)"; return
     ↪  0; }

    # 5. La pile, puis les abonnes
    echo -e "  ${GREEN}[*] run.sh dans osmo-operator-1...${NC}"
    docker exec -d osmo-operator-1 bash -c \
        "mkdir -p /var/log/osmocom && { RUN_NO_PROCESS=0 CALYPSO_NO_ATTACH=1 /etc/osmocom/run.sh; } >
         ↪  /var/log/osmocom/run.sh.log 2>&1"
    echo -ne "  ${GREEN}[*] Attente HLR (4258)${NC}"
    local retry=0
    while ! docker exec osmo-operator-1 bash -c "echo >/dev/tcp/127.0.0.1/4258" 2>/dev/null; do
```

```
        sleep 2; echo -n "."; retry=$((retry + 1))
        [ $retry -ge 45 ] && { echo -e " ${RED}TIMEOUT${NC}"; break; }
    done
    echo -e " ${GREEN}✓${NC}"
    local ms imsi msisdn ki feed
    feed="$(mktemp)"
    echo "enable" > "$feed"
    for ms in $(seq 1 "$n_ms"); do
        imsi="${mcc}${mnc}$(printf '%04d%06d' "$op" "$ms")"
        msisdn="$(osmo_msisdn "$node" "$op" "$ms")"
        ki="$(printf '00112233445566778899aabbccdd%02x%02x' "$ms" "$op")"
        printf 'subscriber imsi %s create\nsubscriber imsi %s update msisdn %s\nsubscriber imsi %s update aud2g
        ↪  comp128v1 ki %s\n' \
            "$imsi" "$imsi" "$msisdn" "$imsi" "$ki" >> "$feed"
        echo "    MS${ms} : IMSI ${imsi}  MSISDN ${msisdn}"
    done
    echo "end" >> "$feed"
    docker cp "$feed" osmo-operator-1:/tmp/hlr_feed.vty; rm -f "$feed"
    docker exec osmo-operator-1 bash -c '(sleep 1; cat /tmp/hlr_feed.vty; sleep 2) | nc -q2 127.0.0.1 4258
    ↪  >/dev/null 2>&1; rm -f /tmp/hlr_feed.vty' || true
    echo -e "  ${GREEN}✓${NC} HLR alimente (${n_ms} abonnes)"
    echo ""
    echo -e "${GREEN}Banc pret.${NC}  docker exec -ti osmo-operator-1 bash   |   ./compose.sh logs   |
    ↪  ./compose.sh down"
}

cmd_down()  { need_root; need_compose; compose --profile bench down --remove-orphans; }
cmd_ps()    { need_compose; compose --profile bench ps; }
cmd_logs()  { need_compose; compose --profile bench logs -f "$@"; }
cmd_shell() { exec docker exec -ti "${1:-osmo-operator-1}" bash; }
cmd_debs()  { ls -1sh "${OSMO_DEB_CACHE:-/var/cache/osmo-debs}"/osmo-build-*.deb 2>/dev/null || echo "cache vide
↪  (${OSMO_DEB_CACHE:-/var/cache/osmo-debs})"; }

case "${1:-}" in
    build) shift; cmd_build "$@" ;;
    up)    shift; cmd_up "$@" ;;
    down)  shift; cmd_down ;;
    ps)    cmd_ps ;;
    logs)  shift; cmd_logs "$@" ;;
    shell) shift; cmd_shell "$@" ;;
    debs)  cmd_debs ;;
    -h|--help|help|'') usage 0 ;;
    *) die "commande inconnue : $1 (build|up|down|ps|logs|shell|debs)" ;;
esac
```

3.10 /opt/GSM/osmo-operator/compose.yaml

6981 octets, 174 lignes → 174 lignes

```
# ============================================================================
#  compose.yaml - les images du banc et le banc a un operateur, en compose v2
# ============================================================================
#
#  DEUX PROFILS, DEUX USAGES.
#
#  build   : les quatre images, chacune par son Dockerfile. C est ce que
#            build.sh (et derriere lui build-iso.sh, start-docker.sh) appelle :
#                docker compose build nitb          Dockerfile      osmocom-nitb
#                docker compose build run           Dockerfile.run  osmocom-run
#                docker compose build lite          Dockerfile.lite osmocom-run:lite
#                docker compose build stp           Dockerfile.stp  osmocom-stp
#            Elles se nomment sur la ligne de commande : pas besoin d activer
#            le profil. `run` part de l image osmocom-nitb DEJA construite (ou
#            tiree du hub : docker pull bastienbaranoff/norf_gsm) - on ne la
```

```
#            declare pas en additional_context expres, sinon compose la
#            reconstruirait a chaque fois.
#
#  bench   : le banc minimal - le hub SS7 (osmo-inter-stp) et UN operateur
#            (osmo-operator-1), avec les memes capacites, reseaux, montages et
#            variables que le `docker run` de start-docker.sh. Les configs sont
#            GENEREES par ./compose.sh up (apply_config_templates de ce depot,
#            create_interop.sh) dans data/compose/ ; le fichier .env qui va
#            avec porte les adresses calculees. Pour N operateurs, le WAN, les
#            ports SIP publies : start-docker.sh reste l orchestrateur.
#                ./compose.sh up      ./compose.sh down     ./compose.sh ps
#
#  Variables (data/compose/.env, ecrit par compose.sh ; defauts ici) :
#    IMAGE_RUN IMAGE_STP        les images a lancer
#    OSMO_COMPOSE_DIR           ou sont les configs generees (./data/compose)
#    OP1_BACKBONE_IP            172.20.0.11   (op_backbone_ip 1)
#    OP1_PRIV_NET/GW/IP         192.168.2.0/24 .1 .10  (op_private_* 1)
#    N_MS PHY_MODE HOST_IP WAN_NODE_ID PULSE_SERVER
# ============================================================================
name: osmo-operator

services:
  # ── Images ──────────────────────────────────────────────────────────────
  nitb:
    image: osmocom-nitb
    build:
      context: .
      dockerfile: Dockerfile
      # OBLIGATOIRE : Dockerfile se termine par l'etape `debs` (le cache .deb
      # seul, pour la CI), qui serait la cible par defaut. Sans ce target,
      # `docker compose build nitb` sort une image vide.
      target: osmocom-nitb
      args:
        OSMO_DEB_REFRESH: ${OSMO_DEB_REFRESH:-0}
        OSMO_UBUNTU_MIRROR: ${OSMO_UBUNTU_MIRROR:-}
    profiles: [build]

  run:
    image: ${IMAGE_RUN:-osmocom-run}
    build:
      context: .
      dockerfile: Dockerfile.run
      args:
        QEMU_CACHE_BUST: ${QEMU_CACHE_BUST:-0}
        BASE: ${RUN_BASE:-osmocom-nitb}
    profiles: [build]

  lite:
    image: ${LITE_IMAGE:-osmocom-run:lite}
    build:
      context: .
      dockerfile: Dockerfile.lite
      args:
        BASE: ${LITE_BASE:-osmocom-run}
    profiles: [build]

  stp:
    image: ${IMAGE_STP:-osmocom-stp}
    build:
      context: .
      dockerfile: Dockerfile.stp
      args:
        BASE_IMAGE: ${STP_BASE_IMAGE:-ubuntu:24.04}
        HUB_NODES: ${HUB_NODES:-3}
        HUB_OPS: ${HUB_OPS:-1}
        OSMO_DEB_REFRESH: ${OSMO_DEB_REFRESH:-0}
```

```
      OSMO_UBUNTU_MIRROR: ${OSMO_UBUNTU_MIRROR:-}
  profiles: [build]

# ── Le banc : hub + operateur 1 ──────────────────────────────────────
inter-stp:
  image: ${IMAGE_STP:-osmocom-stp}
  container_name: osmo-inter-stp
  hostname: osmo-inter-stp
  restart: unless-stopped
  cap_add: [NET_ADMIN]
  networks:
    gsm-inter:
      ipv4_address: 172.20.0.10
  # 2908 sur la boucle locale, en SCTP : le raccord de l operateur NATIF
  # (start-native.sh), voir start-docker.sh start_inter_stp.
  ports:
    - "127.0.0.1:2908:2908/sctp"
  volumes:
    - ${OSMO_COMPOSE_DIR:-./data/compose}/hub/osmo-stp-interop.cfg:/etc/osmocom/osmo-stp-interop.cfg:ro
  profiles: [bench]

operator-1:
  image: ${IMAGE_RUN:-osmocom-run}
  container_name: osmo-operator-1
  hostname: osmo-operator-1
  restart: unless-stopped
  command: ["sleep", "infinity"]
  depends_on: [inter-stp]
  cap_add: [NET_ADMIN, SYS_ADMIN, NET_RAW]
  ulimits:
    rtprio: 18
  shm_size: 8g
  cgroup: host
  devices:
    - /dev/net/tun:/dev/net/tun
  tmpfs:
    - /tmp:exec,rw,mode=1777,size=2g
    - /run:exec,size=64m
    - /run/lock
  volumes:
    - /sys/fs/cgroup:/sys/fs/cgroup:rw
    - /tmp/osmocom-logs/op1:/var/log/osmocom
    - ${OSMO_COMPOSE_DIR:-./data/compose}/op1/osmocom:/etc/osmocom
    - ${OSMO_COMPOSE_DIR:-./data/compose}/op1/asterisk/extensions.conf:/etc/asterisk/extensions.conf
    - ${OSMO_COMPOSE_DIR:-./data/compose}/op1/asterisk/pjsip.conf:/etc/asterisk/pjsip.conf
    - ${OSMO_COMPOSE_DIR:-./data/compose}/op1/asterisk/rtp.conf:/etc/asterisk/rtp.conf
    - ${OSMO_COMPOSE_DIR:-./data/compose}/op1/asterisk/modules.conf:/etc/asterisk/modules.conf
    - ${OSMO_COMPOSE_DIR:-./data/compose}/op1/asterisk/annuaire.conf:/etc/asterisk/annuaire.conf
    - ${OSMO_COMPOSE_DIR:-./data/compose}/op1/bb/mobile.cfg:/root/.osmocom/bb/mobile.cfg
    - ${OSMO_COMPOSE_DIR:-./data/compose}/asound.conf:/etc/asound.conf:rw
  environment:
    OPERATOR_ID: "1"
    OSMO_ROLE: operator
    N_MS: ${N_MS:-2}
    CONTAINER_IP: ${OP1_PRIV_IP:-192.168.2.10}
    GATEWAY_IP: ${OP1_PRIV_GW:-192.168.2.1}
    INTER_STP_IP: 172.20.0.10
    OSMO_HUB_IP: 172.20.0.10
    WAN_NODE_ID: ${WAN_NODE_ID:-1}
    OSMO_WAN_NODE: ${WAN_NODE_ID:-1}
    HOST_IP: ${HOST_IP:-}
    SIP_HOST_PORT: "5060"
    PHY_MODE: ${PHY_MODE:-faketrx}
    HOST_AUDIO_RELAY: ${HOST_AUDIO_RELAY:-}
    PULSE_SERVER: ${PULSE_SERVER:-tcp:172.20.0.1:4713}
    ALSA_OUTPUT: gsm_out
```

```
      ALSA_INPUT: gsm_in
      ALSA_CARD: gsm_out
      GAPK_ALSA_DEV: gsm_out
    networks:
      gsm-inter:
        ipv4_address: ${OP1_BACKBONE_IP:-172.20.0.11}
      gsm-net-op1:
        ipv4_address: ${OP1_PRIV_IP:-192.168.2.10}
    profiles: [bench]

networks:
  # Le backbone SS7 : hub en .10, operateur N en .(10+N) - INTER_NET de start-docker.sh.
  gsm-inter:
    name: gsm-inter
    driver: bridge
    ipam:
      config:
        - subnet: 172.20.0.0/24
          gateway: 172.20.0.1
  # Le reseau prive de l operateur 1 - gsm-net-op1 de start-docker.sh (op_private_*).
  gsm-net-op1:
    name: gsm-net-op1
    driver: bridge
    ipam:
      config:
        - subnet: ${OP1_PRIV_NET:-192.168.2.0/24}
          gateway: ${OP1_PRIV_GW:-192.168.2.1}
```

3.11 /opt/GSM/osmo-operator/full_qmd.sh

21236 octets, 596 lignes → 596 lignes

```
#!/bin/bash
# full_qmd.sh - Bundle du depot en Quarto : doc + tests + .h + .c + .py + .sh,
# chaque fichier dans un bloc code type, theme "sketchy".
#
# DEUX MODES
#   MODE=single (defaut) : un seul .qmd. Simple, mais Deno/Quarto casse
#                          ("Maximum call stack size exceeded") au-dela de
#                          ~1-2 Mo, surtout avec embed-resources.
#   MODE=site            : projet Quarto multi-pages (recommande au-dela de 1 Mo).
#                          Sections decoupees en pages de SPLIT_KB max,
#                          recherche plein-texte integree, index global.
#
# Usage :
# TOUT EST ECRIT DANS UN SEUL DOSSIER ($OUTDIR, par defaut ./bundle-qmd/) :
# le .qmd, sketchy.css, sk-filter.html, et le rendu quarto. Le depot reste
# propre, et une seconde execution ne rescanne pas sa propre sortie.
#
# Usage :
#   MODE=site ./full_qmd.sh              # -> ./bundle-qmd/ (multi-pages)
#   ./full_qmd.sh                        # -> ./bundle-qmd/<depot>-full.qmd
#   ./full_qmd.sh out.qmd                # chemin explicite (mode single)
#   OUTDIR=/tmp/x ./full_qmd.sh          # ailleurs
#   PL4Y_SECTION=calypso ./full_qmd.sh   # section pl4y.store visee
#   SCOPE=hw/arm/calypso ./full_qmd.sh   # only this subdir
#   EXCLUDE='build|pc-bios' ./full_qmd.sh
#   MAX_KB=256 ./full_qmd.sh             # troncature par fichier
#   SPLIT_KB=300 MODE=site ./full_qmd.sh # taille max d'une page
#   EMBED=true ./full_qmd.sh             # html self-contained (risque de crash)
#   RENDER_MERMAID=0 ./full_qmd.sh       # .mmd en code au lieu de diagramme
#   RENDER=0 MODE=site ./full_qmd.sh     # sources .qmd seules, sans rendu

set -uo pipefail
```

```
case "${1:-}" in -h|--help) sed -n '2,30p' "$0"; exit 0;; esac

MODE="${MODE:-single}"
# Dossier unique de sortie, pour les DEUX modes. Le mode single ecrivait avant
# a la racine du depot (calypso-full.qmd + sketchy.css + sk-filter.html, puis
# calypso-full.html et calypso-full_files/ au rendu) : quatre artefacts laches
# dans le depot, que la passe suivante rescannait comme des sources.
# Nom derive du depot (osmo-operator -> ./osmo-operator-qmd) : les bundles de deux
# depots differents peuvent ainsi etre deposes cote a cote sans collision.
OUTDIR="${OUTDIR:-./$(basename "$(cd "$(dirname "$0")" && pwd)")-qmd}"
# Section pl4y.store visee par ce bundle : ecrite dans .pl4y-section a la racine
# du dossier de sortie. build.mjs (dans pl4y) la lit pour savoir sous quelle URL
# publier, plutot que de deviner d'apres le nom du dossier.
PL4Y_SECTION="${PL4Y_SECTION:-}"
OUT="${1:-}"
SCOPE="${SCOPE:-.}"
TITLE="${TITLE:-Calypso QEMU}"
SUBTITLE="${SUBTITLE:-Full source bundle - DSP C54x / osmocom-bb / SHUNT_LEGIT}"
MAX_KB="${MAX_KB:-512}"
SPLIT_KB="${SPLIT_KB:-400}"
RENDER_MERMAID="${RENDER_MERMAID:-1}"
EMBED="${EMBED:-false}"
# En mode site on rend par defaut : le dossier produit est alors DIRECTEMENT
# deposable dans pl4y/bundles/ (il contient _site/, du HTML). Sans rendu, ce
# n'est qu'un projet Quarto, que rien en aval ne sait publier.
# RENDER=0 pour ne generer que les sources .qmd.
if [ "$MODE" = "site" ]; then RENDER="${RENDER:-1}"; else RENDER="${RENDER:-0}"; fi
EXCLUDE_RE="${EXCLUDE:-subprojects|build|pc-
↪ bios|tests/functional|tests/qtest|tests/unit|tests/migration|tests/qemu-
↪ iotests|node_modules|\.git|\.pytest_cache}"

HERE="$(cd "$(dirname "$0")" && pwd)"
cd "$HERE"

# Le dossier existe avant toute chose : les deux modes ecrivent dedans.
mkdir -p "$OUTDIR" || { echo "ERROR: mkdir '$OUTDIR'" >&2; exit 1; }
OUTDIR="$(cd "$OUTDIR" && pwd)"

# Slug de section : explicite via PL4Y_SECTION, sinon derive du depot.
# qemu-calypso est publie sous /calypso/ — d'ou la seule correspondance
# particuliere.
if [ -z "$PL4Y_SECTION" ]; then
    case "$(basename "$HERE")" in
        qemu-calypso) PL4Y_SECTION="calypso" ;;
        *)            PL4Y_SECTION="$(basename "$HERE")" ;;
    esac
fi
printf '%s\n' "$PL4Y_SECTION" > "$OUTDIR/.pl4y-section"

# Sans argument, le .qmd va dans $OUTDIR. Un argument explicite reste
# respecte : `./full_qmd.sh /tmp/x.qmd` ecrit bien la (et son css/js a cote).
if [ -z "$OUT" ]; then
    OUT="$OUTDIR/$(basename "$HERE")-full.qmd"
elif [ -d "$OUT" ]; then
    OUT="${OUT%/}/$(basename "$HERE")-full.qmd"
fi

MAXB=$(( MAX_KB * 1024 ))
SPLITB=$(( SPLIT_KB * 1024 ))
TMPD="$(mktemp -d)"
trap 'rm -rf "$TMPD"' EXIT

GITREV="$(git rev-parse --short HEAD 2>/dev/null || echo 'n/a')"
GITBRANCH="$(git rev-parse --abbrev-ref HEAD 2>/dev/null || echo 'n/a')"
NOW="$(date -Iseconds)"
```

```
# --- cible + auto-exclusion ------------------------------------------------
SELF_FIXED=""      # fichier a exclure du scan (mode single)
SELF_PREFIX=""     # prefixe de dossier a exclure (mode site)

# $OUTDIR est exclu du scan dans LES DEUX modes. Auparavant le mode single
# n'excluait que le .qmd lui-meme : sketchy.css, sk-filter.html et le
# repertoire *_files/ du rendu precedent etaient reavales comme des sources a
# chaque execution, et le bundle grossissait a chaque passe.
SELF_PREFIX="${OUTDIR#"$HERE"/}/"
[ "$SELF_PREFIX" = "$OUTDIR/" ] && SELF_PREFIX=""   # $OUTDIR hors du depot

if [ "$MODE" = "site" ]; then
    DEST=""
else
    mkdir -p "$(dirname "$OUT")" || { echo "ERROR: mkdir pour '$OUT'" >&2; exit 1; }
    : > "$OUT" || { echo "ERROR: cannot write '$OUT'" >&2; exit 1; }
    OUT_ABS="$(cd "$(dirname "$OUT")" && pwd)/$(basename "$OUT")"
    SELF_FIXED="${OUT_ABS#"$HERE"/}"
    DEST="$OUT"
fi

echo "=== full_qmd.sh ==="
echo "Mode     : $MODE"
echo "Scope    : $SCOPE"
echo "Output   : $([ "$MODE" = site ] && echo "$OUTDIR/" || echo "$OUT")"
echo "Exclude  : $EXCLUDE_RE"
echo "Max/file : ${MAX_KB} ko$([ "$MODE" = site ] && echo " | page: ${SPLIT_KB} ko")"
echo

# ---------------------------------------------------------------- helpers --

_lang_for() {
    case "${1##*.}" in
        c|h)        echo c ;;
        cpp|cc|cxx) echo cpp ;;
        py)         echo python ;;
        sh|bash)    echo bash ;;
        md|qmd)     echo markdown ;;
        mmd)        echo mermaid ;;
        *)          echo default ;;
    esac
}

_fence_for() {
    local maxrun n
    maxrun=$(grep -oE '^`{3,}' "$1" 2>/dev/null | awk '{ if (length($0) > m) m = length($0) } END { print m+0 }')
    n=3; [ "${maxrun:-0}" -ge 3 ] && n=$(( maxrun + 1 ))
    printf '%*s' "$n" '' | tr ' ' '`'
}

_anchor_for() {
    printf 'f-%s' "$(printf '%s' "$1" | tr 'A-Z' 'a-z' | sed 's/[^a-z0-9]\+/-/g; s/^-//; s/-$//')"
}

_human() {
    if command -v numfmt >/dev/null 2>&1; then numfmt --to=iec --suffix=B "$1"; else echo "$1 B"; fi
}

_self_filter() {
    { if [ -n "$SELF_FIXED" ]; then grep -vxF "$SELF_FIXED"; else cat; fi; } \
  | { if [ -n "$SELF_PREFIX" ]; then grep -vF "$SELF_PREFIX"; else cat; fi; }
}

_find_into() {
    local excl="$1"; shift
```

```
    local args=() p first=1
    for p in "$@"; do
        if [ "$first" -eq 1 ]; then args+=( -name "$p" ); first=0
        else args+=( -o -name "$p" ); fi
    done
    find "$SCOPE" -type f \( "${args[@]}" \) 2>/dev/null \
        | grep -vE "$EXCLUDE_RE" \
        | { if [ -n "$excl" ]; then grep -vE "$excl"; else cat; fi; } \
        | sed 's|^\./||' \
        | _self_filter \
        | sort > "$TMPD/list"
    wc -l < "$TMPD/list"
}

# ---------------------------------------------------- emission d'un fichier

_add() {
    local f="$1" lang fence size nlines anchor ext eff
    ext="${f##*.}"
    lang="$(_lang_for "$f")"
    size=$(wc -c < "$f" 2>/dev/null || echo 0)
    nlines=$(wc -l < "$f" 2>/dev/null || echo 0)
    anchor="$(_anchor_for "$f")"

    {
        printf '\n## `%s` {#%s}\n\n' "$f" "$anchor"
        printf '::: {.file-meta}\n[.%s]{.tag} %s lignes - %s\n:::\n\n' "$ext" "$nlines" "$(_human "$size")"
    } >> "$DEST"

    if [ "$size" -eq 0 ]; then
        printf '::: {.callout-note appearance="simple"}\nFichier vide.\n:::\n' >> "$DEST"
        eff=0
    elif ! grep -Iq . "$f" 2>/dev/null; then
        printf '::: {.callout-warning appearance="simple"}\nFichier binaire - contenu omis.\n:::\n' >> "$DEST"
        eff=0
    else
        fence="$(_fence_for "$f")"
        if [ "$lang" = "mermaid" ] && [ "$RENDER_MERMAID" = "1" ]; then
            printf '%s{mermaid}\n' "$fence" >> "$DEST"
        else
            printf '%s{.%s .numberLines filename="%s"}\n' "$fence" "$lang" "$f" >> "$DEST"
        fi
        if [ "$size" -gt "$MAXB" ]; then
            head -c "$MAXB" "$f" >> "$DEST"
            printf '\n\n... [TRONQUE a %s ko - %s au total] ...\n' "$MAX_KB" "$(_human "$size")" >> "$DEST"
            eff=$MAXB
        else
            cat "$f" >> "$DEST"
            eff=$size
        fi
        printf '\n%s\n' "$fence" >> "$DEST"
    fi

    # manifest : section | page | fichier | ancre | lignes | taille
    printf '%s\t%s\t%s\t%s\t%s\t%s\n' \
        "$SECNUM" "$(basename "${DEST%.qmd}")" "$f" "$anchor" "$nlines" "$(_human "$size")" \
        >> "$TMPD/manifest"

    PAGE_BYTES=$(( PAGE_BYTES + eff + 400 ))
}

# ------------------------------------------------------------ mode site : page

_new_page() {
    local base
    base="$(printf 'sec%s-%02d' "$SECNUM" "$PART")"
```

```
    DEST="$OUTDIR/$base.qmd"
    printf '%s\t%s\t%s\t%s\n' "$SECNUM" "$SECTITLE" "$base.qmd" "partie $PART" >> "$TMPD/pages"
    cat > "$DEST" <<EOF
---
title: "$SECTITLE$([ "$PART" -gt 1 ] && printf ' - partie %s' "$PART")"
---

EOF
    PAGE_BYTES=0
}

# ---------------------------------------------------------------- une section

_section() {
    SECNUM="$1"; SECTITLE="$2"
    local extra_excl="$3" n f fsize
    shift 3
    n=$(_find_into "$extra_excl" "$@")
    echo "Section $SECNUM : $n files"

    if [ "$MODE" = "site" ]; then
        PART=1; _new_page
        while IFS= read -r f; do
            [ -n "$f" ] || continue
            fsize=$(wc -c < "$f" 2>/dev/null || echo 0)
            [ "$fsize" -gt "$MAXB" ] && fsize=$MAXB
            if [ "$PAGE_BYTES" -gt 0 ] && [ $(( PAGE_BYTES + fsize )) -gt "$SPLITB" ]; then
                PART=$(( PART + 1 )); _new_page
            fi
            _add "$f"
        done < "$TMPD/list"
    else
        {
            printf '\n\n# %s {#sec-%s}\n\n' "$SECTITLE" "$SECNUM"
            printf '::: {.section-count}\n**%s fichiers**\n:::\n' "$n"
        } >> "$DEST"
        PAGE_BYTES=0
        while IFS= read -r f; do
            [ -n "$f" ] && _add "$f"
        done < "$TMPD/list"
    fi
}

# ---------------------------------------------------------------- le CSS

_write_css() {
    cat > "$1" <<'SKETCHYCSS'
@import
 ↪  url('https://fonts.googleapis.com/css2?family=Caveat:wght@600;700&family=Architects+Daughter&family=JetBrains+Mono:wght@400

:root{
  --sk-ink:#20242e;
  --sk-paper:#fffdf6;
  --sk-red:#e4572e;
  --sk-teal:#17968a;
  --sk-yellow:#ffd469;
  --sk-rough:255px 12px 225px 15px / 15px 225px 15px 255px;
  --sk-rough2:15px 225px 15px 255px / 255px 15px 225px 15px;
}

body{
  background-color:var(--sk-paper);
  background-image:
    linear-gradient(rgba(32,36,46,.05) 1px, transparent 1px),
    linear-gradient(90deg, rgba(32,36,46,.05) 1px, transparent 1px);
  background-size:28px 28px;
```

```
  font-family:'Architects Daughter','Neucha',ui-rounded,system-ui,sans-serif;
  color:var(--sk-ink);
}

h1.title{
  font-family:'Caveat',cursive; font-size:3.6rem; line-height:1.05;
  display:inline-block; padding:.1em .5em; transform:rotate(-1.1deg);
  background:var(--sk-yellow); border:3px solid var(--sk-ink);
  border-radius:var(--sk-rough); box-shadow:6px 6px 0 var(--sk-ink);
}
.subtitle{ font-size:1.2rem; opacity:.75; transform:rotate(.4deg); display:inline-block; }

h1:not(.title){
  font-family:'Caveat',cursive; font-size:2.9rem; margin-top:3.2rem;
  border-bottom:4px solid var(--sk-ink); border-radius:var(--sk-rough2);
  padding-bottom:.15em; transform:rotate(-.35deg);
}
h2{
  font-family:'JetBrains Mono',monospace; font-size:1.05rem; font-weight:700;
  margin-top:2.6rem; padding:.35em .7em; display:inline-block;
  background:#fff; border:2.5px dashed var(--sk-ink);
  border-radius:var(--sk-rough); box-shadow:3px 3px 0 rgba(23,150,138,.35);
}
h2 code{ background:none; color:var(--sk-ink); padding:0; font-size:1em; }

div.sourceCode, pre:not(.sourceCode){
  border:2.5px solid var(--sk-ink);
  border-radius:var(--sk-rough);
  box-shadow:5px 5px 0 rgba(32,36,46,.9), 10px 10px 0 rgba(228,87,46,.18);
  background:#fffef9 !important;
  transform:rotate(-.12deg);
}
code, pre, code span{ font-family:'JetBrains Mono',ui-monospace,monospace !important; }
div.sourceCode{ max-height:70vh; overflow:auto; }
code.sourceCode > span > a:first-child::before{ color:rgba(32,36,46,.28); }

.code-with-filename .code-with-filename-file{
  font-family:'Caveat',cursive !important; font-size:1.2rem; font-weight:700;
  background:var(--sk-yellow) !important; color:var(--sk-ink) !important;
  border:2.5px solid var(--sk-ink); border-bottom:none;
  border-radius:12px 110px 6px 6px / 40px 8px 6px 6px;
  transform:rotate(-.5deg); display:inline-block; padding:.15em .9em;
}
.code-with-filename .code-with-filename-file pre{ border:none; box-shadow:none; background:none; }

.file-meta{ font-size:.9rem; opacity:.8; margin:.1rem 0 .8rem .2rem; }
.file-meta .tag{
  background:var(--sk-teal); color:#fff; padding:.05em .55em; margin-right:.5em;
  border-radius:var(--sk-rough2); font-family:'JetBrains Mono',monospace; font-size:.8rem;
}
.section-count{
  display:inline-block; padding:.1em .8em; background:#fff;
  border:2px solid var(--sk-red); color:var(--sk-red);
  border-radius:var(--sk-rough2); transform:rotate(.7deg); margin-bottom:1rem;
}
.bundle-index table{ font-family:'JetBrains Mono',monospace; font-size:.82rem; width:100%; }
.bundle-index{
  border:2.5px solid var(--sk-ink); border-radius:var(--sk-rough);
  padding:.6rem 1rem; background:#fff; box-shadow:5px 5px 0 rgba(23,150,138,.3);
}
.bundle-index a{ text-decoration:none; }
.bundle-index tr:hover{ background:rgba(255,212,105,.35); }

#TOC, nav[role=doc-toc]{ font-family:'JetBrains Mono',monospace; font-size:.78rem; }
#TOC a.active, nav[role=doc-toc] a.active{ background:var(--sk-yellow); border-radius:var(--sk-rough2); }
#quarto-sidebar{ font-family:'JetBrains Mono',monospace; font-size:.85rem; }
```

```
.sk-filter{ position:sticky; top:0; z-index:50; padding:.4rem 0 .8rem; background:var(--sk-paper); }
.sk-filter input{
  width:100%; font-family:'Caveat',cursive; font-size:1.4rem; padding:.25em .8em;
  background:#fff; border:2.5px solid var(--sk-ink); border-radius:var(--sk-rough);
  box-shadow:4px 4px 0 rgba(228,87,46,.35); outline:none;
}
.sk-filter input:focus{ box-shadow:4px 4px 0 var(--sk-teal); }

a{ color:var(--sk-red); text-decoration-style:wavy; text-decoration-thickness:1px; }
blockquote{ border-left:4px solid var(--sk-teal); }
SKETCHYCSS
}

_write_filter_js() {
    cat > "$1" <<'FILTERJS'
<script>
document.addEventListener('DOMContentLoaded', function () {
  var box = document.getElementById('sk-filter');
  if (!box) return;
  var rows = Array.from(document.querySelectorAll('.bundle-index tbody tr'));
  var toc  = Array.from(document.querySelectorAll('#TOC a, nav[role=doc-toc] a'));
  box.addEventListener('input', function () {
    var q = box.value.trim().toLowerCase();
    rows.forEach(function (r) {
      r.style.display = (!q || r.textContent.toLowerCase().includes(q)) ? '' : 'none';
    });
    toc.forEach(function (a) {
      var li = a.closest('li') || a;
      li.style.display = (!q || a.textContent.toLowerCase().includes(q)) ? '' : 'none';
    });
  });
});
</script>
FILTERJS
}

: > "$TMPD/manifest"
: > "$TMPD/pages"

# ============================================================ MODE SINGLE ======

if [ "$MODE" != "site" ]; then

cat >> "$OUT" <<EOF
---
title: "$TITLE"
subtitle: "$SUBTITLE"
date: "$NOW"
date-format: "YYYY-MM-DD HH:mm"
lang: fr
format:
  html:
    theme: [sketchy]
    css: sketchy.css
    toc: true
    toc-depth: 2
    toc-location: left
    toc-title: "Fichiers"
    toc-expand: 1
    number-sections: false
    anchor-sections: true
    smooth-scroll: true
    code-copy: true
    code-overflow: scroll
    code-block-bg: true
```

```
    highlight-style: arrow
    embed-resources: $EMBED
    include-after-body: sk-filter.html
    grid:
      sidebar-width: 340px
      body-width: 1150px
      margin-width: 140px
    mermaid:
      theme: neutral
---

::: {.sk-filter}
<input id="sk-filter" type="text" placeholder="filtrer les fichiers (ex: dsp, .c, shunt)..." autocomplete="off">
:::

::: {.callout-note appearance="simple" icon=false}
**Bundle plat du depot** - doc, tests, headers, sources, python, shell.
Scope \`$SCOPE\` - git \`$GITBRANCH@$GITREV\` - genere $NOW
- exclusions \`$EXCLUDE_RE\` - troncature ${MAX_KB} ko/fichier.
:::
EOF

_write_css "$(dirname "$OUT")/sketchy.css"
_write_filter_js "$(dirname "$OUT")/sk-filter.html"

fi

# ============================================================ MODE SITE ======

if [ "$MODE" = "site" ]; then
    _write_css "$OUTDIR/sketchy.css"
    _write_filter_js "$OUTDIR/sk-filter.html"
fi

# ------------------------------------------------------------------ sections

_section 1 "1 - Documentation" ''                                  '*.md' '*.mmd' '*.qmd'
_section 2 "2 - Tests"         ''                                  'test_*.py' 'conftest.py'
_section 3 "3 - Headers"       ''                                  '*.h'
_section 4 "4 - Sources"       ''                                  '*.c' '*.cpp'
_section 5 "5 - Python"        '/tests/|test_.*\.py$|conftest\.py$' '*.py'
_section 6 "6 - Shell"         ''                                  '*.sh' '*.bash'

# ------------------------------------------------------- index + config site --

if [ "$MODE" = "site" ]; then

    # index.qmd : table globale, liens inter-pages
    {
        cat <<EOF
---
title: "$TITLE"
subtitle: "$SUBTITLE"
date: "$NOW"
date-format: "YYYY-MM-DD HH:mm"
---

::: {.callout-note appearance="simple" icon=false}
**Bundle du depot** - doc, tests, headers, sources, python, shell.
Scope \`$SCOPE\` - git \`$GITBRANCH@$GITREV\` - genere $NOW
- exclusions \`$EXCLUDE_RE\` - troncature ${MAX_KB} ko/fichier - pages de ${SPLIT_KB} ko max.
:::

::: {.sk-filter}
<input id="sk-filter" type="text" placeholder="filtrer les fichiers (ex: dsp, .c, shunt)..." autocomplete="off">
:::
```

```
EOF
        cursec=""
        while IFS=$'\t' read -r sec page f anchor nlines size; do
            if [ "$sec" != "$cursec" ]; then
                [ -n "$cursec" ] && printf '\n:::\n'
                title="$(awk -F'\t' -v s="$sec" '$1==s {print $2; exit}' "$TMPD/pages")"
                printf '\n# %s\n\n::: {.bundle-index}\n\n' "$title"
                printf '| Fichier | Lignes | Taille |\n|:---|---:|---:|\n'
                cursec="$sec"
            fi
            printf '| [`%s`](%s.html#%s) | %s | %s |\n' "$f" "$page" "$anchor" "$nlines" "$size"
        done < "$TMPD/manifest"
        printf '\n:::\n'
    } > "$OUTDIR/index.qmd"

    # _quarto.yml
    {
        cat <<EOF
project:
  type: website
  output-dir: _site

website:
  title: "$TITLE"
  search:
    location: sidebar
    type: textbox
  sidebar:
    style: floating
    collapse-level: 1
    contents:
      - text: "Index"
        href: index.qmd
EOF
        cursec=""
        while IFS=$'\t' read -r sec title page part; do
            if [ "$sec" != "$cursec" ]; then
                printf '      - section: "%s"\n        contents:\n' "$title"
                cursec="$sec"
            fi
            printf '          - href: %s\n            text: "%s"\n' "$page" "$part"
        done < "$TMPD/pages"

        cat <<EOF

format:
  html:
    theme: [sketchy]
    css: sketchy.css
    toc: true
    toc-depth: 2
    toc-location: right
    toc-title: "Fichiers"
    number-sections: false
    anchor-sections: true
    smooth-scroll: true
    code-copy: true
    code-overflow: scroll
    code-block-bg: true
    highlight-style: arrow
    embed-resources: false
    include-after-body: sk-filter.html
    mermaid:
      theme: neutral
EOF
    } > "$OUTDIR/_quarto.yml"
```

```
fi

# ---------------------------------------------------------------------- footer

if [ "$MODE" != "site" ]; then
cat >> "$OUT" <<EOF

# Fin du bundle

::: {.section-count}
genere $NOW - \`$(basename "$0")\`
:::
EOF
fi

echo
echo "=== DONE ==="
if [ "$MODE" = "site" ]; then
    echo "Output : $OUTDIR/  ($(ls "$OUTDIR"/*.qmd | wc -l) pages, $(du -sh "$OUTDIR" | cut -f1))"
    echo "Rendu  : (cd \"$OUTDIR\" && quarto render)"
    echo "Preview: (cd \"$OUTDIR\" && quarto preview)"
else
    echo "Output : $OUT ($(du -h "$OUT" | cut -f1), $(wc -l < "$OUT") lignes)"
    echo "         + sketchy.css + sk-filter.html dans $(dirname "$OUT")/"
    echo "Rendu  : quarto render \"$OUT\"   (sortie dans $(dirname "$OUT")/)"
    if [ "$(wc -c < "$OUT")" -gt 1500000 ]; then
        echo
        echo "!! $(du -h "$OUT" | cut -f1) dans un seul document : Deno risque de casser"
        echo "   (RangeError: Maximum call stack size exceeded). Utilise MODE=site."
    fi
fi

if [ "$RENDER" = "1" ]; then
    if command -v quarto >/dev/null 2>&1; then
        if [ "$MODE" = "site" ]; then (cd "$OUTDIR" && quarto render); else quarto render "$OUT"; fi
    else
        echo "quarto introuvable dans le PATH - rendu ignore." >&2
    fi
fi
```

3.12 /opt/GSM/osmo-operator/full_text.sh

5944 octets, 166 lignes → 166 lignes

```
#!/bin/bash
# full_text.sh - Concat doc + tests + .h + .c + .py + .sh dans cet ordre
# en un seul .txt brut (separateurs ASCII, sans markdown fences).
# Pour ingestion LLM / archive plate.
#
# Usage :
#   ./full_text.sh                          # -> /tmp/calypso-full.txt
#   ./full_text.sh out.txt                  # custom output
#   SCOPE=hw/arm/calypso ./full_text.sh     # only this subdir
#   EXCLUDE='build|pc-bios' ./full_text.sh  # extra excludes

set -uo pipefail

OUT="${1:-./calypso-full.txt}"
SCOPE="${SCOPE:-.}"
EXCLUDE_RE="${EXCLUDE:-subprojects|build|pc-
↪  bios|tests/functional|tests/qtest|tests/unit|tests/migration|tests/qemu-
↪  iotests|node_modules|\.git|\.pytest_cache}"

# Si $OUT est un dossier, append le nom de fichier par defaut.
```

```
# Evite le cas "./full_text.sh /home/nirvana/qemu-calypso" -> echo >> dossier crash.
if [ -d "$OUT" ]; then
    OUT="${OUT%/}/calypso-full.txt"
fi

HERE="$(cd "$(dirname "$0")" && pwd)"
cd "$HERE"

echo "=== full_text.sh ==="
echo "Scope    : $SCOPE"
echo "Output   : $OUT"
echo "Exclude  : $EXCLUDE_RE"
echo

: > "$OUT" || { echo "ERROR: cannot write to '$OUT' (permission ? path invalide ?)" >&2; exit 1; }

# ---- Header ----
cat >> "$OUT" <<EOF
================================================================================
Calypso QEMU - Full text bundle
Generated : $(date -Iseconds)
Scope     : $SCOPE
Sections  : 1.docs  2.tests  3.headers  4.sources  5.python  6.shell
================================================================================

EOF

# Helper : add a file with ASCII separator
_add() {
    local f="$1"
    local rel="${f#./}"
    local size=$(wc -c < "$f" 2>/dev/null || echo 0)
    local nlines=$(wc -l < "$f" 2>/dev/null || echo 0)
    {
        echo ""
        echo "================================================================================"
        echo "FILE: $rel"
        echo "SIZE: $size bytes, $nlines lines"
        echo "================================================================================"
        cat "$f"
        echo ""
    } >> "$OUT"
}

# Find files of a given pattern, filtered by EXCLUDE_RE
_find() {
    find "$SCOPE" -type f \( $1 \) 2>/dev/null \
        | grep -vE "$EXCLUDE_RE" \
        | sort
}

# ---- 1) Documentation ----
{
    echo ""
    echo "################################################################################"
    echo "# SECTION 1 : DOCUMENTATION (.md, .mmd, .qmd)"
    echo "################################################################################"
} >> "$OUT"
DOCS=$(find "$SCOPE" -type f \( -name "*.md" -o -name "*.mmd" -o -name "*.qmd" \) 2>/dev/null \
        | grep -vE "$EXCLUDE_RE" | sort)
N=$(echo "$DOCS" | grep -c . || echo 0)
echo "Section 1 : $N files"
echo "Total docs files : $N" >> "$OUT"
for f in $DOCS; do _add "$f"; done

# ---- 2) Tests ----
```

```
{
    echo ""
    echo "################################################################################"
    echo "# SECTION 2 : TESTS (tests/*.py)"
    echo "################################################################################"
} >> "$OUT"
TESTS=$(find "$SCOPE" -type f \( -name "test_*.py" -o -name "conftest.py" \) 2>/dev/null \
        | grep -vE "$EXCLUDE_RE" | sort)
N=$(echo "$TESTS" | grep -c . || echo 0)
echo "Section 2 : $N files"
echo "Total tests files : $N" >> "$OUT"
for f in $TESTS; do _add "$f"; done

# ---- 3) Headers ----
{
    echo ""
    echo "################################################################################"
    echo "# SECTION 3 : HEADERS (.h)"
    echo "################################################################################"
} >> "$OUT"
HDRS=$(find "$SCOPE" -type f -name "*.h" 2>/dev/null \
        | grep -vE "$EXCLUDE_RE" | sort)
N=$(echo "$HDRS" | grep -c . || echo 0)
echo "Section 3 : $N files"
echo "Total headers files : $N" >> "$OUT"
for f in $HDRS; do _add "$f"; done

# ---- 4) Sources ----
{
    echo ""
    echo "################################################################################"
    echo "# SECTION 4 : SOURCES (.c, .cpp)"
    echo "################################################################################"
} >> "$OUT"
SRCS=$(find "$SCOPE" -type f \( -name "*.c" -o -name "*.cpp" \) 2>/dev/null \
        | grep -vE "$EXCLUDE_RE" | sort)
N=$(echo "$SRCS" | grep -c . || echo 0)
echo "Section 4 : $N files"
echo "Total sources files : $N" >> "$OUT"
for f in $SRCS; do _add "$f"; done

# ---- 5) Python scripts (hors tests/) ----
{
    echo ""
    echo "################################################################################"
    echo "# SECTION 5 : PYTHON SCRIPTS (hors tests/)"
    echo "################################################################################"
} >> "$OUT"
PYS=$(find "$SCOPE" -type f -name "*.py" 2>/dev/null \
       | grep -vE "$EXCLUDE_RE|/tests/|test_.*\.py$|conftest\.py$" | sort)
N=$(echo "$PYS" | grep -c . || echo 0)
echo "Section 5 : $N files"
echo "Total python files : $N" >> "$OUT"
for f in $PYS; do _add "$f"; done

# ---- 6) Shell scripts ----
{
    echo ""
    echo "################################################################################"
    echo "# SECTION 6 : SHELL SCRIPTS (.sh)"
    echo "################################################################################"
} >> "$OUT"
SHS=$(find "$SCOPE" -type f \( -name "*.sh" -o -name "*.bash" \) 2>/dev/null \
        | grep -vE "$EXCLUDE_RE" | sort)
N=$(echo "$SHS" | grep -c . || echo 0)
echo "Section 6 : $N files"
```

```
echo "Total shell files : $N" >> "$OUT"
for f in $SHS; do _add "$f"; done

# ---- Footer ----
{
    echo ""
    echo "================================================================================"
    echo "END OF BUNDLE - generated $(date -Iseconds)"
    echo "================================================================================"
} >> "$OUT"

echo
echo "=== DONE ==="
echo "Output : $OUT"
echo "Size   : $(du -h "$OUT" | cut -f1)"
echo "Lines  : $(wc -l < "$OUT")"
```

3.13 /opt/GSM/osmo-operator/generate_configs.sh

54461 octets, 979 lignes → 975 lignes (2 groupes compactés)

```
#!/usr/bin/env bash
# ============================================================================
#  generate_configs.sh - le SEUL endroit ou la configuration du reseau est
#                        decidee, et le seul qui sait substituer les gabarits.
# ============================================================================
#
#  POURQUOI CE FICHIER EXISTE
#  --------------------------
#  La mecanique est inchangee : les fichiers de `configs/` contiennent des
#  marqueurs `__MCC__`, `__ARFCN__`, `__KI__`... qu'une fonction remplace au
#  demarrage. Ce qui change, c'est qu'il n'y a plus DEUX implementations de
#  cette fonction - `start-docker.sh` en portait une copie de 132 lignes et
#  `lib/gabarits.sh` une autre de 70, identiques au formatage pres, qui
#  derivaient donc en silence. Elles vivent ici, une fois.
#
#  Et surtout : les valeurs n'etaient calculees NULLE PART de facon visible.
#  ARFCN, BSIC, IMSI, IMEI, Ki etaient des formules enfouies au milieu de la
#  substitution (`arfcn=$(( 512 + op_id * 2 ))`). Pour changer l'ARFCN il
#  fallait trouver et editer la formule. Desormais ces formules ne sont plus
#  que des DEFAUTS : `globals.conf`, a la racine, les expose toutes et gagne.
#
#  DEUX USAGES
#  -----------
#    ./generate_configs.sh [--force] [--show] [CLE=VALEUR ...]
#         ecrit/complete globals.conf a la racine, puis affiche l'effectif.
#         Sans --force, un globals.conf existant n'est JAMAIS ecrase : c'est
#         ton fichier, on ne reecrit pas par-dessus tes reglages.
#
#    . ./generate_configs.sh
#         charge globals.conf et definit apply_config_templates().
#         C'est ce que font start-docker.sh (hote) et lib/gabarits.sh.
#
#  QUI APPELLE QUOI
#  ----------------
#    start-docker.sh          -> execute ce script (genere globals.conf) puis le source
#    start-native.sh   -> source globals.conf (docker : la conf est deja la)
#    lib/gabarits.sh   -> source ce script pour apply_config_templates()
#
#  MULTI-OPERATEUR
#  ---------------
#  En mode `virtual` (N_OPERATORS > 1) chaque operateur a besoin de valeurs
#  DISTINCTES. Une valeur posee dans globals.conf s'applique alors a TOUS -
#  ce qui est presque toujours faux pour ARFCN, LAC, IMSI... Laisse-les vides
```

```
#  (le defaut) pour garder la derivation par operateur, et ne fixe que ce que
#  tu veux vraiment figer. Le script te previent si tu figes un champ
#  per-operateur avec N_OPERATORS > 1.
#
#  SPDX-License-Identifier: GPL-2.0-or-later
# ============================================================================

_GC_HERE="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
_GC_FILE="$_GC_HERE/globals.conf"

# ── Les reglages exposes, avec leur defaut historique ──────────────────────
# Format : CLE|VALEUR-PAR-DEFAUT|PER-OP|COMMENTAIRE
#   PER-OP=1 -> la valeur est normalement derivee du numero d'operateur ;
#               la figer n'a de sens qu'avec un seul operateur.
#
#  ATTENTION : le corps de _gc_table est une TABLE, pas du shell. La seule ligne
#  non-donnee admise est « #SECTION|<titre> ». Un « # » ordinaire y devient une
#  entree malformee -- verifie le 27/08, en cassant le module « gabarits » du run.
#  Toute explication se met donc ICI, au-dessus de la fonction.
#
#  [2026-08-27] A5/1 EST LE DEFAUT ASSUME, ET IL A UN COUT CONNU. Le mobile
#  derriere QEMU ne dechiffre pas encore l'A5/1 : le chemin cipher du modele n'est
#  pas exerce (hw/arm/calypso/l1ctl_sock.c, « s'execute donc JAMAIS -- a verifier
#  avant d'activer l'A5/1 »). Comptage des rejets LAPDm dans mobile.log de part et
#  d'autre du CIPHERING MODE COMPLETE : 2 avant, 477 apres, et le LU s'arretait la
#  (T3210). Le side-car osmocom-bb, lui, traite l'A5/1 sans broncher et obtient
#  son LU ACCEPT : c'est le chemin emule qui doit rattraper, pas le reseau qui
#  doit baisser la garde. Repli de travail : ENCRYPTION="a5 0" en ligne de
#  commande, qui traverse jusqu'au fichier pose depuis que 08-gabarits.sh
#  l'exporte.
_gc_table() {
cat <<'TABLE'
#SECTION|Identite du reseau
MCC|001|0|Mobile Country Code (3 chiffres)
MNC|01|0|Mobile Network Code (2 chiffres)
OP_NAME|OsmoQEMU|0|Nom court et long du reseau
#SECTION|Securite et carte SIM
ENCRYPTION|a5 1|0|Chiffrement : "a5 1" (defaut), "a5 0" (aucun), "a5 0 1 3"...
SIM_ALGO|comp128|0|Algorithme d'authentification (comp128, xor...)
KI||1|Ki de la SIM, 16 octets hexa espaces
IMSI||1|IMSI du mobile
IMEI||1|IMEI du mobile
#SECTION|Radio
ARFCN||1|Canal radio (a accorder avec BAND)
BAND|DCS1800|0|Bande : GSM900, DCS1800, PCS1900, GSM850
BSIC||1|Base Station Identity Code, 0..63
LAC||1|Location Area Code
CELL_ID||1|Cell identity
IPA_UNIT_ID||1|Identifiant IPA du BTS
MS_MAX_POWER|15|0|Puissance max autorisee au mobile (0..15)
RXLEV_ACCESS_MIN|0|0|Niveau recu minimal pour acceder a la cellule
CELL_RESEL_HYST|4|0|Hysteresis de reselection de cellule (dB)
T3109|5|0|Timer T3109 du BSC (s) - liberation de canal
#SECTION|GPRS
BVCI||1|BSSGP Virtual Connection Id
NSEI||1|Network Service Entity Id
NSVCI||1|NS Virtual Connection Id
APN|internet|0|Nom du point d'acces
#SECTION|Services
SMS_SC||1|Numero du centre SMS
#SECTION|Dimensionnement
MS_COUNT|2|0|Nombre de mobiles simules
N_OPERATORS|1|0|Nombre d'operateurs (mode "virtual")
HOST_IP|127.0.0.1|0|Adresse de l'hote vue par la pile
TABLE
}
```

```
# ── Ecriture de globals.conf ──────────────────────────────────────
_gc_write() {
    local force=$1
    if [ -f "$_GC_FILE" ] && [ "$force" != "1" ]; then
        printf '  globals.conf existe deja - conserve (--force pour regenerer)\n'
        return 0
    fi
    {
        printf '# ═══════════════════════════════════════════════════════════════\n'
        printf '#  globals.conf - la configuration du reseau, en un seul endroit.\n'
        printf '# ═══════════════════════════════════════════════════════════════\n'
        printf '#\n'
        printf '#  Edite ce fichier a la main : il ne sera JAMAIS reecrit par un run.\n'
        printf '#  Seul "./generate_configs.sh --force" le regenere.\n'
        printf '#\n'
        printf '#  CE FICHIER FAIT AUTORITE - SUR LES VARIABLES QU'"'"'IL DECLARE, ET SUR\n'
        printf '#  ELLES SEULES. Il est lu en dernier : les %s reglages ci-dessous\n' "$(_gc_table | grep -vc
        ↪  '^#SECTION')"
        printf '#  ecrasent leur homonyme dans l'"'"'environnement. Tout le reste (CALYPSO_*,\n'
        printf '#  LOG_DIR, MODE...) passe au travers, intact. Deux facons de le modifier :\n'
        printf '#      vim globals.conf\n'
        printf '#      ./generate_configs.sh ARFCN=520\n'
        printf '#\n'
        printf '#  Un champ VIDE = valeur calculee automatiquement. Les champs notes\n'
        printf '#  [auto/operateur] se deduisent du numero d'"'"'operateur : ne les fige\n'
        printf '#  que si N_OPERATORS=1, sinon tous les operateurs se marchent dessus.\n'
        printf '# ═══════════════════════════════════════════════════════════════\n'
        local key def perop comment
        while IFS='|' read -r key def perop comment; do
            [ -n "$key" ] || continue
            if [ "$key" = "#SECTION" ]; then
                printf '\n# ── %s %s\n\n' "$def" \
                    "$(printf '%.0s─' $(seq 1 $((56 - ${#def}))))"
                continue
            fi
            printf '# %s%s\n' "$comment" \
                "$([ "$perop" = 1 ] && printf '  [auto/operateur]')"
            case "$def" in
                *" "*|"") printf '%s="%s"\n' "$key" "$def" ;;
                *)        printf '%s=%s\n'   "$key" "$def" ;;
            esac
        done < <(_gc_table)
    } > "$_GC_FILE"
    printf '  globals.conf ecrit (%s reglages)\n' \
           "$(_gc_table | grep -vc '^#SECTION')"
}

# ── Poser une valeur DANS le fichier, en place, sans casser la mise en forme ──
_gc_set() {
    local key="${1%%=*}" val="${1#*=}"
    _gc_table | grep -q "^${key}|" || {
        printf '  ⚠ reglage inconnu : %s (voir --show)\n' "$key" >&2; return 1; }
    [ -f "$_GC_FILE" ] || _gc_write 1 >/dev/null
    local q; case "$val" in *" "*|"") q="\"$val\"" ;; *) q="$val" ;; esac
    if grep -qE "^${key}=" "$_GC_FILE"; then
        sed -i "s|^${key}=.*|${key}=${q}|" "$_GC_FILE"
    else
        printf '%s=%s\n' "$key" "$q" >> "$_GC_FILE"
    fi
    printf '  %s = %s\n' "$key" "${val:-(auto)}"
}

# ── Chargement ────────────────────────────────────────────────────
# Charge globals.conf, puis COMPLETE avec les defauts de la table.
#
```

```
# L'ordre compte : le fichier est lu d'abord et fait autorite sur les variables
# qu'il declare ; ensuite, toute cle encore vide recoit son defaut d'usine.
#
# Ce filet n'est pas decoratif. Sans lui, un globals.conf ABSENT - clone frais,
# fichier volontairement gitignore, suppression accidentelle - laissait BAND,
# SIM_ALGO, APN, T3109... vides, et la substitution les ecrivait tels quels dans
# les configs : "band" sans valeur, "ki  <hexa>" sans algorithme. osmo-bsc
# refuse alors de demarrer, et rien dans le journal ne dit pourquoi. Constate
# en vrai le 2026-08-03. Les defauts appartiennent au SCRIPT ; globals.conf
# n'est qu'une couche de surcharge par-dessus.
gc_load() {
    local key def perop comment
    # ── LA LIGNE DE COMMANDE PRIME SUR globals.conf ──────────────────────
    # globals.conf affecte SECHEMENT (MCC=001, ENCRYPTION="a5 1", ...). Le sourcer
    # en premier ecrasait donc ce que l'appelant venait de poser dans son
    # environnement : sur une machine qui a un globals.conf,
    #     ENCRYPTION="a5 0" ./run.sh
    # etait avale sans un mot et le run repartait en a5 1. Constate le 2026-08-27,
    # et c'est la meme famille de panne que les deux autres du jour : une valeur
    # qui depend du chemin par lequel on entre, sans que rien ne le signale.
    #
    # On releve donc les cles DEJA presentes dans l'environnement AVANT de sourcer,
    # et on les repose APRES. Precedence retablie, dans l'ordre habituel :
    #     ligne de commande  >  globals.conf  >  defaut de _gc_table
    local -A _gc_env=()
    while IFS='|' read -r key def perop comment; do
        [ -n "$key" ] && [ "$key" != "#SECTION" ] || continue
        if [ -n "${!key:-}" ]; then _gc_env["$key"]="${!key}"; fi
    done < <(_gc_table)

    [ -f "$_GC_FILE" ] && . "$_GC_FILE"

    if [ "${#_gc_env[@]}" -gt 0 ]; then
        for key in "${!_gc_env[@]}"; do printf -v "$key" '%s' "${_gc_env[$key]}"; done
    fi

    while IFS='|' read -r key def perop comment; do
        [ -n "$key" ] && [ "$key" != "#SECTION" ] || continue
        [ -n "${!key:-}" ] || printf -v "$key" '%s' "$def"
    done < <(_gc_table)
    return 0
}

# ── IMEI tire au hasard, avec sa cle de Luhn ─────────────────────────────
# L'IMEI etait derive du numero d'operateur : deux bancs identiques rendaient
# les memes, et deux MS du meme operateur ne se distinguaient plus dans une
# trace. On garde un TAC plausible (35892500) et on tire les six chiffres de
# serie, puis on calcule la 15e position : un IMEI dont la cle est fausse est
# rejete par certains equipements, et surtout il se voit tout de suite comme
# fabrique a la main.
rand_imei() {
    local body d i sum=0 dbl parity
    body="35892500$(printf '%06d' "$(( (RANDOM << 15 | RANDOM) % 1000000 ))")"
    # Luhn sur les 14 chiffres : on double un chiffre sur deux en partant de la
    # droite, en repliant les resultats a deux chiffres.
    for (( i = ${#body} - 1; i >= 0; i-- )); do
        d=${body:i:1}
        parity=$(( (${#body} - 1 - i) % 2 ))
        if [ "$parity" -eq 0 ]; then
            dbl=$(( d * 2 )); [ "$dbl" -gt 9 ] && dbl=$(( dbl - 9 ))
            sum=$(( sum + dbl ))
        else
            sum=$(( sum + d ))
        fi
    done
    printf '%s%d' "$body" "$(( (10 - sum % 10) % 10 ))"
```

```
}

# ── LE PLAN DE NUMEROTATION, EN UN SEUL ENDROIT ──────────────────────────
#
#   MSISDN = <noeud> 00 <operateur> <rang du MS sur 2 chiffres>
#            noeud ⟨1⟩, operateur 1, MS ⟨2⟩  ->  ⟨3⟩  ×3
    ⟨⟩ = (1,1,100101) (1,2,100102) (2,1,200101)
#
# POURQUOI LE NOEUD EN TETE. Le plan precedent commencait par un 6 fixe
# (600000 + op*100 + ms) : le numero ne disait donc RIEN de la machine qui le
# porte. Sur un banc a plusieurs noeuds, l'operateur 1 du noeud 1 et l'operateur
# 1 du noeud 2 revendiquaient tous deux 600101 - le meme abonne a deux endroits,
# et un SMS ou un appel qui part chez le premier qui repond. Le premier chiffre
# devient donc l'adresse du noeud, et le routage (SMS comme dialplan) se lit
# directement dans le numero.
#
# CE QUI NE CHANGE PAS, ET C'EST VOULU : l'IMSI (MCC MNC <op sur 4> <ms sur 6>)
# et le Ki (00 11 ... dd <ms> <op>). Ces deux-la sont derives a l'IDENTIQUE dans
# quatre endroits - ici pour mobile.cfg, start-native.sh pour les mobiles,
# run_modules/21-abonnes-hlr.sh et start-docker.sh pour le HLR - et ils doivent rester
# d'accord au bit pres, sans quoi l'authentification echoue sur un Kc different
# des deux cotes, sans autre trace qu'un rejet de CIPHER MODE. On ne touche donc
# qu'au MSISDN, qui n'entre dans aucun calcul de securite.
osmo_msisdn_pfx() { printf '%s00' "${1:-1}"; }                      # <noeud>00
osmo_msisdn()     { printf '%d' "$(( ${1:-1} * 100000 + ${2:-1} * 100 + ${3:-1} ))"; }

# Le noeud tel que le voit la generation : l'environnement d'abord (c'est ce que
# start-docker.sh passe a un conteneur et ce que start-native.sh exporte avant un
# --regen), /etc/osmo-role ensuite, 1 par defaut. JAMAIS vide : un plan de
# numerotation sans noeud, c'est le plan de tout le monde.
osmo_node_id() {
    local n="${OSMO_WAN_NODE:-${WAN_NODE_ID:-}}"
    if [ -z "$n" ] && [ -r "${ROLE_FILE:-/etc/osmo-role}" ]; then
        n="$(awk -F= '/^OSMO_WAN_NODE=/{gsub(/[ \r\t]/,"",$2);v=$2} END{print v}' \
             "${ROLE_FILE:-/etc/osmo-role}" 2>/dev/null)"
    fi
    case "$n" in [1-9]) ;; *) n=1 ;; esac
    printf '%s' "$n"
}

# ── La substitution des gabarits - implementation UNIQUE ────────────────
#  Signature inchangee, pour rester compatible avec les deux appelants :
#    apply_config_templates DEST CONTAINER_IP GATEWAY_IP OP_ID \
#                           PC_MSC PC_STP PC_BSC MCC MNC OP_NAME \
#                           INTER_STP INTER_STP_SHUTDOWN N_OPERATORS
apply_config_templates() {
    local dest=$1 container_ip=$2 gateway_ip=$3 op_id=$4

    # ── OSMO_NO_REGEN : ne pas refaire ce qui vient d'etre fait ──────────
    # run.sh (qosmo-grgsm) peut rejouer cette fonction au demarrage. Dans un
    # conteneur, c'est une REGRESSION : les configs viennent d'etre generees par
    # start-docker.sh, avec l'identite du noeud, ses point codes et son inter-STP. Les
    # regenerer depuis les gabarits les remplace par les valeurs par defaut -
    # 1.1.2 pour tout le monde, ASP vers le hub docker, en shutdown - et
    # l'interco meurt sans un mot.
    #
    # Ce commentaire nommait run_modules/08-gabarits.sh comme cet appelant. Ce
    # module N'EXISTE PAS - start-native.sh le constate deja. Les appelants
    # reels sont start-docker.sh (deux fois : la boucle des operateurs et le mode
    # net-host) et build-iso.sh, qui construit l'arbre de l'image. Decrire un
    # module imaginaire a fait regler plus bas une option sur une configuration
    # que personne ne pose.
    #
    # start-native.sh pose donc cette variable par defaut, et --regen la leve
    # pour qui veut vraiment repartir des gabarits.
    if [ "${OSMO_NO_REGEN:-0}" = "1" ]; then
```

```
    echo "  gabarits : conserves (OSMO_NO_REGEN=1 ; --regen pour regenerer)"
    return 0
fi
local pc_msc=$5 pc_stp=$6 pc_bsc=$7 mcc=$8 mnc=$9 op_name=${10}

# ── Le PLMN, avant d'ecrire quoi que ce soit ──────────────────────────
# osmo-msc REFUSE "network country code 000" : il abandonne la lecture du
# fichier a cette ligne, sort, et systemd le relance - trois fois par
# seconde, indefiniment. Le journal du lanceur, lui, affiche une croix a
# cote d'osmo-msc puis annonce "Core Osmocom pret" : on cherche la panne
# dans le SS7 ou dans la BTS pendant que la cause tient en trois chiffres.
# Un MCC vide donne 000 par printf ; c'est ainsi que c'est arrive.
# On s'arrete ICI, ou le nom du parametre est encore connu.
case "$mcc" in
    ''|000|00|0) echo "  ERREUR : MCC invalide ('$mcc') pour l'operateur $op_id - osmo-msc refuserait la
    ↪  config" >&2; return 1 ;;
esac
case "$mnc" in
    ''|'-') echo "  ERREUR : MNC vide pour l'operateur $op_id" >&2; return 1 ;;
esac
local inter_stp=${11} inter_stp_shutdown=${12} n_operators=${13}

# Le noeud entre dans le plan de numerotation : MSISDN = <noeud>00<op><ms>.
# Il n'entre PAS dans l'IMSI ni dans le Ki - ceux-la sont derives a
# l'identique dans quatre endroits et doivent le rester (voir osmo_msisdn).
local node msisdn_pfx
node="$(osmo_node_id)"
msisdn_pfx="$(osmo_msisdn_pfx "$node")"

mkdir -p "$dest/osmocom" "$dest/asterisk" "$dest/bb"
local f bn

for f in configs/*.cfg; do
    [ "$(basename "$f")" = "osmo-stp-interop.cfg" ] && continue
    cp "$f" "$dest/osmocom/"
done
[ -f "configs/osmo-bts-virtual.cfg" ] && cp "configs/osmo-bts-virtual.cfg" "$dest/osmocom/"
for f in configs/*.conf; do
    bn=$(basename "$f"); [ "$bn" = "sms-routing.conf" ] && continue
    cp "$f" "$dest/asterisk/"
done
# scripts/ en entier : c'est le comportement de start-docker.sh, le plus permissif
# des deux qui coexistaient. lib/gabarits.sh en copiait une liste blanche,
# qui omettait silencieusement tout script ajoute depuis.
cp scripts/* "$dest/osmocom/" 2>/dev/null || true
chmod +x "$dest/osmocom"/*.sh 2>/dev/null || true

if [ -f "configs/mobile.cfg.template" ]; then
    cp "configs/mobile.cfg.template" "$dest/bb/mobile.cfg"
    cp "configs/mobile.cfg.template" "$dest/bb/mobile_group1.cfg"
elif [ -f "configs/mobile.cfg" ]; then
    cp "configs/mobile.cfg" "$dest/bb/mobile.cfg"
    cp "configs/mobile.cfg" "$dest/bb/mobile_group1.cfg"
fi

local rctx_msc rctx_stp rctx_bsc rctx_inter
rctx_msc=$(op_rctx_msc "$op_id");   rctx_stp=$(op_rctx_stp "$op_id")
rctx_bsc=$(op_rctx_bsc "$op_id")
# RCTX_INTER_OVERRIDE : pose par build-iso.sh quand l'image est un NOEUD DE
# WAN. Le routing context identifie l'AS aupres du hub ; la formule locale
# (op*100+50) donne la meme valeur sur tous les noeuds, donc plusieurs AS
# indiscernables sur le meme hub. Vide = formule d'origine, rien ne change.
rctx_inter="${RCTX_INTER_OVERRIDE:-$(op_rctx_inter "$op_id")}"

# ── Les valeurs. globals.conf gagne ; sinon, la formule historique. ────
local arfcn="${ARFCN:-$(( 512 + op_id * 2 ))}"
```

```
# ── La SECONDE BTS (le side-car) doit avoir son propre canal ──────────
# Elle etait figee sur l'ARFCN 516 et l'unit-id 6002 - les valeurs qui vont
# a l'operateur 1, et a lui seul. Pour l'operateur 2, dont la BTS
# principale tombe justement sur 516, les DEUX cellules emettaient sur la
# meme frequence : dans un milieu virtuel ou fake_trx apparie par
# frequence, leurs bursts se melangent. Constate sur le banc -
#     TRX 0 of BTS 0 is on ARFCN 516
#     TRX 0 of BTS 1 is on ARFCN 516
# - pendant que le mobile envoyait ses RACH sans jamais recevoir
# d'Immediate Assignment. L'operateur 1 n'y echappait que par coincidence.
# Le decalage de 100 met les side-cars hors d'atteinte de tout le plan
# principal (614, 616, 618... contre 514, 516, 518), et reste dans la
# bande DCS1800 (512-885).
local arfcn_bts1="${ARFCN_BTS1:-$(( 612 + op_id * 2 ))}"
# +2 sur le pas de 10 de la BTS principale : 6010/6012, 6020/6022...
local ipa_unit_id_bts1="${IPA_UNIT_ID_BTS1:-$(( 6000 + op_id * 10 + 2 ))}"
local bsic="${BSIC:-$(( (op_id * 7) % 64 ))}"
# Le side-car a son propre BSIC : c'est lui qui departage deux cellules
# dans les mesures du mobile, et deux cellules du meme operateur qui
# portent le meme code sont indiscernables l'une de l'autre.
local bsic_bts1="${BSIC_BTS1:-$(( (bsic + 1) % 64 ))}"
local lac="${LAC:-0x000${op_id}}"
local cell_id="${CELL_ID:-$(( 6000 + op_id ))}"
# Pas 6000+op_id : le gabarit de la SECONDE BTS
# (configs/osmo-bts-trx-bts1.cfg) porte "ipa unit-id 6002 0" EN DUR. Avec
# l'ancienne formule, l'operateur 2 recevait lui aussi 6002 : ses deux BTS se
# presentaient au BSC sous le meme identifiant, il en acceptait une et
# ejectait l'autre, en boucle. Symptome observe : "BTS failures : 26 OML,
# 26 RSL", un lien OML qui se reconnecte toutes les deux secondes, et aucun
# abonne attache - alors que l'operateur 1 (6001) fonctionnait, par simple
# coincidence de numerotation.
# Le pas de 10 laisse les valeurs de la seconde BTS hors d'atteinte, quel
# que soit le nombre d'operateurs.
local ipa_unit_id="${IPA_UNIT_ID:-$(( 6000 + op_id * 10 ))}"
local bvci="${BVCI:-$(( op_id * 10 + 2 ))}"
local nsei="${NSEI:-$(( op_id * 10 ))}"
local nsvci="${NSVCI:-$(( op_id * 10 ))}"
# Meme plan d'abonne que le HLR (start-docker.sh, section "HLR subscribers") :
#    IMSI = MCC MNC <operateur sur 4 chiffres> <index du MS sur 6>
# Ce gabarit sert le MS #1. L'ancien %010d donnait 0000000002 la ou le HLR
# provisionne 0002000001 : la fiche existait, l'IMSI presente ne la visait
# pas, et l'attachement echouait sans que rien ne dise pourquoi.
local imsi="${IMSI:-${mcc}${mnc}$(printf '%04d%06d' "$op_id" 1)}"
local imei="${IMEI:-$(rand_imei)}"
# IMEI : tire au hasard a chaque generation (voir rand_imei).
# Ki : deux derniers octets = index du MS, puis operateur - l'ordre du HLR
# (printf '...dd%02x%02x' ms_idx op_id). Le "ff" final ne correspondait a
# aucune fiche : sans Ki commun, pas de Kc, et le CIPHER MODE est rejete.
local ki="${KI:-00 11 22 33 44 55 66 77 88 99 aa bb cc dd $(printf '%02x %02x' 1 "$op_id")}"
local sms_sc="${SMS_SC:-+336661234$(printf '%04d' "$op_id")}"

local inter_local_ip sip_remote_ip rtp_start rtp_end sip_host_port
# Idem : sur un noeud de WAN, l'adresse source de l'ASP n'est pas une IP du
# plan docker mais celle du segment - ou 0.0.0.0 quand elle vient du DHCP.
inter_local_ip="${INTER_LOCAL_IP_OVERRIDE:-$(op_backbone_ip "$op_id")}"
# La patte SIP vers Asterisk n'est PAS la dorsale : elle est locale dans les
# deux topologies, et Asterisk ne l'identifie que sur la boucle (pjsip.conf
# [gsm_msc-identify] match=127.0.0.1). Elle avait herite d'inter_local_ip -
# donc de 172.20.0.11, absente de l'ISO - et tout INVITE sortait en
# "Invalid argument (22)" puis 503. Variable distincte, defaut loopback.
sip_remote_ip="${SIP_REMOTE_IP_OVERRIDE:-127.0.0.1}"
rtp_start=$(linphone_rtp_start "$op_id"); rtp_end=$(linphone_rtp_end "$op_id")
sip_host_port=$(linphone_sip_port "$op_id")

# ── Le plan radio, ECRIT, a cote des configurations ──────────────────
# ARFCN, unit-id, PLMN etaient recalcules ailleurs a partir des memes
```

```
# formules - dans start-native.sh pour les mobiles, dans un module de
# qosmo-grgsm pour le side-car. Trois copies d'une meme regle, et la serie de
# pannes qui va avec : un unit-id 6002 fige, un ARFCN 516 fige, des IMSI
# sur le PLMN de l'operateur 1. Chaque fois, les valeurs concordaient pour
# l'operateur 1 et divergeaient pour les suivants.
# Le generateur est le seul a decider. Il ECRIT donc ce qu'il a decide, et
# les autres le LISENT au lieu de le refaire.
{
    printf '# radio-plan.env - ecrit par generate_configs.sh. Ne pas editer.\n'
    printf '# Le plan radio et le PLMN de CET operateur, tels qu ils viennent\n'
    printf '# d etre poses dans les configurations voisines. start-native.sh le\n'
    printf '# lit pour accorder les mobiles et pour piloter le side-car.\n'
    printf 'OPERATOR_ID=%s\n'      "$op_id"
    printf 'PLAN_NODE=%s\n'        "$node"
    printf 'PLAN_MSISDN_PFX=%s\n'  "$msisdn_pfx"
    printf 'PLAN_MCC=%s\n'         "$mcc"
    printf 'PLAN_MNC=%s\n'         "$mnc"
    printf 'PLAN_ARFCN=%s\n'       "$arfcn"
    printf 'PLAN_ARFCN_BTS1=%s\n'  "$arfcn_bts1"
    printf 'PLAN_UNIT_ID=%s\n'     "$ipa_unit_id"
    printf 'PLAN_UNIT_ID_BTS1=%s\n' "$ipa_unit_id_bts1"
    printf 'PLAN_BSIC_BTS1=%s\n'    "$bsic_bts1"
    printf 'PLAN_LAC=%s\n'         "$lac"
    printf 'PLAN_BSIC=%s\n'        "$bsic"
    printf 'PLAN_CELL_ID=%s\n'     "$cell_id"
} > "$dest/osmocom/radio-plan.env"

# extensions.conf porte un #include "annuaire.conf" (les fiches d'abonnes,
# ecrites par start-docker.sh pour que le nom de l'appelant arrive au combine).
# Asterisk avertit a chaque chargement quand un include manque, et un banc
# monte sans start-docker.sh - une VM, start-native.sh seul - n'en a pas. On pose
# donc toujours un fichier, quitte a ce qu'il soit vide : le repli _X. de
# extensions.conf fait le reste, et l'avertissement disparait.
if [ ! -f "$dest/asterisk/annuaire.conf" ]; then
    {
        printf '; annuaire.conf - fiches d abonnes, ecrites par start-docker.sh.\n'
        printf '; Vide ici : ce banc a ete monte sans annuaire. Les appels\n'
        printf '; passent, ils affichent le numero au lieu du nom.\n'
        printf '[sub-annuaire]\n'
    } > "$dest/asterisk/annuaire.conf"
fi

for f in "$dest/osmocom"/*.cfg "$dest/asterisk"/*.conf "$dest/bb"/*.cfg; do
    [ -f "$f" ] || continue
    sed -i \
        -e "s|__ENCRYPTION__|${ENCRYPTION}|g" \
        -e "s|__INTER_NET_GATEWAY__|172.20.0.1|g" \
        -e "s|__CONTAINER_IP__|${container_ip}|g" \
        -e "s|__GATEWAY_IP__|${gateway_ip}|g" \
        -e "s|__HLR_IP__|127.0.0.2|g" \
        -e "s|__INTER_STP_IP__|${inter_stp}|g" \
        -e "s|__INTER_STP_SHUTDOWN__|${inter_stp_shutdown}|g" \
        -e "s|__INTER_LOCAL_IP__|${inter_local_ip}|g" \
        -e "s|__SIP_REMOTE_IP__|${sip_remote_ip}|g" \
        -e "s|__MSISDN_PFX__|${msisdn_pfx}|g" \
        -e "s|__OPERATOR_ID__|${op_id}|g" \
        -e "s|__PC_MSC__|${pc_msc}|g" -e "s|__PC_STP__|${pc_stp}|g" -e "s|__PC_BSC__|${pc_bsc}|g" \
        -e "s|__RCTX_MSC__|${rctx_msc}|g" -e "s|__RCTX_STP__|${rctx_stp}|g" \
        -e "s|__RCTX_BSC__|${rctx_bsc}|g" -e "s|__RCTX_INTER__|${rctx_inter}|g" \
        -e "s|__MCC__|${mcc}|g" -e "s|__MNC__|${mnc}|g" -e "s|__OP_NAME__|${op_name}|g" \
        -e "s|__ARFCN__|${arfcn}|g" -e "s|__IPA_UNIT_ID__|${ipa_unit_id}|g" \
        -e "s|__ARFCN_BTS1__|${arfcn_bts1}|g" \
        -e "s|__IPA_UNIT_ID_BTS1__|${ipa_unit_id_bts1}|g" \
        -e "s|__BSIC_BTS1__|${bsic_bts1}|g" \
        -e "s|__CELL_ID__|${cell_id}|g" -e "s|__BSIC__|${bsic}|g" -e "s|__LAC__|${lac}|g" \
        -e "s|__BVCI__|${bvci}|g" -e "s|__NSEI__|${nsei}|g" -e "s|__NSVCI__|${nsvci}|g" \
```

```
            -e "s|__IMSI__|${imsi}|g" -e "s|__IMEI__|${imei}|g" -e "s|__KI__|${ki}|g" \
            -e "s|__SMS_SC__|${sms_sc}|g" -e "s|__HOST_IP__|${HOST_IP}|g" \
            -e "s|__BAND__|${BAND}|g" -e "s|__SIM_ALGO__|${SIM_ALGO}|g" \
            -e "s|__MS_MAX_POWER__|${MS_MAX_POWER}|g" \
            -e "s|__RXLEV_ACCESS_MIN__|${RXLEV_ACCESS_MIN}|g" \
            -e "s|__CELL_RESEL_HYST__|${CELL_RESEL_HYST}|g" \
            -e "s|__T3109__|${T3109}|g" -e "s|__APN__|${APN}|g" \
            -e "s|__NEIGHBOR_MODE_BTS0__|$([ -n "${MS_MAGNETITE:-}" ] && [ "${MAGNETITE_VOISINS:-0}" != 1 ] &&
             ↪  echo manual || echo automatic)|g" \
            -e "s|__TS1_PCHAN_BTS0__|${TS1_PCHAN:-$([ "${OSMO_TRX:-0}" = 1 ] && echo TCH/F || echo SDCCH8)}|g" \
            -e "s|__SIP_HOST_PORT__|${sip_host_port}|g" \
            -e "s|__ALSA_OUTPUT__|${ALSA_OUTPUT}|g" -e "s|__ALSA_INPUT__|${ALSA_INPUT}|g" \
            -e "s|__RTP_START__|${rtp_start}|g" -e "s|__RTP_END__|${rtp_end}|g" \
            "$f"
    done

    generate_pjsip_interop_trunks     "$op_id" "$n_operators" >> "$dest/asterisk/pjsip.conf"
    generate_extensions_interop_out   "$op_id" "$n_operators" >> "$dest/asterisk/extensions.conf"
    # Routes SMS (fallback = generateur par defaut). Condition d'echec : le
    # MSISDN reel du premier MS absent OU route parasite (op0000/op000).
    # Si non remplie -> regen.
    # Le controle suivait encore "600000 + op*100 + 1" alors que le generateur
    # ecrit desormais <noeud>00<op><ms> : il ne trouvait plus la ligne qu'il
    # venait d'ecrire et regenerait a chaque fois - sans effet visible, mais
    # c'est exactement le genre de test qui finit par masquer une vraie panne.
    _src="$dest/osmocom/sms-routing.conf"
    _generate_sms_routing_conf_fallback "$op_id" "$n_operators" >  "$_src"
    if ! grep -q "^$(osmo_msisdn "$node" "$op_id" 1) = " "$_src" 2>/dev/null \
       || grep -qE "^${op_id}0000? = " "$_src" 2>/dev/null; then
        _generate_sms_routing_conf_fallback "$op_id" "$n_operators" >  "$_src"
    fi

    # ── L'adressage SS7 du noeud : rendu a set-node-id.sh ──────────────────
    # Les gabarits portent les defauts d'une machine SEULE (point-code 1.1.2,
    # ASP vers 127.0.0.1, shutdown). Sur un noeud de WAN ils sont faux - et ils
    # revenaient ici a CHAQUE regeneration : l'adressage pose par set_stp_ip.sh
    # etait efface au demarrage suivant sans que rien ne le signale. L'interco
    # retombait alors que le fichier venait d'etre "regenere proprement".
    #
    # set-node-id.sh (ce que set_stp_ip.sh appelle lui-meme) est le seul endroit
    # qui tienne d'accord les trois fichiers - osmo-stp, osmo-msc, osmo-bsc - et
    # leurs point codes croises. On le rejoue APRES la substitution plutot que
    # de dupliquer ici un sed qui divergerait le jour ou le plan changerait.
    # Son echec est celui de la generation : des configs porteuses du point-code
    # du gabarit ne valent pas mieux que des configs non generees.
    _apply_node_ss7_addressing "$dest" || return 1
}

# Rejoue l'adressage SS7 du noeud sur les configs fraichement generees.
#   $1 = repertoire destination (celui qui contient osmocom/)
# Sans effet - et silencieux - quand la machine n'est pas un noeud de WAN, quand
# le script est absent, ou quand OSMO_NO_STP_IP=1 le demande explicitement.
_apply_node_ss7_addressing() {
    local dest="$1"
    local role_file="${ROLE_FILE:-/etc/osmo-role}"
    local here="${GEN_HERE:-$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)}"

    [ "${OSMO_NO_STP_IP:-0}" = "1" ] && return 0
    [ -d "$dest/osmocom" ] || return 0

    # ── L'ENVIRONNEMENT D'ABORD, le fichier de role ensuite ────────────────
    # Cette fonction s'execute AUSSI dans un conteneur : run.sh (qosmo-grgsm) peut
    # y rejouer apply_config_templates, ce qui regenere tout /etc/osmocom a
    # partir des gabarits - et efface l'identite SS7 que start-docker.sh venait
    # d'ecrire. Le conteneur repartait alors avec le plan du gabarit :
    # point-code 1.1.2 pour TOUS les operateurs, donc deux equipements a la meme
```

```
# adresse SS7 et aucun ASP attache.
#
# Or le conteneur n'a pas de /etc/osmo-role - il a des VARIABLES, que
# start-docker.sh lui passe (OSMO_WAN_NODE, OSMO_HUB_IP). On les lit en premier :
# c'est la seule source disponible la ou le probleme se produit.
local role node hub
node="${OSMO_WAN_NODE:-${WAN_NODE_ID:-}}"
hub="${OSMO_HUB_IP:-}"
role="${OSMO_ROLE:-}"
if [ -r "$role_file" ]; then
    [ -n "$role" ] || role="$(awk -F= '/^OSMO_ROLE=/{gsub(/[ \r\t]/,"",$2);v=$2}    END{print v}'
    ↪  "$role_file")"
    [ -n "$node" ] || node="$(awk -F= '/^OSMO_WAN_NODE=/{gsub(/[ \r\t]/,"",$2);v=$2} END{print v}'
    ↪  "$role_file")"
    [ -n "$hub"  ] || hub="$(awk  -F= '/^OSMO_HUB_IP=/{gsub(/[ \r\t]/,"",$2);v=$2}  END{print v}'
    ↪  "$role_file")"
fi
# UN ROLE EXPLICITE, ET RIEN D'AUTRE.
# Deduire "operateur" de la simple presence d'un numero de noeud etait faux
# sur l'hote : start-docker.sh y porte WAN_NODE_ID = le noeud de la MACHINE, alors
# qu'il genere les configs de CONTENEURS qui sont, eux, d'autres noeuds. Le
# hook reappliquait donc le noeud de l'hote a chaque operateur : tous se
# retrouvaient avec le meme point code, et deux equipements a la meme
# adresse SS7 ne peuvent pas s'attacher tous les deux.
#
# Ce rattrapage ne vaut donc QUE pour une machine qui se declare operateur
# dans /etc/osmo-role (une VM, un noeud natif) ou via OSMO_ROLE. Quand
# start-docker.sh calcule lui-meme l'identite de chaque conteneur, il n'a besoin
# de personne.

# OSMO_WAN_NODE pose dans l'ENVIRONNEMENT vaut declaration lui aussi :
# c'est ce que start-docker.sh passe au conteneur (-e OSMO_WAN_NODE), et un
# conteneur n'a pas de /etc/osmo-role, donc pas de OSMO_ROLE. Avec le seul
# test sur le role, le rattrapage sortait ici sans un mot la ou il est
# justement necessaire, et le conteneur gardait le point-code du gabarit.
# WAN_NODE_ID, lui, ne vaut PAS declaration : sur l'hote il designe le noeud
# de la MACHINE, pas celui du conteneur dont on genere les configs.
[ "$role" = "operator" ] || [ -n "${OSMO_WAN_NODE:-}" ] || return 0
[[ "$node" =~ ^[1-9]$ ]] || return 0

local setid="$here/network/set-node-id.sh"
[ -r "$setid" ] || return 0

# Le hub, quelle qu'en soit la source. Sans lui, set-node-id.sh DEMANDE -
# et comme il est appele ici depuis une boucle de demarrage, il attendait
# une reponse que personne ne venait donner : le lancement se figeait juste
# apres la creation du reseau de l'operateur, sans un mot.
[ -n "$hub" ] || hub="${WAN_STP_TARGET:-${INTER_STP_IP:-}}"

# --op de set-node-id.sh est l'INDICE de l'operateur dans le point code
# 1.<noeud><op>.<role>, pas leur nombre. 1 en dur : un noeud de WAN n'en
# porte qu'un ici. OPS_PER_NODE, lue jusqu'ici, n'est posee nulle part dans
# le depot - elle valait donc toujours 1, mais par accident, et le jour ou
# quelqu'un l'aurait exportee l'identite du noeud aurait change sans raison.
# Ne pas y mettre l'identifiant d'operateur : dans start-docker.sh il vaut l'indice
# du CONTENEUR, et le noeud 2 passerait de 1.21.2 a 1.22.2.
local args=(--node "$node" --op 1 --native --conf-dir "$dest/osmocom")
[ -n "$hub" ] && args+=(--hub-ip "$hub")

# </dev/null : cette reecriture est automatique, elle ne pose JAMAIS de
# question. Si une valeur manque, on prefere un echec signale a un blocage.
if bash "$setid" "${args[@]}" </dev/null >/dev/null 2>&1; then
    echo "  ✓ adressage SS7 du noeud ${node} reapplique (set-node-id.sh)"
else
    echo "  ⚠ adressage SS7 du noeud ${node} NON reapplique - l'ASP inter-STP" >&2
    echo "    gardera les defauts du gabarit (127.0.0.1, shutdown)." >&2
```

```
        # Avertir puis rendre 0, c'etait laisser le demarrage se poursuivre sur
        # des configs dont l'identite SS7 est restee au gabarit : rien en amont
        # ne relit le point code qui vient d'etre ecrit, et la panne ne se
        # voyait qu'a l'ASP qui ne s'attachait jamais.
        return 1
    fi
}

# ── sms-routing.conf : LE ROUTAGE SE LIT DANS LE NUMERO ──────────────────
# La cle de routage est le NOEUD, pas l'operateur, et c'est le premier chiffre
# du MSISDN (<noeud>00<op><ms>). Avant, [operators] portait une entree par
# operateur pointant sur op_backbone_ip - 172.20.0.11, une adresse du plan
# DOCKER absente de toute VM - et l'operateur 1 du noeud 1 comme celui du noeud
# 2 revendiquaient la meme cle "1". Sur un banc a plusieurs noeuds, un SMS
# partait donc chez le premier qui repondait, quand il partait.
#
# Les adresses viennent de la TABLE WAN (/etc/osmo-wan.conf), la seule source
# qui sache ou joignent reellement les autres noeuds. Sans table - un banc a un
# seul noeud - il ne reste qu'une entree, la notre.
#
#   $1 fichier  $2 noeud  $3 operateur  $4 nb de MS  $5 IP locale du coeur
#   $6 table WAN
_write_sms_routing_native() {
    local out="$1" node="$2" op_id="$3" n_ms="$4" host_ip="$5" wanf="$6"
    local spec="" ent id ip nops n o m
    local -a ids=()

    # WAN_NODES="id:ip:indicatif[:operateurs] ..." - lu sans sourcer la table :
    # ce fichier pose aussi WAN_NODE_ID et WAN_OPS, qui ecraseraient ceux de
    # l'appelant.
    if [ -r "$wanf" ]; then
        spec="$(sed -n 's/^WAN_NODES=//p' "$wanf" | tr -d '"'"'"'' | tail -1)"
    fi

    # NOTRE adresse, celle par laquelle les autres noeuds nous joignent. La
    # table d'abord (c'est un choix explicite), la route ensuite, la boucle
    # locale en dernier - un banc isole doit quand meme router ses SMS.
    local self_ip=""
    for ent in $spec; do
        IFS=: read -r id ip _ nops <<< "$ent"
        [ "$id" = "$node" ] && self_ip="$ip"
    done
    if [ -z "$self_ip" ]; then
        self_ip="$(ip route get 1.1.1.1 2>/dev/null | sed -n 's/.*src \([0-9.]*\).*/\1/p' | head -1)"
    fi
    [ -n "$self_ip" ] || self_ip="$host_ip"

    {
        printf '# sms-routing.conf - ecrit par generate_configs.sh.\n'
        printf '# Cle de routage = NUMERO DE NOEUD, premier chiffre du MSISDN.\n'
        printf '# MSISDN = <noeud>00<operateur><rang du MS sur 2 chiffres>\n\n'
        printf '[local]\n'
        printf 'operator_id = %s\n' "$node"
        printf 'sc_address  = 1999001%s444\n\n' "$op_id"
        printf '[operators]\n'
        if [ -n "$spec" ]; then
            for ent in $spec; do
                IFS=: read -r id ip _ nops <<< "$ent"
                [ -n "$id" ] || continue
                [ "$id" = "$node" ] && ip="$self_ip"
                printf '%s = %s\n' "$id" "$ip"
            done
        else
            printf '%s = %s\n' "$node" "$self_ip"
        fi
        printf '\n[routes]\n'
```

```
        if [ -n "$spec" ]; then
            for ent in $spec; do
                IFS=: read -r id ip _ nops <<< "$ent"
                [ -n "$id" ] || continue
                # nops non renseigne = 3 champs dans l'entree. Le "${nops:-1}"
                # d'un case teste la valeur ETENDUE, pas la variable : il ne
                # corrigeait donc rien et seq recevait une borne vide.
                nops="${nops:-1}"
                case "$nops" in ''|*[!0-9]*) nops=1 ;; esac
                for o in $(seq 1 "$nops"); do
                    for m in $(seq 1 "$n_ms"); do
                        printf '%s = %s\n' "$(osmo_msisdn "$id" "$o" "$m")" "$id"
                    done
                done
            done
        else
            for m in $(seq 1 "$n_ms"); do
                printf '%s = %s\n' "$(osmo_msisdn "$node" "$op_id" "$m")" "$node"
            done
        fi
        printf '\n[relay]\nport = 7890\nconnect_timeout = 10\nretry_count = 3\nretry_delay = 5\n'
    } > "$out"
}

# ── Les retouches NATIVES : la recette de l'ISO, en UN SEUL endroit ──────────
# Ce que build-iso.sh posait a la main APRES apply_config_templates, et que
# personne ne rejouait ensuite. C'est la difference entre "les gabarits sont
# substitues" et "la machine a une configuration qui demarre" :
#
#   sms-routing.conf  le fallback des gabarits enumere les operateurs sur
#                     op_backbone_ip (172.20.0.11) - une adresse du plan
#                     DOCKER, qui n'existe sur aucune VM - et n'y met que deux
#                     routes par operateur. En natif tout boucle sur l'adresse
#                     locale du coeur, et il faut UNE route par MS reellement
#                     embarque, sans quoi le relais rend "No route for
#                     destination" des le troisieme.
#   osmo-sgsn.cfg     "gtp local-ip" est fige a 127.0.0.1 dans le gabarit ; en
#                     natif le GGSN ecoute sur l'adresse du coeur. Le SGSN
#                     annoncait une adresse et le GGSN en servait une autre.
#   osmo-msc.cfg      "hlr remote-ip", meme histoire.
#   run.sh            "trxcon <options>" et "mobile <options>" reduits au seul
#                     binaire : sur l'ISO ce sont les modules qui decident des
#                     options, pas la ligne heritee de l'image docker.
#
# Sans cette fonction, une regeneration rendait des configurations SUBSTITUEES
# mais NATIVEMENT FAUSSES : le SGSN se liait a une adresse qu'il n'a pas, le
# relais SMS visait le plan docker - et rien ne le disait avant le demarrage
# suivant. Elle est donc appelee des DEUX cotes, sur la meme arborescence :
#   build-iso.sh        sur $TEMP_CONFIG puis sur $ROOTFS/etc (construction)
#   start-native.sh     sur le repertoire temporaire d'un --regen (execution)
# Idempotente : la rejouer sur un arbre deja retouche ne change rien.
#
#   $1  racine qui porte osmocom/ : $TEMP_CONFIG, $ROOTFS/etc, /etc
#   $2  numero d'operateur      (defaut 1)
#   $3  nombre de MS embarques  (defaut 2)
#   $4  adresse locale du coeur (defaut 127.0.0.2)
#   $5  numero de noeud         (defaut : osmo_node_id)
#   $6  table WAN a lire        (defaut /etc/osmo-wan.conf)
apply_native_post_patches() {
    local dest="${1:?apply_native_post_patches : racine manquante}"
    local op_id="${2:-1}" n_ms="${3:-2}" host_ip="${4:-127.0.0.2}"
    local node="${5:-$(osmo_node_id)}" wanf="${6:-${WAN_CONF_FILE:-/etc/osmo-wan.conf}}"
    # ── DEUX ADRESSES DE PLUS, ET POURQUOI ELLES NE SONT PAS host_ip ─────────
    # sgsn_gtp_ip  Le SGSN et le GGSN ouvrent tous deux un socket GTP. Leur
    #              donner la meme adresse - ce que faisait ce patch, qui ecrivait
    #              host_ip dans « gtp local-ip » alors que le gabarit du GGSN
```

```
#              met ce meme host_ip dans « gtp bind-ip » - donne un GGSN qui
#              prend le port et un SGSN qui meurt sur « Address already in
#              use », relance par systemd sans fin. Le packet attach ne se
#              termine jamais. Par defaut : la passerelle du segment prive
#              du noeud, 192.168.<noeud+1>.1, posee par osmo-ip-plan.sh.
# hlr_ip       « gsup remote-ip » et le « remote-ip » du bloc hlr du MSC
#              designent OU ECOUTE LE HLR, pas l adresse locale du noeud.
#              Les aligner sur host_ip ne se voyait pas tant que les deux
#              valaient 127.0.0.2 ; des que host_ip bouge, le MSC et le SGSN
#              appellent une adresse ou personne ne repond. On suit donc le
#              « bind ip » d osmo-hlr.cfg.
local sgsn_gtp_ip="${7:-}"
local hlr_ip="${8:-127.0.0.2}"
local cfg="$dest/osmocom"
[ -d "$cfg" ] || return 0
[ "$n_ms" -ge 1 ] 2>/dev/null || n_ms=1
case "$node" in [1-9]) ;; *) node=1 ;; esac
# APRES la validation du noeud, jamais avant : un $node vide donnerait
# 192.168.1.1 - le LAN du banc, ou vivent deja les VM et le hub SS7.
: "${sgsn_gtp_ip:=192.168.$((node + 1)).1}"

_write_sms_routing_native "$cfg/sms-routing.conf" \
    "$node" "$op_id" "$n_ms" "$host_ip" "$wanf"

if [ -f "$cfg/osmo-sgsn.cfg" ]; then
    sed -i \
        -e "s/^\([[:space:]]*gtp[[:space:]]\+local-ip[[:space:]]\+\).*/\1${sgsn_gtp_ip}/" \
        -e "s/^\([[:space:]]*gsup[[:space:]]\+remote-ip[[:space:]]\+\).*/\1${hlr_ip}/" \
        -e "/^[[:space:]]*bind udp local\$/,/^[[:space:]]*!\$/
        ↪  s/^\([[:space:]]*listen[[:space:]]\+\).*/\1${host_ip} 23000/" \
        "$cfg/osmo-sgsn.cfg"
fi

if [ -f "$cfg/osmo-msc.cfg" ]; then
    sed -i \
        -e "/^hlr\$/,/^!\$/ s/^\([[:space:]]*remote-ip[[:space:]]\+\).*/\1${hlr_ip}/" \
        "$cfg/osmo-msc.cfg"
fi

# ── osmo-stp.cfg : LE local-ip DE L ASP VERS L INTER-STP ──────────────
# [2026-08-31] Ce fichier manquait a la liste, et c'est une adresse de
# CONTENEUR qui restait dans une config d'HOTE.
#
# Le gabarit pose  local-ip __INTER_LOCAL_IP__  sous  asp asp-to-inter,
# substitue par op_backbone_ip(N) = 172.20.0.<10+N>, soit 172.20.0.11 pour
# l'operateur 1. C'est juste DANS le conteneur, qui porte cette adresse.
# Sur l'hote elle N'EXISTE PAS - build-iso.sh la retire meme explicitement
# ("172.20.0.11 est RETIREE : c'est l'adresse du PREMIER CONTENEUR").
#
# Un local-ip inbindable ne produit pas d'erreur lisible : l'ASP ne monte
# simplement jamais, et le diagnostic sort en aval, tres loin de la cause -
#     as-inter : AS_DOWN
#     Route par defaut -> inter-STP absente ou non avail
# ce qui envoie chercher un probleme de hub ou de filtrage alors que la
# patte locale n'a jamais pu s'ouvrir. Constate le 31/08 sur le banc natif.
#
# Le local-ip vaut l'adresse qui ROUTE vers le hub (ip route get), pas
# 127.0.0.1 : un hub distant (LAN) est injoignable depuis la boucle locale
# (Network is unreachable, l'ASP ne monte jamais). Sans hub distant, on
# garde 127.0.0.1. Le remote-ip, lui, est deja resolu par le gabarit.
#
# awk et pas sed : il faut ne modifier QUE le local-ip qui suit
# « asp asp-to-inter », pas ceux du bloc d'ecoute juste au-dessus
# (local-ip 127.0.0.1 / 127.0.0.2), qui sont corrects.
# [2026-10-15] 127.0.0.1 NE VAUT QUE POUR UN HUB EN 127.0.0.1. Vers un hub
# sur une adresse reseau - meme celle de CETTE machine (192.168.1.31) - une
```

```
# association SCTP liee a 127.0.0.1 est refusee par le noyau : « Network is
# unreachable » toutes les 5 s, as-inter AS_DOWN, barreau ss7 en echec. On
# prend donc l adresse source que le noyau choisit pour joindre le hub
# (ip route get), et 127.0.0.1 seulement si le hub est en boucle locale ou
# si aucune route ne mene a lui.
if [ -f "$cfg/osmo-stp.cfg" ]; then
    local inter_rip inter_lip=127.0.0.1
    inter_rip="$(awk '/^[[:space:]]*asp[[:space:]]+asp-to-inter[[:space:]]/{f=1;next}
     ↪  f&&/^[[:space:]]*remote-ip[[:space:]]/{print $2; exit} f&&/^ (asp|as) /{exit}' "$cfg/osmo-stp.cfg")"
    case "$inter_rip" in
        ""|127.*|__*) ;;   # __* : gabarit pas encore substitue
        *) inter_lip="$(ip -4 route get "$inter_rip" 2>/dev/null | awk '{for(i=1;i<NF;i++) if($i=="src")
         ↪  {print $(i+1); exit}}')"
           [ -n "$inter_lip" ] || inter_lip=127.0.0.1 ;;
    esac
    awk -v lip="$inter_lip" '
        /^[[:space:]]*asp[[:space:]]+asp-to-inter[[:space:]]/ { in_asp = 1; print; next }
        in_asp && /^[[:space:]]*local-ip[[:space:]]/ {
            sub(/local-ip[[:space:]]+.*/, "local-ip " lip); in_asp = 0; print; next
        }
        /^[[:space:]]*(as|asp|cs7|listen)[[:space:]]/ && !/asp-to-inter/ { in_asp = 0 }
        { print }
    ' "$cfg/osmo-stp.cfg" > "$cfg/osmo-stp.cfg.tmp" \
        && mv -f "$cfg/osmo-stp.cfg.tmp" "$cfg/osmo-stp.cfg"
fi

# ── pjsip.conf : LE BLOC [transport-udp] ──────────────────────────────
# [2026-08-31] Troisieme adresse de conteneur dans une config d'hote, et la
# plus couteuse. configs/pjsip.conf ne porte AUCUN placeholder : le bloc
# transport est fige sur le plan docker -
#     external_media_address=10.2.0.2
#     external_signaling_address=10.2.0.2
#     local_net=172.20.0.0/16
# et generate_configs.sh n'y ajoutait que les trunks interop, jamais le
# transport. Sur un banc natif, 10.2.0.2 n'existe nulle part.
#
# CE QUE CA CASSE. external_* est une option de NAT : pour tout pair HORS
# local_net, Asterisk REECRIT le Contact et le SDP avec cette adresse. Un
# Linphone sur le LAN (192.168.x, hors du 172.20.0.0/16 declare) recevait
# donc "10.2.0.2" et n'avait plus personne a joindre - enregistrement ou
# audio muet, selon le moment. Le trunk Zadarma le montrait en clair :
# il nous interrogeait sur sip:474117@127.0.0.1:5060, un Contact
# inexploitable qu'il avait appris de nous.
#
# LE CORRECTIF : sur un hote plat il n'y a pas de frontiere NAT a declarer.
# On SUPPRIME les external_* - sans eux, pjsip repond avec l'adresse de la
# socket, c'est-a-dire la bonne, par destination - et on declare local_net
# sur tout l'espace prive plutot que sur le seul plan docker. Ce jeu est
# topologie-INDEPENDANT : il reste vrai que l'ISO tourne sur un LAN en
# 192.168.x, en 10.x, derriere docker, ou avec l'APN GSM.
if [ -f "$dest/asterisk/pjsip.conf" ]; then
    awk '
        /^\[transport-udp\]/ { in_t = 1; print; next }
        in_t && /^\[/         { in_t = 0 }
        in_t && /^[[:space:]]*external_(media|signaling)_address[[:space:]]*=/ { next }
        in_t && /^[[:space:]]*local_net[[:space:]]*=/ {
            if (!done) {
                print "local_net=⟨1⟩.⟨2⟩.⟨3⟩.0/⟨4⟩"  ×5
⟨⟩ = (127,0,0,8) (10,0,0,8) (172,16,0,12) (192,168,0,16) (176,16,32,24)
                done = 1
            }
            next
        }
        { print }
    ' "$dest/asterisk/pjsip.conf" > "$dest/asterisk/pjsip.conf.tmp" \
        && mv -f "$dest/asterisk/pjsip.conf.tmp" "$dest/asterisk/pjsip.conf"
```

```
    fi

    # ── mobile.cfg : LA CIBLE GSMTAP ──────────────────────────────────
    # [2026-08-31] Deuxieme adresse de conteneur dans une config d'hote.
    # Le gabarit pose  remote-host __INTER_NET_GATEWAY__  sous  gsmtap,
    # substitue par 172.20.0.1 - la passerelle du bridge docker. Elle n'existe
    # QUE tant qu'un reseau docker est monte : sur un banc natif, ou des que
    # les conteneurs sont arretes, l'adresse disparait et les paquets GSMTAP
    # partent dans le vide.
    # Personne ne le signale : le mobile continue a fonctionner, l'appel passe,
    # et c'est Wireshark qui reste desesperement vide sur udp/4729 - ce qu'on
    # met naturellement sur le compte de la radio ou du filtre de capture.
    # Les autres composants (osmo-bts-trx, osmo-pcu, osmo-bsc) tapent deja sur
    # 127.0.0.1:4729 ; le mobile etait le seul a viser ailleurs.
    if [ -f "$cfg/mobile.cfg" ]; then
        sed -i \
            -e "/^gsmtap\$/,/^!\$/ s/^\([[:space:]]*remote-host[[:space:]]\+\).*/\1127.0.0.1/" \
            "$cfg/mobile.cfg"
    fi

    if [ -f "$cfg/run.sh" ]; then
        sed -i \
            -e 's#^[[:space:]]*\([^[:space:]]*/\)\?trxcon\([[:space:]].*\)\?$#trxcon#' \
            -e 's#^[[:space:]]*\([^[:space:]]*/\)\?mobile\([[:space:]].*\)\?$#mobile#' \
            "$cfg/run.sh"
    fi
    chmod +x "$cfg"/*.sh 2>/dev/null || true
    return 0
}

# ── Affichage de l'effectif ──────────────────────────────────────────
_gc_show() {
    local key def perop comment val src
    printf '\n  %-13s %-26s %s\n' "REGLAGE" "VALEUR EFFECTIVE" "ORIGINE"
    printf '  %-13s %-26s %s\n' "-------" "----------------" "-------"
    while IFS='|' read -r key def perop comment; do
        [ -n "$key" ] && [ "$key" != "#SECTION" ] || continue
        val="${!key:-}"
        if [ -n "$val" ]; then src="globals.conf / env"
        elif [ "$perop" = 1 ]; then src="derive de l'operateur"; val="(auto)"
        else src="defaut"; val="$def"; fi
        printf '  %-13s %-26s %s\n' "$key" "$val" "$src"
    done < <(_gc_table)
    printf '\n'
}

# ── Garde-fou multi-operateur ────────────────────────────────────────
_gc_warn_perop() {
    [ "${N_OPERATORS:-1}" -gt 1 ] 2>/dev/null || return 0
    local key def perop comment figes=""
    while IFS='|' read -r key def perop comment; do
        [ "$key" != "#SECTION" ] && [ "$perop" = 1 ] && [ -n "${!key:-}" ] && figes="$figes $key"
    done < <(_gc_table)
    [ -n "$figes" ] || return 0
    printf '\n  ⚠ N_OPERATORS=%s mais ces champs par-operateur sont FIGES :%s\n' \
           "$N_OPERATORS" "$figes"
    printf '    Tous les operateurs partageront la meme valeur - collision garantie\n'
    printf '    sur la radio et/ou les identites. Videz-les dans globals.conf.\n\n'
}

# ── Mode execute ─────────────────────────────────────────────────────
if [ "${BASH_SOURCE[0]}" = "$0" ]; then
    force=0; show=0; _gc_sets=()
    for a in "$@"; do
        case "$a" in
            --force) force=1 ;;
```

```
            --show)  show=1 ;;
            -h|--help)
                sed -n '2,60p' "$0" | sed 's/^#\{0,1\} \{0,1\}//'; exit 0 ;;
            *=*)     _gc_sets+=("$a") ;;
            *) printf 'option inconnue : %s (voir --help)\n' "$a" >&2; exit 2 ;;
        esac
    done
    _gc_write "$force"
    for kv in "${_gc_sets[@]}"; do _gc_set "$kv"; done
    gc_load
    _gc_warn_perop
    [ "$show" = 1 ] && _gc_show
    exit 0
fi

# ── Mode source ──────────────────────────────────────────────────────
gc_load
```

3.14 /opt/GSM/osmo-operator/globals.conf

15810 octets, 453 lignes → 453 lignes

```
# ═══════════════════════════════════════════════════════════════
#  globals.conf — la configuration du réseau, en un seul endroit.
# ═══════════════════════════════════════════════════════════════
#
#  Édite ce fichier à la main : il ne sera JAMAIS réécrit par un run.
#  Seul « ./generate_configs.sh --force » le régénère.
#
#  CE FICHIER FAIT AUTORITÉ — SUR LES VARIABLES QU'IL DÉCLARE, ET SUR
#  ELLES SEULES. Il est lu en dernier : les 26 réglages ci-dessous
#  écrasent leur homonyme dans l'environnement. Tout le reste (CALYPSO_*,
#  LOG_DIR, MODE…) passe au travers, intact. Deux façons de le modifier :
#      vim globals.conf
#      ./generate_configs.sh ARFCN=520
#
#  Un champ VIDE = valeur calculée automatiquement. Les champs notés
#  [auto/opérateur] se déduisent du numéro d'opérateur : ne les fige
#  que si N_OPERATORS=1, sinon tous les opérateurs se marchent dessus.
# ═══════════════════════════════════════════════════════════════

# ── Identité du réseau ──────────────────────────────────────────

# Mobile Country Code (3 chiffres)
MCC="001"
# Mobile Network Code (2 chiffres)
MNC="01"
# Nom court et long du réseau
OP_NAME="OsmoQEMU"

# ── Sécurité et carte SIM ───────────────────────────────────────

# Chiffrement : « a5 0 » (aucun), « a5 1 », « a5 0 1 3 »…
ENCRYPTION="a5 1"
# Algorithme d'authentification (comp128, xor…)
SIM_ALGO="comp128"
# Ki de la SIM, 16 octets hexa espacés  [auto/opérateur]
KI=""
# IMSI du mobile  [auto/opérateur]
IMSI=""
# IMEI du mobile  [auto/opérateur]
IMEI=""

# ── Radio ───────────────────────────────────────────────────────
```

```
# Canal radio (à accorder avec BAND)  [auto/opérateur]
ARFCN="1"
# Bande de la BTS n°1 SEULEMENT (la BTS 0 est en GSM900 dans osmo-bsc.cfg) : GSM900, DCS1800, PCS1900, GSM850
# Doit correspondre a l ARFCN de la BTS n°1 (614 par defaut, DCS1800) sinon le BSC refuse son OML.
BAND="DCS1800"
# Base Station Identity Code, 0..63  [auto/opérateur]
BSIC=""
# Location Area Code  [auto/opérateur]
LAC=""
# Cell identity  [auto/opérateur]
CELL_ID=""
# Identifiant IPA du BTS  [auto/opérateur]
IPA_UNIT_ID=""
# Puissance max autorisée au mobile (0..15)
MS_MAX_POWER="15"
# Niveau reçu minimal pour accéder à la cellule
RXLEV_ACCESS_MIN="0"
# Hystérésis de resélection de cellule (dB)
CELL_RESEL_HYST="4"
# Timer T3109 du BSC (s) — libération de canal
T3109="5"

# ── GPRS ──────────────────────────────────────────────────

# BSSGP Virtual Connection Id  [auto/opérateur]
BVCI=""
# Network Service Entity Id  [auto/opérateur]
NSEI=""
# NS Virtual Connection Id  [auto/opérateur]
NSVCI=""
# Nom du point d'accès
APN="internet"

# ── Services ──────────────────────────────────────────────

# Numéro du centre SMS  [auto/opérateur]
SMS_SC=""

# ── Dimensionnement ───────────────────────────────────────

# Nombre de mobiles simulés
MS_COUNT="2"
# Nombre d'opérateurs (mode « virtual »)
N_OPERATORS="1"
# Adresse de l'hôte vue par la pile
HOST_IP="127.0.0.1"

# >>> VARIABLES DU LANCEUR : bloc genere par tools/globals-complet.py (ne pas editer les marqueurs) >>>
#
#  Valeur VIDE = defaut du script (en commentaire). Regenerer : python3 tools/globals-complet.py
#

# ── startup_scripts/01-defauts-options.sh ─────────────────

# defaut : /etc/osmocom   (01-defauts-options.sh:14)
#OSMOCOM_CFG=""
# defaut : faketrx-qemu   (01-defauts-options.sh:18)
#CALYPSO_PROFILE=""
# defaut : 0   (01-defauts-options.sh:25)
#SAP_MODE=""
# defaut : auto   (01-defauts-options.sh:26)
#SAP_TYPE=""
# defaut : /tmp/osmocom_sap   (01-defauts-options.sh:27)
#SAP_SOCK=""
# defaut : 1   (01-defauts-options.sh:80)
```

```
#DSP_MODE=""
# defaut : /opt/GSM/firmware/magnetite/c139-flash.img   (01-defauts-options.sh:327)
#MAGNETITE_FLASH=""
# defaut : 0   (01-defauts-options.sh:401)
#BTS_CLASSIC=""
# defaut : (vide)   (01-defauts-options.sh:414)
#OSMO_TRX_TS=""
# defaut : 0   (01-defauts-options.sh:421)
#OSMO_TRX_TS1=""
# defaut : 3   (01-defauts-options.sh:422)
#OSMO_TRX_TS1_INSTANCE=""
# defaut : 0   (01-defauts-options.sh:423)
#OSMO_TRX_TCH=""
# defaut : 0   (01-defauts-options.sh:434)
#MODEM_GPRS=""
# defaut : ${OSMO_WAN_NODE:-}   (01-defauts-options.sh:497)
#WAN_NODE_ID=""
# defaut : (vide)   (01-defauts-options.sh:497)
#OSMO_WAN_NODE=""
# defaut : /etc/osmo-wan.conf   (01-defauts-options.sh:507)
#WAN_CONF_FILE=""

# ── startup_scripts/02-noeud-couleurs.sh ──────────────────────

# defaut : ${INTER_STP_IP:-}   (02-noeud-couleurs.sh:19)
#OSMO_HUB_IP=""
# defaut : (vide)   (02-noeud-couleurs.sh:19)
#INTER_STP_IP=""
# defaut : (vide)   (02-noeud-couleurs.sh:78)
#CALYPSO_PY_BANC_EXTRA=""

# ── startup_scripts/04-environnement.sh ───────────────────────

# defaut : /opt/GSM   (04-environnement.sh:12)
#GSM_ROOT=""
# defaut : qosmo   (04-environnement.sh:17)
#CALYPSO_FORK=""
# defaut : /run/osmo-direct   (04-environnement.sh:25)
#RUN_DIR=""
# defaut : $RUN_DIR/logs   (04-environnement.sh:28)
#LOG_DIR=""

# ── startup_scripts/05-run-sh.sh ──────────────────────────────

# defaut : $GSM_ROOT/c54x_exe/run.sh   (05-run-sh.sh:25)
#BANC_DSP=""
# defaut : none   (05-run-sh.sh:34)
#CALYPSO_BRIDGE=""

# ── startup_scripts/06-banc-dsp-menu.sh ───────────────────────

# defaut : /usr/local/bin/${CALYPSO_FORK:-qosmo}   (06-banc-dsp-menu.sh:58)
#QOSMO_LAUNCHER=""
# defaut : 0   (06-banc-dsp-menu.sh:83)
#MENU_MODE=""
# defaut : 0   (06-banc-dsp-menu.sh:84)
#NO_MENU=""
# defaut : 1   (06-banc-dsp-menu.sh:192)
#NODE_ID=""
# defaut : 1   (06-banc-dsp-menu.sh:193)
#NODE_OP=""
# defaut : /etc/osmo-role   (06-banc-dsp-menu.sh:196)
#ROLE_FILE=""
# defaut : 0   (06-banc-dsp-menu.sh:198)
#WAN_MESH=""
# defaut : 0   (06-banc-dsp-menu.sh:203)
```

```
#REGEN_GABARITS=""
# defaut : 0   (06-banc-dsp-menu.sh:221)
#RUN_NO_PROCESS=""
# defaut : mobile   (06-banc-dsp-menu.sh:263)
#CALYPSO_L2_CLIENT=""
# defaut : 320   (06-banc-dsp-menu.sh:282)
#CALYPSO_PULSE_LATENCY_MSEC=""
# defaut : (vide)   (06-banc-dsp-menu.sh:293)
#FIRMWARE_SRC=""
# defaut : $FIRMWARE_SRC   (06-banc-dsp-menu.sh:295)
#FIRMWARE_ELF=""
# defaut : $FIRMWARE_SRC   (06-banc-dsp-menu.sh:297)
#FIRMWARE_DIR=""
# defaut : (vide)   (06-banc-dsp-menu.sh:306)
#FIRMWARE_BIN=""
# defaut : rtp   (06-banc-dsp-menu.sh:443)
#MS_TCH_FORMAT=""
# defaut : 1   (06-banc-dsp-menu.sh:567)
#OPERATOR_ID=""
# defaut : (vide)   (06-banc-dsp-menu.sh:637)
#PLAN_ARFCN=""
# defaut : ${PLAN_ARFCN_BTS1:-$(( 612 + MS_OP_ID * 2 ))}   (06-banc-dsp-menu.sh:648)
#SC_ARFCN=""
# defaut : $(( 612 + MS_OP_ID * 2 ))   (06-banc-dsp-menu.sh:648)
#PLAN_ARFCN_BTS1=""
# defaut : ${PLAN_UNIT_ID_BTS1:-$(( 6000 + MS_OP_ID * 10 + 2 ))}   (06-banc-dsp-menu.sh:649)
#SC_UNIT_ID=""
# defaut : $(( 6000 + MS_OP_ID * 10 + 2 ))   (06-banc-dsp-menu.sh:649)
#PLAN_UNIT_ID_BTS1=""

# ── startup_scripts/08-mobile-ms1.sh ──────────────────────────

# defaut : $BB_DIR/mobile.cfg   (08-mobile-ms1.sh:4)
#MOBILE_CFG_MS1_PATH=""

# ── startup_scripts/09-mobile-cfg-tmux.sh ─────────────────────

# defaut : $HERE/pont/pont.py   (09-mobile-cfg-tmux.sh:76)
#PONT_PY=""
# defaut : ${PLAN_ARFCN:-$MS_ARFCN1}   (09-mobile-cfg-tmux.sh:97)
#PONT_ARFCN=""
# defaut : ${PLAN_BSIC:-$(( MS_OP_ID * 7 % 64 ))}   (09-mobile-cfg-tmux.sh:98)
#PONT_BSIC=""
# defaut : $(( MS_OP_ID * 7 % 64 ))   (09-mobile-cfg-tmux.sh:98)
#PLAN_BSIC=""
# defaut : 0   (09-mobile-cfg-tmux.sh:129)
#CALYPSO_NO_KILLALL=""
# defaut : 0   (09-mobile-cfg-tmux.sh:193)
#AIR_MESH=""
# defaut : ${HERE}/pont/airmesh.py   (09-mobile-cfg-tmux.sh:194)
#AIRMESH_PY=""
# defaut : 2   (09-mobile-cfg-tmux.sh:203)
#N_MS=""
# defaut : 6700   (09-mobile-cfg-tmux.sh:205)
#BB_BASE_PORT=""
# defaut : 3   (09-mobile-cfg-tmux.sh:205)
#BB_PORT_STEP=""
# defaut : (vide)   (09-mobile-cfg-tmux.sh:211)
#WAN_NODES=""
# defaut : 4739   (09-mobile-cfg-tmux.sh:215)
#AIRMESH_PORT=""
# defaut : 0   (09-mobile-cfg-tmux.sh:266)
#OSMO_TRX=""
# defaut : $(dirname "$BANC_DSP")/mobile_pont.cfg   (09-mobile-cfg-tmux.sh:270)
#MOBILE_CFG=""
# defaut : 0   (09-mobile-cfg-tmux.sh:324)
```

```
#CALYPSO_NO_RESTART=""
# defaut : 0   (09-mobile-cfg-tmux.sh:328)
#CALYPSO_NO_ATTACH=""
# defaut : (vide)   (09-mobile-cfg-tmux.sh:332)
#QEMU_ICOUNT=""
# defaut : 0   (09-mobile-cfg-tmux.sh:333)
#ASSEMBLY_LOGS=""
# defaut : in_asm   (09-mobile-cfg-tmux.sh:338)
#ASSEMBLY_LOGS_FLAGS=""
# defaut : (vide)   (09-mobile-cfg-tmux.sh:339)
#ASSEMBLY_LOGS_FILTRE=""

# ── startup_scripts/10-identite-noeud.sh ──────────────────────

# defaut : 1   (10-identite-noeud.sh:73)
#CALYPSO_STOP_KILL_PYTHON=""

# ── startup_scripts/11-regen-configs.sh ──────────────────────

# defaut : 127.0.0.2   (11-regen-configs.sh:13)
#OSMO_NATIVE_CORE_IP=""
# defaut : 127.0.0.1   (11-regen-configs.sh:14)
#OSMO_NATIVE_GW_IP=""

# ── startup_scripts/12-pcap-softsim.sh ──────────────────────

# defaut : 0   (12-pcap-softsim.sh:81)
#PAYLOAD_ISM=""
# defaut : /dev/ttyUSB0   (12-pcap-softsim.sh:88)
#ISM_SERIAL=""
# defaut : 9600   (12-pcap-softsim.sh:97)
#ISM_SERIAL_BAUD=""
# defaut : /tmp/osmo_ism_tx   (12-pcap-softsim.sh:98)
#ISM_TX=""
# defaut : /tmp/osmo_ism_rx   (12-pcap-softsim.sh:98)
#ISM_RX=""
# defaut : 0   (12-pcap-softsim.sh:202)
#WAN_NODE_COUNT=""
# defaut : 0   (12-pcap-softsim.sh:441)
#OSMO_NO_PCAP_RING=""
# defaut : 32   (12-pcap-softsim.sh:504)
#OSMO_PCAP_RING_MB=""
# defaut : 5   (12-pcap-softsim.sh:504)
#OSMO_PCAP_RING_FILES=""

# ── startup_scripts/13-adresses.sh ──────────────────────

# defaut : 0   (13-adresses.sh:3)
#CALYPSO_NO_AUTOSTOP=""
# defaut : 1   (13-adresses.sh:13)
#CALYPSO_STOP_KILL_SOFTSIM=""
# defaut : 0   (13-adresses.sh:88)
#CALYPSO_NO_DASHBOARD=""
# defaut : 1   (13-adresses.sh:144)
#OSMO_RACCORD_MOBILE=""
# defaut : /opt/GSM/pysim-venv   (13-adresses.sh:274)
#PYSIM_VENV=""
# defaut : /opt/GSM/softsim   (13-adresses.sh:299)
#SOFTSIM_DIR=""
# defaut : /tmp/osmo-sap.log   (13-adresses.sh:300)
#SAP_LOG=""
# defaut : 0   (13-adresses.sh:332)
#DSP_BANC=""
# defaut : 3   (13-adresses.sh:360)
#SAP_VERBOSITY=""
```

```
# ── startup_scripts/14-demarrage.sh ──────────────────────────

# defaut : none   (14-demarrage.sh:33)
#IQ=""
# defaut : 120000   (14-demarrage.sh:41)
#INSNS=""
# defaut : 1   (14-demarrage.sh:42)
#LOCKSTEP=""
# defaut : 1   (14-demarrage.sh:43)
#CALYPSO_BSP_STREAM=""
# defaut : 1   (14-demarrage.sh:44)
#CALYPSO_RHEA_DMA_XFER=""
# defaut : rom   (14-demarrage.sh:60)
#BANC_VOCODEUR=""
# defaut : 1   (14-demarrage.sh:61)
#CALYPSO_VSP=""
# defaut : (vide)   (14-demarrage.sh:90)
#MS_MAGNETITE=""
# defaut : 0   (14-demarrage.sh:91)
#MS_AIR=""
# defaut : 150   (14-demarrage.sh:114)
#OSMO_TRX_BTS_ATTENTE=""

# ── /opt/GSM/c54x_exe/run.sh ─────────────────────────────────

# defaut : dsp   (run.sh:52)
#MODE=""
# defaut : 0   (run.sh:53)
#PONT=""
# defaut : /tmp/osmocom_l2_$INSTANCE   (run.sh:68)
#L2_SOCK=""
# defaut : /tmp/qemu-monitor-pont$INSTANCE.sock   (run.sh:69)
#MONITOR=""
# defaut : /tmp/calypso_dsp$INSTANCE.sock   (run.sh:70)
#DSP_SOCK=""
# defaut : $((1233 + INSTANCE))   (run.sh:71)
#GDB_STUB=""
# defaut : $((44443 + INSTANCE))   (run.sh:72)
#GDB_TELNET=""
# defaut : none   (run.sh:73)
#PANNEAU_LOGS=""
# defaut : $((6702 + 10 * (INSTANCE - 1) + (INSTANCE >= 3 ? 30 : 0)))   (run.sh:76)
#CALYPSO_BSP_PORT=""
# defaut : /tmp/calypso_keys$INSTANCE.sock   (run.sh:77)
#CALYPSO_KEYS_SOCK=""
# defaut : 1   (run.sh:78)
#AIR=""
# defaut : /opt/GSM/osmo-operator/pont/air.py   (run.sh:81)
#AIR_PY=""
# defaut : /opt/GSM/qosmo   (run.sh:82)
#QOSMO=""
# defaut : $QOSMO/build/qemu-system-arm   (run.sh:83)
#QEMU=""
# defaut : (vide)   (run.sh:91)
#FIRMWARE_APP=""
# defaut : 1   (run.sh:108)
#OSMO_TRX_PHONE=""
# defaut : $(awk '/^ *bts 0$/{f=1} f&&/base_station_id_code/{print $2; exit}' /etc/osmocom/osmo-bsc.cfg
 ↪  2>/dev/null)   (run.sh:124)
#OSMO_TRX_BSIC=""
# defaut : /opt/GSM/osmocom-bb-transceiver/src/host/layer23/src/transceiver/transceiver   (run.sh:125)
#TRANSCEIVER=""
# defaut : 0x03   (run.sh:134)
#OSMO_TRX_TS_MASK=""
# defaut : 0x03   (run.sh:135)
#OSMO_TRX_TX_MASK=""
```

```
# defaut : 2   (run.sh:136)
#CALYPSO_TRX_TX_SLOTS=""
# defaut : --no-sync -e 3   (run.sh:146)
#OSMO_TRX_ARGS=""
# defaut : 1   (run.sh:149)
#MONTANT_ROM_TX_CHAQUE=""
# defaut : 1   (run.sh:153)
#CALYPSO_BSP_TRX=""
# defaut : -11   (run.sh:157)
#CALYPSO_BSP_TRX_DELTA=""
# defaut : (vide)   (run.sh:171)
#FIRMWARE_FLASH=""
# defaut : 0   (run.sh:180)
#CALYPSO_PACER_RATTRAPAGE=""
# defaut : 1   (run.sh:181)
#CALYPSO_PONT_ARM_LOCK=""
# defaut : 1   (run.sh:184)
#CALYPSO_BSP_SB_FIN_TRAME=""
# defaut : 1   (run.sh:185)
#CALYPSO_BSP_NB_MARGE=""
# defaut : (vide)   (run.sh:188)
#PONT_CAN_NB_TOA=""
# defaut : 0.5   (run.sh:193)
#CALYPSO_BSP_SB_DEC=""
# defaut : 1   (run.sh:194)
#CALYPSO_BSP_BRUIT_FOND=""
# defaut : $AIR   (run.sh:199)
#CALYPSO_BSP_RATTRAPE=""
# defaut : 1   (run.sh:200)
#CALYPSO_RACH_VERS_AIR=""
# defaut : 0   (run.sh:205)
#PONT_UL_TN_DEFAUT=""
# defaut : 0   (run.sh:209)
#AIR_B_FN_DELTA=""
# defaut : 0   (run.sh:210)
#MONTANT_RACH_DIFFERE=""
# defaut : $( . /etc/osmocom/radio-plan.env 2>/dev/null; echo "${PLAN_ARFCN:-}")   (run.sh:211)
#CALYPSO_FAUX_AIR=""
# defaut : ti925t   (run.sh:213)
#QEMU_CPU=""
# defaut : c139   (run.sh:214)
#CALYPSO_LCD=""
# defaut : /opt/GSM/fc-host-tools/rvinterf/lowlevel/rvinterf   (run.sh:215)
#RVINTERF_BIN=""
# defaut : /opt/GSM/osmocom-bb/src/host/osmocon/osmocon   (run.sh:217)
#OSMOCON=""
# defaut : $(command -v mobile || echo /usr/local/bin/mobile)   (run.sh:218)
#MOBILE=""
# defaut : (vide)   (run.sh:239)
#RT_FIFO=""
# defaut : (vide)   (run.sh:240)
#RT_CPUS=""
# defaut : (vide)   (run.sh:241)
#RT_RENICE_AUTRES=""
# defaut : 1   (run.sh:263)
#GDB=""
# defaut : $QOSMO/tools/gdb-telnet.py   (run.sh:266)
#GDB_TELNET_PY=""
# defaut : -v   (run.sh:278)
#VERB=""
# defaut : 30000   (run.sh:280)
#AMP=""
# defaut : 1   (run.sh:294)
#PONT_AIRREC=""
# defaut : 4194304   (run.sh:385)
#QEMU_LOG_BUFFERED=""
```

```
# defaut : 64   (run.sh:422)
#CALYPSO_PONT_RETRY_DIV=""
# defaut : 10   (run.sh:722)
#JOURNAUX_GARDES=""
# defaut : (vide)   (run.sh:763)
#BRUIT_MODE=""
# defaut : (vide)   (run.sh:764)
#BRUIT_SNR_DB=""
# defaut : 16702   (run.sh:765)
#BRUIT_PORT_DSP=""
# defaut : 14730   (run.sh:766)
#BRUIT_PORT_GRGSM=""
# defaut : $BRUIT_SNR_DB   (run.sh:811)
#CALYPSO_BSP_SNR_DB=""

# <<< FIN DES VARIABLES DU LANCEUR <<<
```

3.15 /opt/GSM/osmo-operator/globals.conf.example

12832 octets, 371 lignes → 371 lignes

```
# ═══════════════════════════════════════════════════════════════
#  globals.conf — la configuration du réseau, en un seul endroit.
# ═══════════════════════════════════════════════════════════════
#
#  Édite ce fichier à la main : il ne sera JAMAIS réécrit par un run.
#  Seul « ./generate_configs.sh --force » le régénère.
#
#  CE FICHIER FAIT AUTORITÉ — SUR LES VARIABLES QU'IL DÉCLARE, ET SUR
#  ELLES SEULES. Il est lu en dernier : les 26 réglages ci-dessous
#  écrasent leur homonyme dans l'environnement. Tout le reste (CALYPSO_*,
#  LOG_DIR, MODE…) passe au travers, intact. Deux façons de le modifier :
#      vim globals.conf
#      ./generate_configs.sh ARFCN=520
#
#  Un champ VIDE = valeur calculée automatiquement. Les champs notés
#  [auto/opérateur] se déduisent du numéro d'opérateur : ne les fige
#  que si N_OPERATORS=1, sinon tous les opérateurs se marchent dessus.
# ═══════════════════════════════════════════════════════════════

# ── Identité du réseau ──────────────────────────────────────────

# Mobile Country Code (3 chiffres)
MCC="001"
# Mobile Network Code (2 chiffres)
MNC="01"
# Nom court et long du réseau
OP_NAME="OsmoQEMU"

# ── Sécurité et carte SIM ──────────────────────────────────────

# Chiffrement : « a5 0 » (aucun), « a5 1 », « a5 0 1 3 »…
# choix : a5 0 | a5 1 | a5 3 | a5 0 1 | a5 0 1 3
ENCRYPTION="a5 1"
# Algorithme d'authentification (comp128, xor…)
# choix : comp128 | comp128v2 | comp128v3 | xor | milenage
SIM_ALGO="comp128"
# Ki de la SIM, 16 octets hexa espacés  [auto/opérateur]
KI=""
# IMSI du mobile  [auto/opérateur]
IMSI=""
# IMEI du mobile  [auto/opérateur]
IMEI=""
```

```
# ── Radio ──────────────────────────────────────────────────

# Canal radio (à accorder avec BAND)  [auto/opérateur]
ARFCN="1"
# Bande de la BTS n°1 SEULEMENT (la BTS 0 est en GSM900 dans osmo-bsc.cfg) : GSM900, DCS1800, PCS1900, GSM850
# Doit correspondre a l ARFCN de la BTS n°1 (614 par defaut, DCS1800) sinon le BSC refuse son OML.
# choix : GSM900 | DCS1800 | PCS1900 | GSM850
BAND="DCS1800"
# Base Station Identity Code, 0..63  [auto/opérateur]
BSIC=""
# Location Area Code  [auto/opérateur]
LAC=""
# Cell identity  [auto/opérateur]
CELL_ID=""
# Identifiant IPA du BTS  [auto/opérateur]
IPA_UNIT_ID=""
# Puissance max autorisée au mobile (0..15)
MS_MAX_POWER="15"
# Niveau reçu minimal pour accéder à la cellule
RXLEV_ACCESS_MIN="0"
# Hystérésis de resélection de cellule (dB)
CELL_RESEL_HYST="4"
# Timer T3109 du BSC (s) — libération de canal
T3109="5"

# ── GPRS ───────────────────────────────────────────────────

# BSSGP Virtual Connection Id  [auto/opérateur]
BVCI=""
# Network Service Entity Id  [auto/opérateur]
NSEI=""
# NS Virtual Connection Id  [auto/opérateur]
NSVCI=""
# Nom du point d'accès
APN="internet"

# ── Services ───────────────────────────────────────────────

# Numéro du centre SMS  [auto/opérateur]
SMS_SC=""

# ── Dimensionnement ────────────────────────────────────────

# Nombre de mobiles simulés
MS_COUNT="2"
# Nombre d'opérateurs (mode « virtual »)
N_OPERATORS="1"
# Adresse de l'hôte vue par la pile
HOST_IP="127.0.0.1"

# >>> VARIABLES DU LANCEUR : bloc genere par tools/globals-complet.py (ne pas editer les marqueurs) >>>
#
#  Valeur VIDE = defaut du script (en commentaire). Regenerer : python3 tools/globals-complet.py
#

# ── startup_scripts/01-defauts-options.sh ──────────────────

# defaut : /etc/osmocom   (01-defauts-options.sh:14)
OSMOCOM_CFG=""
# defaut : faketrx-qemu   (01-defauts-options.sh:18)
# choix : faketrx-qemu | faketrx | qemu | noproc
CALYPSO_PROFILE=""
# defaut : 0   (01-defauts-options.sh:25)
# choix : 0 | 1
SAP_MODE=""
# defaut : auto   (01-defauts-options.sh:26)
```

```
# choix : auto | pcsc | softsim
SAP_TYPE=""
# defaut : /tmp/osmocom_sap   (01-defauts-options.sh:27)
SAP_SOCK=""
# defaut : 1   (01-defauts-options.sh:80)
# choix : 0 | 1
DSP_MODE=""
# defaut : /opt/GSM/firmware/magnetite/c139-flash.img   (01-defauts-options.sh:327)
MAGNETITE_FLASH=""
# defaut : 0   (01-defauts-options.sh:401)
# choix : 0 | 1
BTS_CLASSIC=""
# defaut : (vide)   (01-defauts-options.sh:414)
# choix : 1 | 2 | 2/1
OSMO_TRX_TS=""
# defaut : 0   (01-defauts-options.sh:421)
# choix : 0 | 1
OSMO_TRX_TS1=""
# defaut : 3   (01-defauts-options.sh:422)
OSMO_TRX_TS1_INSTANCE=""
# defaut : 0   (01-defauts-options.sh:423)
# choix : 0 | 1
OSMO_TRX_TCH=""
# defaut : 0   (01-defauts-options.sh:434)
# choix : 0 | 1
MODEM_GPRS=""
# defaut : ${OSMO_WAN_NODE:-}   (01-defauts-options.sh:497)
# choix : 0 | 1
WAN_NODE_ID=""
# defaut : (vide)   (01-defauts-options.sh:497)
OSMO_WAN_NODE=""
# defaut : /etc/osmo-wan.conf   (01-defauts-options.sh:507)
WAN_CONF_FILE=""

# ── startup_scripts/02-noeud-couleurs.sh ──────────────────────

# defaut : ${INTER_STP_IP:-}   (02-noeud-couleurs.sh:19)
OSMO_HUB_IP=""
# defaut : (vide)   (02-noeud-couleurs.sh:19)
INTER_STP_IP=""
# defaut : (vide)   (02-noeud-couleurs.sh:78)
CALYPSO_PY_BANC_EXTRA=""

# ── startup_scripts/04-environnement.sh ───────────────────────

# defaut : /opt/GSM   (04-environnement.sh:12)
GSM_ROOT=""
# defaut : qosmo   (04-environnement.sh:17)
CALYPSO_FORK=""
# defaut : /run/osmo-direct   (04-environnement.sh:25)
RUN_DIR=""
# defaut : $RUN_DIR/logs   (04-environnement.sh:28)
LOG_DIR=""

# ── startup_scripts/05-run-sh.sh ──────────────────────────────

# defaut : $GSM_ROOT/c54x_exe/run.sh   (05-run-sh.sh:25)
BANC_DSP=""
# defaut : none   (05-run-sh.sh:34)
CALYPSO_BRIDGE=""

# ── startup_scripts/06-banc-dsp-menu.sh ───────────────────────

# defaut : /usr/local/bin/${CALYPSO_FORK:-qosmo}   (06-banc-dsp-menu.sh:58)
QOSMO_LAUNCHER=""
# defaut : 0   (06-banc-dsp-menu.sh:83)
```

```
# choix : 0 | 1
MENU_MODE=""
# defaut : 0   (06-banc-dsp-menu.sh:84)
# choix : 0 | 1
NO_MENU=""
# defaut : 1   (06-banc-dsp-menu.sh:192)
# choix : 0 | 1
NODE_ID=""
# defaut : 1   (06-banc-dsp-menu.sh:193)
# choix : 0 | 1
NODE_OP=""
# defaut : /etc/osmo-role   (06-banc-dsp-menu.sh:196)
ROLE_FILE=""
# defaut : 0   (06-banc-dsp-menu.sh:198)
# choix : 0 | 1
WAN_MESH=""
# defaut : 0   (06-banc-dsp-menu.sh:203)
# choix : 0 | 1
REGEN_GABARITS=""
# defaut : 0   (06-banc-dsp-menu.sh:221)
# choix : 0 | 1
RUN_NO_PROCESS=""
# defaut : mobile   (06-banc-dsp-menu.sh:263)
CALYPSO_L2_CLIENT=""
# defaut : 320   (06-banc-dsp-menu.sh:282)
CALYPSO_PULSE_LATENCY_MSEC=""
# defaut : (vide)   (06-banc-dsp-menu.sh:293)
FIRMWARE_SRC=""
# defaut : $FIRMWARE_SRC   (06-banc-dsp-menu.sh:295)
FIRMWARE_ELF=""
# defaut : $FIRMWARE_SRC   (06-banc-dsp-menu.sh:297)
FIRMWARE_DIR=""
# defaut : (vide)   (06-banc-dsp-menu.sh:306)
FIRMWARE_BIN=""
# defaut : rtp   (06-banc-dsp-menu.sh:443)
MS_TCH_FORMAT=""
# defaut : 1   (06-banc-dsp-menu.sh:567)
# choix : 0 | 1
OPERATOR_ID=""
# defaut : (vide)   (06-banc-dsp-menu.sh:637)
PLAN_ARFCN=""
# defaut : ${PLAN_ARFCN_BTS1:-$(( 612 + MS_OP_ID * 2 ))}   (06-banc-dsp-menu.sh:648)
SC_ARFCN=""
# defaut : $(( 612 + MS_OP_ID * 2 ))   (06-banc-dsp-menu.sh:648)
PLAN_ARFCN_BTS1=""
# defaut : ${PLAN_UNIT_ID_BTS1:-$(( 6000 + MS_OP_ID * 10 + 2 ))}   (06-banc-dsp-menu.sh:649)
SC_UNIT_ID=""
# defaut : $(( 6000 + MS_OP_ID * 10 + 2 ))   (06-banc-dsp-menu.sh:649)
PLAN_UNIT_ID_BTS1=""

# ── startup_scripts/08-mobile-ms1.sh ──────────────────────────

# defaut : $BB_DIR/mobile.cfg   (08-mobile-ms1.sh:4)
MOBILE_CFG_MS1_PATH=""

# ── startup_scripts/09-mobile-cfg-tmux.sh ─────────────────────

# defaut : $HERE/pont/pont.py   (09-mobile-cfg-tmux.sh:76)
PONT_PY=""
# defaut : ${PLAN_ARFCN:-$MS_ARFCN1}   (09-mobile-cfg-tmux.sh:97)
PONT_ARFCN=""
# defaut : ${PLAN_BSIC:-$(( MS_OP_ID * 7 % 64 ))}   (09-mobile-cfg-tmux.sh:98)
PONT_BSIC=""
# defaut : $(( MS_OP_ID * 7 % 64 ))   (09-mobile-cfg-tmux.sh:98)
PLAN_BSIC=""
# defaut : 0   (09-mobile-cfg-tmux.sh:129)
```

```
# choix : 0 | 1
CALYPSO_NO_KILLALL=""
# defaut : 0   (09-mobile-cfg-tmux.sh:193)
# choix : 0 | 1
AIR_MESH=""
# defaut : ${HERE}/pont/airmesh.py   (09-mobile-cfg-tmux.sh:194)
AIRMESH_PY=""
# defaut : 2   (09-mobile-cfg-tmux.sh:203)
N_MS=""
# defaut : 6700   (09-mobile-cfg-tmux.sh:205)
BB_BASE_PORT=""
# defaut : 3   (09-mobile-cfg-tmux.sh:205)
BB_PORT_STEP=""
# defaut : (vide)   (09-mobile-cfg-tmux.sh:211)
WAN_NODES=""
# defaut : 4739   (09-mobile-cfg-tmux.sh:215)
AIRMESH_PORT=""
# defaut : 0   (09-mobile-cfg-tmux.sh:266)
# choix : 0 | 1
OSMO_TRX=""
# defaut : $(dirname "$BANC_DSP")/mobile_pont.cfg   (09-mobile-cfg-tmux.sh:270)
MOBILE_CFG=""
# defaut : 0   (09-mobile-cfg-tmux.sh:324)
# choix : 0 | 1
CALYPSO_NO_RESTART=""
# defaut : 0   (09-mobile-cfg-tmux.sh:328)
# choix : 0 | 1
CALYPSO_NO_ATTACH=""
# defaut : (vide)   (09-mobile-cfg-tmux.sh:332)
QEMU_ICOUNT=""
# defaut : 0   (09-mobile-cfg-tmux.sh:333)
# choix : 0 | 1
ASSEMBLY_LOGS=""
# defaut : in_asm   (09-mobile-cfg-tmux.sh:338)
ASSEMBLY_LOGS_FLAGS=""
# defaut : (vide)   (09-mobile-cfg-tmux.sh:339)
ASSEMBLY_LOGS_FILTRE=""

# ── startup_scripts/10-identite-noeud.sh ──────────────────────

# defaut : 1   (10-identite-noeud.sh:73)
# choix : 0 | 1
CALYPSO_STOP_KILL_PYTHON=""

# ── startup_scripts/11-regen-configs.sh ──────────────────────

# defaut : 127.0.0.2   (11-regen-configs.sh:13)
OSMO_NATIVE_CORE_IP=""
# defaut : 127.0.0.1   (11-regen-configs.sh:14)
OSMO_NATIVE_GW_IP=""

# ── startup_scripts/12-pcap-softsim.sh ──────────────────────

# defaut : 0   (12-pcap-softsim.sh:81)
# choix : 0 | 1
PAYLOAD_ISM=""
# defaut : /dev/ttyUSB0   (12-pcap-softsim.sh:88)
ISM_SERIAL=""
# defaut : 9600   (12-pcap-softsim.sh:97)
ISM_SERIAL_BAUD=""
# defaut : /tmp/osmo_ism_tx   (12-pcap-softsim.sh:98)
ISM_TX=""
# defaut : /tmp/osmo_ism_rx   (12-pcap-softsim.sh:98)
ISM_RX=""
# defaut : 0   (12-pcap-softsim.sh:202)
# choix : 0 | 1
```

```
WAN_NODE_COUNT=""
# defaut : 0   (12-pcap-softsim.sh:441)
# choix : 0 | 1
OSMO_NO_PCAP_RING=""
# defaut : 32   (12-pcap-softsim.sh:504)
OSMO_PCAP_RING_MB=""
# defaut : 5   (12-pcap-softsim.sh:504)
OSMO_PCAP_RING_FILES=""

# ── startup_scripts/13-adresses.sh ──────────────────────────

# defaut : 0   (13-adresses.sh:3)
# choix : 0 | 1
CALYPSO_NO_AUTOSTOP=""
# defaut : 1   (13-adresses.sh:13)
# choix : 0 | 1
CALYPSO_STOP_KILL_SOFTSIM=""
# defaut : 0   (13-adresses.sh:88)
# choix : 0 | 1
CALYPSO_NO_DASHBOARD=""
# defaut : 1   (13-adresses.sh:144)
# choix : 0 | 1
OSMO_RACCORD_MOBILE=""
# defaut : /opt/GSM/pysim-venv   (13-adresses.sh:274)
PYSIM_VENV=""
# defaut : /opt/GSM/softsim   (13-adresses.sh:299)
SOFTSIM_DIR=""
# defaut : /tmp/osmo-sap.log   (13-adresses.sh:300)
SAP_LOG=""
# defaut : 0   (13-adresses.sh:332)
# choix : 0 | 1
DSP_BANC=""
# defaut : 3   (13-adresses.sh:360)
SAP_VERBOSITY=""

# ── startup_scripts/14-demarrage.sh ─────────────────────────

# defaut : none   (14-demarrage.sh:33)
IQ=""
# defaut : 120000   (14-demarrage.sh:41)
INSNS=""
# defaut : 1   (14-demarrage.sh:42)
# choix : 0 | 1
LOCKSTEP=""
# defaut : 1   (14-demarrage.sh:43)
# choix : 0 | 1
CALYPSO_BSP_STREAM=""
# defaut : 1   (14-demarrage.sh:44)
# choix : 0 | 1
CALYPSO_RHEA_DMA_XFER=""
# defaut : rom   (14-demarrage.sh:60)
BANC_VOCODEUR=""
# defaut : 1   (14-demarrage.sh:61)
# choix : 0 | 1
CALYPSO_VSP=""
# defaut : (vide)   (14-demarrage.sh:90)
MS_MAGNETITE=""
# defaut : 0   (14-demarrage.sh:91)
# choix : 0 | 1
MS_AIR=""
# defaut : 150   (14-demarrage.sh:114)
OSMO_TRX_BTS_ATTENTE=""

# <<< FIN DES VARIABLES DU LANCEUR <<<
```

3.16 /opt/GSM/osmo-operator/hellogist

445 octets, 1 lignes → 1 lignes

```
UA="Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"; H=(-H "Accept: text/html,/;q=0.8" -H
↪  "Accept-Language: fr,en;q=0.7" -H "Referer: https://www.gocomics.com/"); curl -sL -A "$UA" "${H[@]}"
↪  "https://www.gocomics.com/calvinandhobbes/$(date +%Y/%m/%d)" | grep -oP '<meta property="og:image"
↪  content="\K[^"]+' | head -1 | xargs -r curl -sL -A "$UA" "${H[@]}" -o "calvin_$(date +%F).gif" && xdg-open
↪  "calvin_$(date +%F).gif"
```

3.17 /opt/GSM/osmo-operator/install.sh

8534 octets, 192 lignes → 192 lignes

```
#!/bin/bash
# ============================================================================
#  install.sh - installation native, par modules
# ============================================================================
#
#  Meme architecture que run.sh : ce fichier n'est qu'un moteur. Chaque etape
#  vit dans install_modules/NN-<slug>.sh et respecte le contrat decrit dans
#  install_modules/_lib/inst.sh.
#
#      sudo ./install.sh               tout
#      ./install.sh --list             les etapes, sans rien faire
#      ./install.sh --dry-run          deroule sans effet de bord
#      sudo ./install.sh --only deps   une seule etape
#      sudo ./install.sh --skip build  tout sauf celle-la
#      ./install.sh --check            ce qui est deja installe
#      sudo ./install.sh --reinstall   rejoue TOUT, meme ce qui est deja la
#                                      (sources mises a jour, configs et
#                                      raccourcis reecrits, binaires refaits)
#
#  Ou : GSM_ROOT (defaut /opt/GSM) recoit sources et binaires.
#  Le bureau (icones, raccourcis "Lancer le banc GSM" et "multi-operator") est
#  l etape `bureau` : elle pose les MEMES fichiers que l ISO (data/desktop/).
#
#  SI VOUS DECOUVREZ LE PROJET, PREFEREZ DOCKER : `./tools/make-docker-image.sh` produit une image
#  complete et reproductible. Cette installation native s'adresse a ceux qui
#  veulent travailler sur les sources.
#
#  La liste des paquets n'est pas dupliquee ici : elle est EXTRAITE du
#  Dockerfile, qui reste la reference. Les deux ne peuvent donc pas diverger.
# ----------------------------------------------------------------------------
set -uo pipefail

INST_TREE="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
MODDIR="$INST_TREE/install_modules"
: "${GSM_ROOT:=/opt/GSM}"
: "${OSMOCOM_HOME:=$HOME/.osmocom}"
export INST_TREE GSM_ROOT OSMOCOM_HOME

DRY=0 ONLY="" SKIP="" ACTION=install REINSTALL=0
while [ $# -gt 0 ]; do
    case "$1" in
        --list)    ACTION=list ;;
        --check)   ACTION=check ;;
        --dry-run) DRY=1 ;;
        --reinstall) REINSTALL=1 ;;
        --only)    ONLY="${2:-}"; shift ;;
        --skip)    SKIP="${2:-}"; shift ;;
        -h|--help) sed -n '3,24p' "$0" | sed 's/^#\{1,2\} \{0,1\}//'; exit 0 ;;
        *) printf 'option inconnue : %s\n' "$1" >&2; exit 2 ;;
    esac
```

```
    shift
done

if [ -t 1 ] && [ -z "${NO_COLOR:-}" ]; then
    TTY=1; C_OK=$'\033[32m'; C_KO=$'\033[31m'; C_SK=$'\033[33m'; C_DIM=$'\033[2m'; C_Z=$'\033[0m'
else
    TTY=0; C_OK=""; C_KO=""; C_SK=""; C_DIM=""; C_Z=""
fi
begin() { if [ $TTY -eq 1 ]; then printf '[ %s..%s ] %s' "$C_DIM" "$C_Z" "$1"; else printf '[ .. ] %s\n' "$1"; fi; }
end()   { [ $TTY -eq 1 ] && printf '\r\033[K'; printf '[%s%s%s] %s' "$2" "$1" "$C_Z" "$3"
          [ -n "${4:-}" ] && printf ' %s(%s)%s' "$C_DIM" "$4" "$C_Z"; printf '\n'; }

LOGDIR="${LOG_DIR:-/tmp/osmo-nitb-install}"; mkdir -p "$LOGDIR" 2>/dev/null || true
export REINSTALL

[ -d "$MODDIR" ] || { printf 'install_modules/ introuvable\n' >&2; exit 2; }
. "$MODDIR/_lib/inst.sh"
shopt -s nullglob; for f in "$MODDIR"/[0-9][0-9]-*.sh; do . "$f"; done; shopt -u nullglob
[ ${#INST_ORDER[@]} -eq 0 ] && { printf 'aucun module d installation\n' >&2; exit 2; }

in_csv() { case ",$1," in *",$2,"*) return 0;; esac; return 1; }
selected=()
for s in "${INST_ORDER[@]}"; do
    [ -n "$ONLY" ] && { in_csv "$ONLY" "$s" || continue; }
    [ -n "$SKIP" ] && { in_csv "$SKIP" "$s" && continue; }
    selected+=("$s")
done

if [ "$ACTION" = list ]; then
    printf 'Etapes - %d, dans cet ordre\n\n' "${#selected[@]}"
    for s in "${selected[@]}"; do
        printf '  %-10s %-38s %s%s\n' "$s" "${INST_DESC[$s]}" \
            "$([ "${INST_REQUIRED[$s]}" = 1 ] && echo obligatoire || echo optionnelle)" \
            "$([ -n "${INST_DEPS[$s]}" ] && echo "  apres: ${INST_DEPS[$s]}")"
    done
    printf '\nGSM_ROOT = %s\n' "$GSM_ROOT"
    exit 0
fi

if [ "$ACTION" = check ]; then
    rc=0
    for s in "${selected[@]}"; do
        p="$(inst_prefix "$s")"
        if declare -F "${p}_done" >/dev/null; then
            if "${p}_done" >/dev/null 2>&1; then end " OK " "$C_OK" "${INST_DESC[$s]}" "deja installe"
            else end "----" "$C_SK" "${INST_DESC[$s]}" "a faire"; rc=1; fi
        else
            end " ?  " "$C_DIM" "${INST_DESC[$s]}" "pas de controle"
        fi
    done
    exit $rc
fi

# root : exige seulement si une etape retenue le demande
need_root=0
for s in "${selected[@]}"; do [ "${INST_ROOT[$s]}" = 1 ] && need_root=1; done
if [ $DRY -eq 0 ] && [ $need_root -eq 1 ] && [ "$(id -u)" -ne 0 ]; then
    printf 'Cette installation ecrit dans %s, /usr/local et /etc/osmocom.\n' "$GSM_ROOT" >&2
    printf 'Relancez avec sudo, ou inspectez sans rien modifier :  ./install.sh --dry-run\n' >&2
    exit 2
fi

declare -A STATE=()
nb_ok=0 nb_skip=0 nb_fail=0
for s in "${selected[@]}"; do
    p="$(inst_prefix "$s")"; log="$LOGDIR/$s.log"
```

```
_INST_REASON=""; _INST_HINT=""

# Une etape prerequise que l appelant a ECARTEE (--only, --skip) n est pas
# un echec : c est son choix. Seule une etape jouee et RATEE bloque. Sans
# ca, "--only deps" sautait deps parce que prereqs n etait pas dans la
# selection - l exemple de l en-tete ne marchait pas.
dep_ko=""
for d in ${INST_DEPS[$s]}; do
    in_csv "$(IFS=,; echo "${selected[*]}")" "$d" || continue
    case "${STATE[$d]:-absent}" in ok|skip) ;; *) dep_ko="$d"; break;; esac
done
if [ -n "$dep_ko" ]; then
    end "SKIP" "$C_SK" "${INST_DESC[$s]}" "depend de $dep_ko, non satisfaite"
    STATE[$s]=skip; nb_skip=$((nb_skip+1)); continue
fi

begin "${INST_DESC[$s]}"

# --reinstall : on ne demande plus « deja fait ? », on refait. Les modules
# lisent REINSTALL pour ecraser ce qu ils conservent d ordinaire (cp -n,
# depot laisse tel quel).
if [ "$REINSTALL" -eq 0 ] && declare -F "${p}_done" >/dev/null && "${p}_done" >>"$log" 2>&1; then
    end "SKIP" "$C_SK" "${INST_DESC[$s]}" "deja installe"
    STATE[$s]=skip; nb_skip=$((nb_skip+1)); continue
fi

if declare -F "${p}_check" >/dev/null; then
    "${p}_check" >>"$log" 2>&1; rc=$?
    if [ $rc -eq $INST_RC_FAIL ]; then
        end "FAIL" "$C_KO" "${INST_DESC[$s]}" "${_INST_REASON:-prerequis non satisfait}"
        [ -n "$_INST_HINT" ] && printf '       %s→ %s%s\n' "$C_DIM" "$_INST_HINT" "$C_Z"
        printf '       %sjournal : %s%s\n' "$C_DIM" "$log" "$C_Z"
        nb_fail=$((nb_fail+1)); STATE[$s]=fail
        [ "${INST_REQUIRED[$s]}" = 1 ] && { printf '\ninstallation interrompue\n'; exit 1; }
        continue
    fi
fi

if [ $DRY -eq 1 ]; then
    end " -- " "$C_DIM" "${INST_DESC[$s]}" "simule"; STATE[$s]=ok; continue
fi

"${p}_run" >>"$log" 2>&1; rc=$?
case $rc in
    $INST_RC_DONE) end "SKIP" "$C_SK" "${INST_DESC[$s]}" "${_INST_REASON:-rien a faire}"
                   STATE[$s]=skip; nb_skip=$((nb_skip+1)); continue ;;
    $INST_RC_FAIL) end "FAIL" "$C_KO" "${INST_DESC[$s]}" "${_INST_REASON:-echec}"
                   [ -n "$_INST_HINT" ] && printf '       %s→ %s%s\n' "$C_DIM" "$_INST_HINT" "$C_Z"
                   printf '       %sjournal : %s%s\n' "$C_DIM" "$log" "$C_Z"
                   nb_fail=$((nb_fail+1)); STATE[$s]=fail
                   [ "${INST_REQUIRED[$s]}" = 1 ] && { printf '\ninstallation interrompue\n'; exit 1; }
                   continue ;;
esac

# installer n'est pas avoir installe : on controle apres coup
if declare -F "${p}_verify" >/dev/null; then
    if ! "${p}_verify" >>"$log" 2>&1; then
        end "FAIL" "$C_KO" "${INST_DESC[$s]}" "faite, mais le controle echoue : ${_INST_REASON:-}"
        [ -n "$_INST_HINT" ] && printf '       %s→ %s%s\n' "$C_DIM" "$_INST_HINT" "$C_Z"
        printf '       %sjournal : %s%s\n' "$C_DIM" "$log" "$C_Z"
        nb_fail=$((nb_fail+1)); STATE[$s]=fail
        [ "${INST_REQUIRED[$s]}" = 1 ] && { printf '\ninstallation interrompue\n'; exit 1; }
        continue
    fi
fi
```

```
    end " OK " "$C_OK" "${INST_DESC[$s]}"; STATE[$s]=ok; nb_ok=$((nb_ok+1))
done

printf '\n%d ok · %d ignores · %d echecs\n' "$nb_ok" "$nb_skip" "$nb_fail"
if [ $nb_fail -eq 0 ] && [ $DRY -eq 0 ]; then
    printf '\n  %sverifier%s   ./install.sh --check\n'          "$C_DIM" "$C_Z"
    printf '  %slancer%s     ./start-native.sh --list   puis   sudo ./start-native.sh\n\n' "$C_DIM" "$C_Z"
fi
[ $nb_fail -gt 0 ] && exit 1
exit 0
```

3.18 /opt/GSM/osmo-operator/launch.sh

44186 octets, 803 lignes → 803 lignes

```
#!/bin/bash
# ═══════════════════════════════════════════════════════════════════════
# launch.sh - Point d entree DOUBLE-CLIC du banc osmo-operator.
#
# Monte le poste de travail complet en un geste :
#     1. wireshark    sur udp/4729 (GSMTAP)        - root
#     2. linphone     le softphone SIP             - utilisateur de la session
#     3. firefox      sur http://127.0.0.1:8080    - des que le dashboard repond
#     4. start.sh                          - le banc, au PREMIER PLAN
#
# Les trois premiers partent en arriere-plan et ne peuvent pas faire echouer le
# lancement : un outil absent se signale et on continue. start.sh, lui,
# tient le terminal - c est sa sortie qu on veut lire.
#
# Trois choses separent un script qui marche au terminal d un script qui marche
# au double-clic, et elles sont traitees en tete :
#   1. PAS DE TERMINAL. Un .desktop lance sans tty : start.sh ecrirait
#      dans le vide. On se relance dans un emulateur de terminal.
#   2. PAS DE ROOT. Le banc a besoin des privileges (netns, systemd, captures).
#      Depuis une icone : pkexec (fenetre graphique), repli sudo.
#   3. LA FENETRE SE FERME AVANT QU ON LISE L ERREUR. On la retient a la fin.
#
# Usage au terminal (aucun de ces detours ne s applique) :
#     sudo ./launch.sh [arguments passes a start.sh]
# ═══════════════════════════════════════════════════════════════════════
set -uo pipefail

DIR="$(cd "$(dirname "$(readlink -f "$0")")" && pwd)"
TARGET="$DIR/start.sh"
GSMTAP_PORT="${GSMTAP_PORT:-4729}"
DASH_URL="${OSMO_DASH_URL:-http://127.0.0.1:8080}"
DASH_PORT="${DASH_URL##*:}"
# Le tutoriel s ouvre en DEUXIEME ONGLET, a cote du tableau de bord. Servi par
# le dashboard plutot qu en file:// : une seule copie fait foi, celle que le
# dashboard expose, et l URL marche a l identique depuis une autre machine du
# banc. Voir /usr/local/bin/osmo-tutorial.
TUTO_URL="${OSMO_TUTORIAL_URL:-${DASH_URL%/}/tutorial.html}"

GREEN='\033[0;32m'; YELLOW='\033[1;33m'; CYAN='\033[0;36m'; RED='\033[0;31m'; NC='\033[0m'

[ -x "$TARGET" ] || { echo -e "${RED}start.sh introuvable ou non executable dans $DIR${NC}" >&2; exit 1; }

# ── UN SEUL LANCEMENT A LA FOIS ──────────────────────────────────────────
# [2026-09-04] Un double-clic sur le telephone rouge, c est DEUX lancements. Le
# second n attend pas le premier : il commence par tout arreter (un clic = un
# banc neuf) et demonte donc la pile que le premier est en train de monter. On
# obtient un banc a moitie debout, deux journaux entrelaces dans deux fenetres,
# et une panne qui ne ressemble a aucune panne connue - pour un geste qui
# n etait meme pas voulu.
```

```
#
# DEUX FACONS D ETRE DEJA EN TRAIN DE DEMARRER, DEUX TESTS :
#   1. un autre launch.sh tient le verrou. C est un flock non bloquant sur un
#      descripteur qui SURVIT a l exec de start.sh : il couvre donc tout
#      le demarrage, et le noyau le libere quand le processus meurt, meme tue -
#      rien a nettoyer, et surtout aucun verrou fantome apres une coupure.
#   2. l unite osmo-banc est « activating ». Un banc lance par systemd
#      (`systemctl start osmo-banc`, l action « en service » de l icone, ou une
#      image gravee avec --banc) ne passe par aucun launch.sh : il ne tient
#      aucun verrou, et sans ce second test un clic pendant ce demarrage-la
#      passerait au travers.
# `--stop` n est jamais bloque : c est justement le geste qui reprend la main
# quand un lancement s eternise.
LOCK_FILE="${OSMO_LAUNCH_LOCK:-/run/lock/osmo-launch.lock}"
verrou_pris() {
    command -v flock >/dev/null 2>&1 || return 0   # pas de flock : on ne bloque rien
    mkdir -p "$(dirname "$LOCK_FILE")" 2>/dev/null || true
    exec 9>>"$LOCK_FILE" 2>/dev/null || return 0
    flock -n 9
}
banc_en_demarrage() {
    [ "$(systemctl is-active "${OSMO_BANC_SERVICE:-osmo-banc.service}" 2>/dev/null)" = activating ]
}
lancement_possible() {
    banc_en_demarrage && return 1
    verrou_pris
}
# [2026-09-04] LE VERROU DOIT MOURIR AVEC LE DEMARRAGE, PAS AVEC LA FENETRE.
# Tenu par le descripteur 9, il vivait aussi longtemps que le processus - donc
# jusqu a ce que quelqu un appuie sur Entree devant « Fenetre maintenue
# ouverte », et au-dela si la fenetre restait la (ou si on s attachait a tmux,
# l exec gardant le descripteur). Une fenetre oubliee sur un coin de bureau
# suffisait alors a faire repondre « un lancement est deja en cours » a TOUS
# les clics suivants, y compris « Demarrer le banc en service ». On le rend
# donc des que le banc est monte.
verrou_liberer() { exec 9>&- 2>/dev/null || true; }

# ── 0. DEMARRER COMME UN SERVICE (action du clic droit) ──────────────────
# [2026-09-04] L icone n avait que deux gestes : « lancer » (qui ouvre un
# terminal et y deroule tout le demarrage) et « arreter ». Il manquait celui
# qu on veut sur une machine qui tourne toute seule : monter le banc COMME UN
# SERVICE - pas de fenetre a garder ouverte, pas de sortie a surveiller, et un
# banc qui survit a la fermeture de la session.
#
# On parle donc a systemd et a personne d autre, AVANT le bloc terminal
# ci-dessous : cette action ne doit jamais ouvrir de fenetre. Le compte rendu
# passe par une notification de bureau, seul endroit ou l operateur regarde
# quand il n a pas de terminal. `systemctl restart` rend la main a la fin du
# demarrage (jusqu a TimeoutStartSec=900) : rien n attend ici, ni le bureau ni
# l icone - c est ce delai qui permet d annoncer un vrai resultat plutot qu un
# « c est parti » sans suite.
case "${1:-}" in --service|--service-suivi|--stop|--dashboard|--console|--vty|--multi|--multi-suivi|--multi-stop)
 ↪  _bloc_direct=1 ;; *) _bloc_direct=0 ;; esac
if [ "$_bloc_direct" = "1" ]; then
    # Root SANS terminal : pkexec demande le mot de passe dans une fenetre du
    # bureau. On ne retombe PAS sur sudo ici - sudo voudrait un terminal, et
    # c est precisement ce que ces deux actions promettent de ne pas ouvrir.
    if [ "$(id -u)" -ne 0 ] && command -v pkexec >/dev/null 2>&1 \
       && [ -n "${DISPLAY:-}" ]; then
        exec pkexec env DISPLAY="${DISPLAY:-}" XAUTHORITY="${XAUTHORITY:-}" \
             "$0" "$@"
    fi
    _action="$1"; shift
    _svc="${OSMO_BANC_SERVICE:-osmo-banc.service}"
    _icon=/usr/share/osmo-operator/icons/osmo-launch.svg
    _note() {
```

```
    if command -v notify-send >/dev/null 2>&1; then
        notify-send -i "$_icon" "Banc GSM" "$1" 2>/dev/null || true
    fi
    echo -e "  ${CYAN}→${NC} $1"
}

# ── ET LES OPERATEURS EN CONTENEUR ──────────────────────────────────
# [2026-09-04] « Arreter le banc » LAISSAIT LE BANC DEBOUT. Le natif
# s arretait bien, mais les operateurs docker du multi (osmo-operator-2,
# -3, le hub osmo-inter-stp) continuaient de tourner : chacun porte SON
# dashboard, SON asterisk, SON HLR et son propre gapk, dans son propre
# cgroup (/system.slice/docker-<id>.scope) - donc hors de portee de
# `systemctl stop osmo-banc`, quoi que fasse son ExecStop. Le 8080
# repondait encore, les abonnes restaient « rattaches », le telephone
# sonnait toujours : de l exterieur, l arret n avait rien arrete.
#
# osmo-multi.service porte pourtant Requires=osmo-banc.service, et systemd
# propage bien l arret - MAIS SEULEMENT SI CETTE UNITE EST ACTIVE. Tuee en
# cours de demarrage (constate : Main process exited, signal=15/TERM), elle
# passe « failed » : plus rien ne porte les conteneurs, aucun ExecStop ne
# tourne, et ils survivent a tous les arrets suivants. On arrete donc
# l unite quand elle existe, PUIS on balaie ce qui reste.
#
# Le balai n est pas redondant avec `start-multi.sh --stop` : celui-ci ne
# connait que la topologie declaree dans /etc/osmocom/osmo-multi.conf, et
# un conteneur d une topologie precedente lui echapperait. Le filtre est
# celui que start-docker.sh applique deja en tete de course.
_arret_multi() {
    local _m="${OSMO_MULTI_SERVICE:-osmo-multi.service}"
    if systemctl cat "$_m" >/dev/null 2>&1; then
        timeout 200 systemctl stop "$_m" >/dev/null 2>&1 || true
        systemctl reset-failed "$_m" 2>/dev/null || true
    fi
    command -v docker >/dev/null 2>&1 || return 0
    local _c
    _c="$(docker ps -q --filter name=osmo- 2>/dev/null)" || return 0
    [ -n "$_c" ] || return 0
    _note "Arret des operateurs en conteneur..."
    # shellcheck disable=SC2086  # une liste d ID, un par ligne
    docker rm -f $_c >/dev/null 2>&1 || true
}

# ── LE MULTI-OPERATEUR : L ICONE DEMARRE L UNITE, ET LE DIT ─────────
# [2026-09-07] L ICONE « multi-operator » NE DEMARRAIT RIEN, SANS UN MOT.
# Elle appelait directement `pkexec systemctl start osmo-multi.service`, et
# cet appel rend 0 dans DEUX cas qu on ne distingue pas d un clic :
#   1. osmo-multi.service porte deux ConditionPathExists (start-multi.sh et
#      /etc/osmocom/osmo-multi.conf, la topologie qu ecrit addition.sh).
#      Une condition non remplie n est PAS un echec pour systemd : l unite
#      est SAUTEE, `systemctl start` sort en 0, `systemctl --failed` reste
#      vide. Sur une ISO fraiche - ou personne n a encore lance les
#      Supplements - c est le cas NORMAL : le clic ne pouvait rien faire.
#   2. l unite est deja active (RemainAfterExit=yes) : start ne rejoue rien.
# Dans les deux cas, aucune fenetre, aucune notification, rien : l icone
# passait pour morte. On teste donc les conditions AVANT de demarrer, on
# nomme ce qui manque, et on annonce le resultat par notification - comme
# « Demarrer le banc en service » juste au-dessus.
_svcm="${OSMO_MULTI_SERVICE:-osmo-multi.service}"
# ── LE LANCEMENT SUIVI : L ACTION, ET LE JOURNAL DE L UNITE QUI DEFILE ──
# [2026-10-12] Ce que la fenetre de lancement du bureau (osmo-gtk-run.py
# --progress) execute : « systemctl start » ne montre rien, le banc ecrit
# dans le journal de son unite. On le suit en parallele, et la fenetre y
# lit les reperes « [n/10] » de run.sh pour sa barre. Le verrou, les
# notifications, le code de retour : ceux de --service / --multi.
if [ "$_action" = "--service-suivi" ] || [ "$_action" = "--multi-suivi" ]; then
    if [ "$_action" = "--service-suivi" ]; then _u="$_svc"; _sub=--service; else _u="$_svcm"; _sub=--multi; fi
```

```
    journalctl -u "$_u" -f -n 0 -o cat 2>/dev/null & _j=$!
    "$0" "$_sub" "$@"; _rc=$?
    sleep 1; kill "$_j" 2>/dev/null; wait "$_j" 2>/dev/null
    exit "$_rc"
fi
if [ "$_action" = "--multi-stop" ]; then
    _note "Arret du multi-operateur..."
    _arret_multi
    systemctl reset-failed "$_svcm" 2>/dev/null || true
    _note "Multi-operateur arrete (le natif reste a osmo-banc)."
    exit 0
fi
if [ "$_action" = "--multi" ]; then
    if ! systemctl cat "$_svcm" >/dev/null 2>&1; then
        _note "Unite $_svcm absente : lancez d abord l icone « Supplements »."
        exit 1
    fi
    # Les memes chemins que les ConditionPathExists de l unite. On les lit
    # ICI plutot que de croire le code de retour de systemctl.
    if [ ! -f "$DIR/launch/start-multi.sh" ]; then
        _note "start-multi.sh introuvable dans $DIR - depot incomplet."
        exit 1
    fi
    if [ ! -f /etc/osmocom/osmo-multi.conf ]; then
        _note "Pas de topologie multi-operateur (/etc/osmocom/osmo-multi.conf) : l unite serait sautee en
        ↪  silence. Lancez l icone « Supplements » (addition.sh), qui installe docker, l image et cette
        ↪  topologie."
        exit 1
    fi
    case "$(systemctl is-active "$_svcm" 2>/dev/null)" in
        active)     _note "Le multi-operateur tourne deja. Journal : journalctl -u ${_svcm%.service} -f"; exit
        ↪  0 ;;
        activating) _note "Le multi-operateur demarre deja - journalctl -u ${_svcm%.service} -f"; exit 0 ;;
    esac
    systemctl reset-failed "$_svcm" 2>/dev/null || true
    _note "Demarrage de $_svcm - comptez plusieurs minutes (images, conteneurs, HLR)."
    # `start` et non `restart` : ExecStop demonte les conteneurs, et un
    # restart sur une unite deja montee couterait un demontage inutile.
    if systemctl start "$_svcm"; then
        # Type=oneshot + conditions : un 0 ne prouve pas que ca a tourne.
        if [ "$(systemctl is-active "$_svcm" 2>/dev/null)" = active ]; then
            _note "Multi-operateur en service. Journal : journalctl -u ${_svcm%.service} -f"
            exit 0
        fi
        _note "Unite sautee (condition non remplie) - systemctl status ${_svcm%.service}"
        exit 1
    fi
    _note "Echec de $_svcm - journalctl -u ${_svcm%.service} -n 80"
    exit 1
fi

# ── LE TABLEAU DE BORD ──────────────────────────────────────────────
# Un clic, une page. xdg-open passe par le navigateur par defaut, donc par
# la fenetre deja ouverte s il y en a une : un onglet de plus, pas une
# deuxieme fenetre.
if [ "$_action" = "--dashboard" ]; then
    _u="${OSMO_DASH_URL:-http://127.0.0.1:8080}"
    # Firefox EN PREMIER, pas xdg-open : c est le navigateur de l image, et
    # une seconde invocation lui ajoute un onglet au lieu d ouvrir une
    # fenetre de plus. xdg-open ne sert que de repli.
    if command -v firefox >/dev/null 2>&1; then
        setsid firefox "$_u" >/dev/null 2>&1 &
    elif command -v xdg-open >/dev/null 2>&1; then
        setsid xdg-open "$_u" >/dev/null 2>&1 &
    else
        _note "Aucun navigateur - ouvrez $_u a la main."; exit 1
```

```
    fi
    exit 0
fi

# ── LA CONSOLE DU MOBILE (VTY) ──────────────────────────────────────
# telnet, comme le dit la documentation Osmocom : le VTY attend un terminal
# ligne a ligne. Le `read` final garde la fenetre le temps de lire l erreur
# quand le port est ferme.
if [ "$_action" = "--vty" ]; then
    # [2026-10-12] 4247, ou 4347 en --dsp : start.sh publie le port
    # reel dans /run/osmo-banc/ms1-vty ; sans le fichier, on regarde qui
    # ecoute (4347 d abord, puis 4247).
    _p="${OSMO_MS_VTY_PORT:-}"
    [ -z "$_p" ] && [ -r /run/osmo-banc/ms1-vty ] && _p="$(tr -dc 0-9 < /run/osmo-banc/ms1-vty)"
    if [ -z "$_p" ]; then
        if ss -ltnH "( sport = :4347 )" 2>/dev/null | grep -q .; then _p=4347; else _p=4247; fi
    fi
    # [2026-10-01] Dans une fenetre GTK du banc (terminal VTE dedans), plus
    # dans un gnome-terminal : aucun favori n ouvre de terminal.
    _c="telnet 127.0.0.1 $_p || { echo; echo 'VTY 127.0.0.1:$_p injoignable - banc arrete ?'; exit 1; }"
    exec python3 "$(dirname "$(readlink -f "$0")")/tools/osmo-gtk-run.py" --tty \
         --title "Console du mobile (VTY $_p)" -- bash -c "$_c"
    _note "Aucune fenetre possible - tapez : telnet 127.0.0.1 $_p"
    exit 1
fi

# ── LA CONSOLE DU BANC ──────────────────────────────────────────────
# Celle-ci, ET ELLE SEULE, ouvre un terminal : `tmux attach` est une session
# interactive, elle n a nulle part ailleurs ou vivre. Le lancement du banc,
# lui, n en ouvre plus (voir le bloc « UN CLIC = UN SERVICE »).
if [ "$_action" = "--console" ]; then
    _sess="${TMUX_SESSION:-calypso}"
    if ! tmux has-session -t "$_sess" 2>/dev/null; then
        _note "Pas de session « $_sess » : le banc ne tourne pas."
        exit 1
    fi
    # [2026-10-01] Le terminal VTE vit dans une fenetre GTK du banc
    # (tools/osmo-gtk-run.py) : aucun favori n ouvre de terminal.
    exec python3 "$(dirname "$(readlink -f "$0")")/tools/osmo-gtk-run.py" --tty \
         --title "Console du banc (tmux $_sess)" -- tmux attach -t "$_sess"
    _note "Aucune fenetre possible - tapez : tmux attach -t $_sess"
    exit 1
fi

# ── ARRETER ─────────────────────────────────────────────────────────
# [2026-09-04] « Arreter le banc » etait traite APRES le bloc terminal :
# le clic ouvrait donc une fenetre entiere pour y lancer un systemctl stop,
# et sur une machine sans emulateur de terminal il ne se passait
# simplement RIEN. Un arret n a rien a montrer : il se fait ici, avant
# toute fenetre.
if [ "$_action" = "--stop" ]; then
    _etat="$(systemctl is-active "$_svc" 2>/dev/null || true)"
    if systemctl cat "$_svc" >/dev/null 2>&1; then
        _note "Arret de $_svc..."
        timeout 200 systemctl stop "$_svc" || true
    fi
    # ExecStop ne tourne QUE si l unite etait active ou en cours de
    # demarrage. Quand elle est « failed » (start.sh sorti en
    # erreur), systemd a deja tue son cgroup - mais le coeur (osmo-stp,
    # hlr, msc, bsc...) vit dans SES PROPRES unites, que run.sh a
    # demarrees par systemctl : elles restent debout, le telephone du
    # tableau de bord aussi, et l arret semblait ne rien faire. On demonte
    # alors a la main ce que ExecStop n a pas eu l occasion de demonter.
    case "$_etat" in
        active|activating|reloading) ;;   # ExecStop a fait le travail
        *)
```

```
                [ -x "$TARGET" ] && timeout 120 "$TARGET" --stop >/dev/null 2>&1 || true
                # Les deux unites que le second ExecStop de osmo-banc.service
                # arrete : elles ne sont PAS dans le cgroup du banc et
                # survivraient a un arret par ce chemin-la.
                systemctl stop osmo-egprs-web.service osmo-hlr.service 2>/dev/null || true
                ;;
        esac
        # Le natif est a terre ; les operateurs en conteneur, eux, ne
        # dependent d aucun de ces chemins (voir _arret_multi).
        _arret_multi
        # Sans ca l unite reste « failed » et le prochain demarrage part d un
        # etat d echec.
        systemctl reset-failed "$_svc" 2>/dev/null || true
        _note "Banc arrete."
        exit 0
    fi

    if ! systemctl cat "$_svc" >/dev/null 2>&1; then
        _note "Unite $_svc absente : utilisez « Lancer le banc GSM »."
        exit 1
    fi
    # Meme garde que le lancement en fenetre : deux demarrages a la fois se
    # demontent l un l autre.
    if ! lancement_possible; then
        _note "Un lancement est deja en cours - rien de fait. journalctl -u ${_svc%.service} -f"
        exit 1
    fi
    # Les options du clic droit (--dsp...) passent a l unite comme dans
    # banc_unit_restart : meme mecanique, meme portee (jusqu au reboot).
    if [ $# -gt 0 ]; then
        systemctl set-environment OSMO_BANC_ARGS="$*" 2>/dev/null || true
    else
        systemctl unset-environment OSMO_BANC_ARGS 2>/dev/null || true
    fi
    systemctl reset-failed "$_svc" 2>/dev/null || true
    _note "Demarrage de $_svc - comptez plusieurs minutes."
    if systemctl restart "$_svc"; then
        _note "Banc demarre. Console : tmux attach -t calypso"
        exit 0
    fi
    _note "Echec de $_svc - journalctl -u ${_svc%.service} -n 80"
    exit 1
fi

# ── UN CLIC = UN SERVICE ; UN DEUXIEME CLIC = UNE FENETRE QUI DIT POURQUOI ──
# [2026-09-04] Un clic sur le telephone rouge ouvrait une fenetre de terminal et
# y deroulait tout le demarrage. C est utile quand on lance a la main, ca ne
# l est pas depuis une icone : on veut le banc, pas sa sortie, et la fenetre
# reste ensuite ouverte sur un « Entree pour fermer » que personne ne lit.
#
# Depuis une icone (ni stdin ni stdout sur un terminal), on part donc en mode
# service : systemd monte le banc, une notification annonce le resultat, aucune
# fenetre ne s ouvre. C est exactement l action « Demarrer le banc en service »
# du clic droit - un seul chemin, pas deux comportements a tenir a jour.
#
# LE DEUXIEME CLIC NE DOIT RIEN OUVRIR NON PLUS - mais il doit se VOIR. Sans
# rien, un double-clic sortirait en silence et l icone passerait pour morte. Il
# repond donc par une notification de bureau : le verrou est annonce, aucune
# fenetre n apparait.
#
# Lancement a la main dans un terminal : rien ne change, on garde ce terminal,
# et c est la que l avertissement encadre du verrou (plus bas) s affiche.
if { [ ! -t 0 ] || [ ! -t 1 ]; } \
   && [ "${OSMO_LAUNCH_TERM:-0}" != "1" ] && [ "${OSMO_TERM_TAKEN:-0}" != "1" ] \
   && [ "${OSMO_LAUNCH_NO_SERVICE:-0}" != "1" ] \
   && systemctl cat "${OSMO_BANC_SERVICE:-osmo-banc.service}" >/dev/null 2>&1; then
```

```
    if lancement_possible; then
        # [2026-10-12] AVEC UNE FENETRE, SANS SHELL : depuis le bureau, le
        # lancement s affiche dans la fenetre de progression du banc
        # (tools/osmo-gtk-run.py --progress : barre, etape, console repliee).
        # OSMO_LAUNCH_GTK=0 revient aux seules notifications.
        _gtk="$(dirname "$(readlink -f "$0")")/tools/osmo-gtk-run.py"
        if [ "${OSMO_LAUNCH_GTK:-1}" != 0 ] && [ -n "${DISPLAY:-}" ] && [ -x "$_gtk" ]; then
            # le verrou est rendu ICI : c est le --service lance dans la
            # fenetre (en root) qui le reprend ; garde par la fenetre, il
            # ferait refuser ce lancement meme
            verrou_liberer
            exec python3 "$_gtk" --progress --root --title "Banc GSM 2G" \
                 --icon /usr/share/osmo-operator/icons/osmo-launch.svg -- "$0" --service-suivi "$@"
        fi
        # Le verrou pris ici survit a l exec : c est le meme processus qui va
        # tenir le demarrage, sous son autre nom.
        exec "$0" --service "$@"
    fi
    # Verrou pris : on le dit, et on s arrete la. Pas de fenetre.
    if command -v notify-send >/dev/null 2>&1; then
        notify-send -i /usr/share/osmo-operator/icons/osmo-launch.svg "Banc GSM" \
            "Un lancement est deja en cours. Rien de fait - suivez-le avec journalctl -u osmo-banc -f." \
            2>/dev/null || true
    fi
    echo "Un lancement est deja en cours - rien de fait." >&2
    exit 1
fi

# ── 1. Un terminal, si on n en a pas — OU s il n est pas a nous ──────────────
# On teste la presence d un TTY, pas $DISPLAY : c est la difference reelle entre
# « lance a la main » et « lance par une icone ». OSMO_LAUNCH_TERM marque le
# tour de relance pour ne pas boucler.
#
# [2026-08-31] DEUX DEFAUTS, LE MEME MALENTENDU : la garde demandait « y a-t-il
# UN terminal ? », jamais « est-ce LE MIEN ? ».
#   1. stdin SEUL. `./launch.sh < fichier` depuis un vrai terminal se croyait
#      lance par une icone et ouvrait une fenetre en trop, dont la sortie
#      n allait pas a l appelant. On teste donc stdin ET stdout.
#   2. UN TTY HERITE N EST PAS LE NOTRE. start.sh TIENT le terminal
#      jusqu a la fin (cf. l en-tete de ce fichier). Appele par un autre
#      lanceur — start-multi.sh, qui monte le natif AVANT ses conteneurs — il
#      heritait du terminal de celui-ci, ecrivait par-dessus sa sortie et ne
#      lui rendait la main qu a la fin du natif. OSMO_TERM_TAKEN=1 dit « ce
#      terminal est deja occupe » : on ouvre alors LA NOTRE, tty ou pas.
_lt_besoin=0
{ [ ! -t 0 ] || [ ! -t 1 ]; } && _lt_besoin=1
[ "${OSMO_TERM_TAKEN:-0}" = "1" ] && _lt_besoin=1
if [ "$_lt_besoin" = "1" ] && [ "${OSMO_LAUNCH_TERM:-0}" != "1" ]; then
    export OSMO_LAUNCH_TERM=1
    unset OSMO_TERM_TAKEN            # la fenetre qu on ouvre ici EST la notre
    # [2026-10-01] La fenetre est celle du banc (GTK + terminal VTE,
    # tools/osmo-gtk-run.py), plus un gnome-terminal.
    exec python3 "$(dirname "$(readlink -f "$0")")/tools/osmo-gtk-run.py" --tty \
         --title "Banc GSM" -- "$0" "$@"
    # AUCUN emulateur installe. On ne reprend pas le terminal de l appelant -
    # ce serait refaire exactement la panne que cette garde empeche. On se
    # detache, et la sortie va dans un journal plutot que dans le vide.
    _lt_log="${OSMO_LAUNCH_LOG:-/var/log/osmo-launch.log}"
    echo -e "${YELLOW}Aucun emulateur de terminal - lancement detache, journal : ${_lt_log}${NC}" >&2
    exec setsid "$0" "$@" </dev/null >>"$_lt_log" 2>&1
fi

# Le drapeau a joue son role. On le RETIRE de l environnement : exporte, il
# descendait dans start.sh et tout ce qui suit, ou il ferait croire a
# n importe quel descendant rappelant ce script qu il est deja dans sa fenetre
# - et plus aucune ne s ouvrirait. La valeur reste ici, dans une variable de
```

```
# shell, la ou on en a besoin (retenir la fenetre a la fin).
_own_window=0
[ "${OSMO_LAUNCH_TERM:-0}" = "1" ] && _own_window=1
unset OSMO_LAUNCH_TERM

# ── 2. Les privileges ──────────────────────────────────────────────
if [ "$(id -u)" -ne 0 ]; then
    if command -v pkexec >/dev/null 2>&1 && [ -n "${DISPLAY:-}" ]; then
        # pkexec lave l environnement : DISPLAY et XAUTHORITY sont repasses a la
        # main, sinon rien de graphique ne peut s ouvrir ensuite.
        exec pkexec env DISPLAY="${DISPLAY:-}" XAUTHORITY="${XAUTHORITY:-}" \
             OSMO_LAUNCH_TERM="$_own_window" "$0" "$@"
    fi
    command -v sudo >/dev/null 2>&1 && exec sudo -E "$0" "$@"
    echo -e "${RED}Root requis et ni pkexec ni sudo disponibles.${NC}" >&2
    exit 1
fi

# ── L UTILISATEUR DE LA SESSION GRAPHIQUE ──────────────────────────
# Meme critere que helpers/prepare_host.sh et start-docker.sh : le NOM du compte ne
# dit rien, la presence de son socket pulse si. Sur l ISO --desktop la session
# est ouverte sous root (gdm3 AutomaticLogin=root) - « root » est donc ici une
# reponse valide, pas une erreur.
session_user() {
    local u uid sock
    for u in "${HOST_PULSE_USER:-}" "${SUDO_USER:-}" "$(logname 2>/dev/null || true)"; do
        [ -n "$u" ] || continue
        uid="$(id -u "$u" 2>/dev/null)" || continue
        [ -S "/run/user/${uid}/pulse/native" ] && { echo "$u"; return 0; }
    done
    for sock in /run/user/*/pulse/native; do
        [ -S "$sock" ] || continue
        uid="${sock#/run/user/}"; uid="${uid%%/*}"
        u="$(getent passwd "$uid" 2>/dev/null | cut -d: -f1)"
        [ -n "$u" ] && { echo "$u"; return 0; }
    done
    echo "${SUDO_USER:-$(id -un)}"
}
GUI_USER="$(session_user)"
GUI_UID="$(id -u "$GUI_USER" 2>/dev/null || echo 0)"
GUI_HOME="$(getent passwd "$GUI_USER" 2>/dev/null | cut -d: -f6)"
export DISPLAY="${DISPLAY:-:0}"
export XAUTHORITY="${XAUTHORITY:-${GUI_HOME:-/root}/.Xauthority}"

# Lance une application graphique sous le compte de la session. `setsid` la
# detache : elle survit a la fermeture de ce terminal, et son code de retour ne
# peut pas faire tomber launch.sh (set -e n est pas arme ici, mais on garde le
# || true pour l intention).
gui_run() {
    setsid sudo -u "$GUI_USER" \
        env DISPLAY="$DISPLAY" XAUTHORITY="$XAUTHORITY" \
            DBUS_SESSION_BUS_ADDRESS="unix:path=/run/user/${GUI_UID}/bus" \
        "$@" </dev/null >/dev/null 2>&1 &
    disown 2>/dev/null || true
}

# ── LE PAVAGE 2x2 ──────────────────────────────────────────────────
# Chaque action prend UN QUART de l ecran :
#
#        +---------------------+---------------------+
#        |  terminal du banc   |  firefox / dashboard|
#        |  (start.sh)         |  127.0.0.1:8080     |
#        +---------------------+---------------------+
#        |  wireshark          |  linphone           |
#        |  udp/4729 GSMTAP    |  postes 100 / 200   |
#        +---------------------+---------------------+
```

```
#
# Pourquoi wmctrl et pas les raccourcis GNOME : le pavage au clavier est une
# action d utilisateur, il n existe pas en ligne de commande. wmctrl parle a
# l EWMH, ce que Mutter expose EN X11 - et l ISO force X11 (build-iso.sh ecrit
# WaylandEnable=false dans gdm3/custom.conf, la session Wayland tombant sur
# llvmpipe en machine virtuelle). Sous Wayland, rien de tout ceci ne repondrait
# et le pavage serait simplement ignore, sans casser le lancement.
#
# ⚠ Retirer les etats maximise AVANT de poser la geometrie : une fenetre
# maximisee accepte l ordre et n en tient aucun compte - elle reste plein ecran
# et l on croit wmctrl casse.
tile_windows() {
    command -v wmctrl  >/dev/null 2>&1 || { echo -e "  ${YELLOW}○${NC} wmctrl absent - fenetres non pavees";
     ↪  return 0; }
    command -v xdpyinfo >/dev/null 2>&1 || { echo -e "  ${YELLOW}○${NC} xdpyinfo absent - fenetres non pavees";
     ↪  return 0; }

    local sw sh qw qh
    sw="$(xdpyinfo 2>/dev/null | awk '/dimensions:/{split($2,a,"x"); print a[1]; exit}')"
    sh="$(xdpyinfo 2>/dev/null | awk '/dimensions:/{split($2,a,"x"); print a[2]; exit}')"
    [ -n "$sw" ] && [ -n "$sh" ] || return 0
    qw=$(( sw / 2 )); qh=$(( sh / 2 ))

    # $1 = motif WM_CLASS (insensible a la casse), $2 = x, $3 = y
    place() {
        local pat="$1" x="$2" y="$3" i id
        for i in $(seq 1 90); do
            id="$(wmctrl -l -x 2>/dev/null | grep -iE "$pat" | awk '{print $1; exit}')"
            [ -n "$id" ] && break
            sleep 1
        done
        [ -n "$id" ] || return 0
        wmctrl -i -r "$id" -b remove,maximized_vert,maximized_horz 2>/dev/null || true
        wmctrl -i -r "$id" -e "0,${x},${y},${qw},${qh}" 2>/dev/null || true
    }

    place 'gnome-terminal|xfce4-terminal|konsole|xterm'  0    0
    place 'firefox|navigator'                           "$qw" 0
    place 'wireshark'                                    0   "$qh"
    place 'linphone'                                    "$qw" "$qh"
}

# ── LE BANC EST UNE UNITE SYSTEMD QUAND ELLE EXISTE ──────────────────────
# [2026-09-04] services/osmo-banc.service demarre le banc sans terminal et le
# tient dans la session tmux « calypso ». (Elle est POSEE mais plus activee au
# boot depuis le 2026-09-05 : c est ce script, ou l icone, qui la demarre.)
# Quand l unite est
# installee, ce script ne lance plus start.sh lui-meme : il (re)demarre
# l unite, ouvre les applications du bureau comme avant, puis S ATTACHE a tmux
# dans son terminal - le meme ecran qu avant, mais un seul proprietaire du
# banc. Lancer start.sh a la main par-dessus tuerait les processus du
# service sans que systemd le sache (unite « active (exited) », pile morte).
#
# Les arguments (--dsp, --op 2, --regen...) passent a l unite par
# OSMO_BANC_ARGS (systemctl set-environment : vaut jusqu au reboot, puis
# /etc/default/osmo-banc reprend). Sans argument, on efface un reste eventuel.
#
# OSMO_BANC_DIRECT=1 ./launch.sh contourne l unite (debogage a la main).
BANC_UNIT="${OSMO_BANC_SERVICE:-osmo-banc.service}"
banc_unit_present() {
    [ "${OSMO_BANC_DIRECT:-0}" = "1" ] && return 1
    command -v systemctl >/dev/null 2>&1 || return 1
    [ -d /run/systemd/system ] || return 1
    systemctl cat "$BANC_UNIT" >/dev/null 2>&1
}
# L arret vit en tete de fichier (bloc « ARRETER »), avant toute fenetre :
```

```
# `launch.sh --stop` n arrive donc jamais jusqu ici.
# Suit le journal de l unite pendant le demarrage : c est la sortie de
# start.sh / run.sh qu on lisait avant dans ce meme terminal.
banc_unit_restart() {
    if [ $# -gt 0 ]; then
        systemctl set-environment OSMO_BANC_ARGS="$*" 2>/dev/null || true
        echo -e "  ${CYAN}→${NC} options : ${CYAN}$*${NC} (OSMO_BANC_ARGS, jusqu au reboot)"
    else
        systemctl unset-environment OSMO_BANC_ARGS 2>/dev/null || true
    fi
    echo -e "  ${CYAN}→${NC} redemarrage de ${BANC_UNIT} (un clic = un banc neuf)"
    journalctl -u "${BANC_UNIT%.service}" -f -n 0 -o cat 2>/dev/null &
    local _jpid=$!
    systemctl restart "$BANC_UNIT"; local rc=$?
    sleep 1; kill "$_jpid" 2>/dev/null; wait "$_jpid" 2>/dev/null
    if [ "$rc" -ne 0 ]; then
        echo -e "  ${RED}✗${NC} ${BANC_UNIT} en echec (rc=$rc) - journalctl -u ${BANC_UNIT%.service} -n 80"
        systemctl status --no-pager -n 20 "$BANC_UNIT" 2>/dev/null || true
    else
        echo -e "  ${GREEN}✓${NC} banc demarre par ${BANC_UNIT}"
    fi
    return "$rc"
}

# Le verrou (voir son bloc en tete de fichier). Ici, et pas plus haut : `--stop`
# doit passer, et les privileges doivent etre regles - un flock pris avant le
# re-exec sudo serait tenu par un processus qui disparait aussitot.
if [ "${1:-}" != "--stop" ] && ! lancement_possible; then
    echo
    echo -e "${YELLOW}+-------------------------------------------------------------+${NC}"
    echo -e "${YELLOW}|  UN LANCEMENT EST DEJA EN COURS                             |${NC}"
    echo -e "${YELLOW}|  Cette fenetre ne lancera RIEN.                             |${NC}"
    echo -e "${YELLOW}+-------------------------------------------------------------+${NC}"
    echo
    echo -e "  Le banc met plusieurs minutes a monter (QEMU, Calypso, mobiles)."
    echo -e "  Un second lancement ARRETERAIT le premier : un clic = un banc"
    echo -e "  neuf, et le neuf commence par demonter celui d avant."
    echo
    echo -e "  Suivre le demarrage   : ${CYAN}journalctl -u osmo-banc -f${NC}"
    echo -e "  Console du banc       : ${CYAN}tmux attach -t calypso${NC}"
    echo -e "  Vraiment recommencer  : clic droit sur l icone, ${CYAN}Arreter le banc${NC},"
    echo -e "                          puis relancer."
    echo
    command -v notify-send >/dev/null 2>&1 && \
        notify-send -i /usr/share/osmo-operator/icons/osmo-launch.svg "Banc GSM" \
            "Un lancement est deja en cours - cette fenetre ne fera rien." 2>/dev/null || true
    if [ "$_own_window" = "1" ]; then
        echo "Entree pour fermer."
        read -r _ || true
    fi
    exit 1
fi

# ── TOUT ARRETER AVANT DE RELANCER ──────────────────────────────
# [2026-08-31] Un clic = un banc NEUF. Sans ca, un lancement par-dessus un banc
# deja debout ne relance presque rien : les demons Osmocom ne relisent PAS leur
# configuration a chaud. On l a paye plusieurs fois - un osmo-stp realigne sur
# le bon point code gardait l ancien en memoire, et `kill -HUP` n y change rien
# (verifie : config a jour sur le disque, ASP toujours ASP_DOWN sur l ancienne
# adresse). Le seul geste qui compte est l arret complet.
#
# On arrete le NATIF (il delegue a run.sh --stop, qui demonte proprement radio
# et coeur). Rien ici ne doit faire echouer le lancement : ce qui est deja
# arrete l est tres bien, d ou les || true.
tout_arreter() {
    # Avec l unite, l arret est fait par `systemctl restart` plus bas.
```

```
    banc_unit_present && return 0
    echo -e "  ${CYAN}→${NC} arret du banc en place (un clic = un banc neuf)"
    if [ -x "$DIR/start.sh" ]; then
        timeout 120 "$DIR/start.sh" --stop >/dev/null 2>&1 || true
        echo -e "    ${GREEN}✓${NC} pile native arretee"
    fi
    # Les conteneurs ne sont PAS touches ici : start-docker.sh fait deja
    # `docker rm -f $(docker ps -aq --filter name=osmo-)` en tete de course.
    # Le refaire serait redondant, et surtout destructeur depuis launch.sh -
    # un clic sur le telephone rouge, qui ne monte que le natif, balayerait
    # les conteneurs multi-operateur d a cote.
    # Les ponts audio survivent aux conteneurs (setsid) : sans ca on empile un
    # relais de plus a chaque relance, et le son se dedouble.
    pkill -f 'paplay --server=tcp:' 2>/dev/null || true
}

tout_arreter

# ── LES OUTILS D EXPLOITATION SONT-ILS DEMANDES ? ──────────────────────
# [2026-08-31] OSMO_LAUNCH_APPS=0 : ni wireshark, ni linphone. C est ce que
# demande start-multi.sh quand il declenche ce lanceur pour remonter le natif -
# il veut la PILE, pas une seconde instance des outils.
#
# ⚠ FIREFOX N EST PAS CONCERNE, et c est deliberé : le navigateur porte le
# dashboard du natif et le tutoriel, et start-multi.sh y AJOUTE les onglets des
# dashboards conteneurs. Le couper ici priverait un clic sur l antenne bleue de
# la base sur laquelle ces onglets viennent se greffer - on n aurait plus que
# les deux consoles conteneurs, sans celle du natif ni le tutoriel.
# Firefox, lui, sait ne pas se dedoubler : une seconde invocation ajoute des
# onglets a la fenetre existante.
#
# [2026-09-04] PAR DEFAUT A 0. Un clic sur le telephone rouge ouvrait QUATRE
# fenetres d un coup : son terminal, Wireshark, Linphone et le navigateur -
# empilees, a repositionner a la main, alors qu on voulait seulement monter le
# banc. Wireshark et Linphone sont des outils qu on ouvre QUAND on en a besoin,
# depuis leur propre icone ; ils ne font pas partie du demarrage. Restent le
# terminal (la sortie du demarrage) et le navigateur (le tableau de bord).
# OSMO_LAUNCH_APPS=1 ./launch.sh les rouvre pour qui les veut tous.
OSMO_LAUNCH_APPS="${OSMO_LAUNCH_APPS:-0}"
if [ "$OSMO_LAUNCH_APPS" != "1" ]; then
    echo -e "  ${CYAN}i${NC} wireshark et linphone non lances (OSMO_LAUNCH_APPS=1 pour les avoir)"
fi

# ── 1. WIRESHARK sur udp/4729 ──────────────────────────────────────────
# En ROOT, et pas sous $GUI_USER : la capture sur `any` demande CAP_NET_RAW.
# -k demarre la capture tout de suite (sinon on ouvre sur un ecran de choix
# d interface, et rien n est capture tant qu on n a pas clique).
# [2026-08-31] LES DEUX RAISONS DE NE PAS LANCER SONT DISTINGUEES.
# Le test cumulait "demande ?" et "installe ?" dans un seul if, et le else
# annoncait toujours "absent". En mode banc seul (OSMO_LAUNCH_APPS=0, pose par
# start-multi.sh), on lisait donc "wireshark absent" pour un wireshark
# parfaitement installe - un message qui envoie verifier un paquet present.
if [ "$OSMO_LAUNCH_APPS" != "1" ]; then
    echo -e "  ${CYAN}○${NC} wireshark : non demande (mode banc seul)"
elif command -v wireshark >/dev/null 2>&1; then
    setsid wireshark -k -i any -f "udp port ${GSMTAP_PORT}" \
        </dev/null >/dev/null 2>&1 &
    disown 2>/dev/null || true
    echo -e "  ${GREEN}✓${NC} wireshark : capture GSMTAP sur ${CYAN}udp/${GSMTAP_PORT}${NC}"
else
    echo -e "  ${YELLOW}○${NC} wireshark absent - capture GSMTAP ignoree"
fi

# ── 2. LINPHONE ────────────────────────────────────────────────────────
# Le compte SIP est pre-provisionne (configs/linphonerc, pose par build-iso.sh).
# ⚠ Linphone ne relit PAS sa config a chaud : s il tourne deja, il garde son
```

```
# etat en memoire. On ne le relance donc pas de force ici - on le laisse tel
# quel plutot que de couper un appel en cours.
if [ "$OSMO_LAUNCH_APPS" != "1" ]; then
    echo -e "  ${CYAN}○${NC} linphone : non demande (mode banc seul)"
elif command -v linphone >/dev/null 2>&1; then
    if pgrep -x linphone >/dev/null 2>&1; then
        echo -e "  ${GREEN}✓${NC} linphone : deja lance (laisse en place)"
    else
        gui_run linphone
        echo -e "  ${GREEN}✓${NC} linphone : lance (user=${CYAN}${GUI_USER}${NC})"
    fi
else
    echo -e "  ${YELLOW}○${NC} linphone absent"
fi

# ── 3. FIREFOX sur le dashboard, QUAND IL REPOND ──────────────────────
# Ouvrir l URL tout de suite donnerait une page d erreur : le dashboard n est
# pas encore la, et une fois l onglet en echec l utilisateur doit recharger a la
# main. On attend donc que le port reponde, en tache de fond, jusqu a 180 s.
if command -v firefox >/dev/null 2>&1; then
    (
        for _i in $(seq 1 180); do
            if (exec 3<>"/dev/tcp/127.0.0.1/${DASH_PORT}") 2>/dev/null; then
                exec 3>&- 2>/dev/null
                # Deux URL sur la meme ligne de commande = deux ONGLETS, dans
                # la meme fenetre. Un second `firefox <url>` marcherait aussi
                # (le premier processus recupere l argument), mais il faudrait
                # le cadencer : lance trop tot, il fait la course avec le
                # demarrage du navigateur et ouvre parfois une 2e fenetre.
                gui_run firefox "$DASH_URL" "$TUTO_URL"
                exit 0
            fi
            sleep 1
        done
    ) &
    disown 2>/dev/null || true
    echo -e "  ${GREEN}✓${NC} firefox : ouvrira ${CYAN}${DASH_URL}${NC} des que le dashboard repond"
else
    echo -e "  ${YELLOW}○${NC} firefox absent - dashboard a ouvrir a la main sur ${DASH_URL}"
fi

# Le pavage part en tache de fond : chaque fenetre est placee des qu elle
# apparait, sans retarder le demarrage du banc.
tile_windows &
disown 2>/dev/null || true

# ── 4. LE BANC, AU PREMIER PLAN ──────────────────────────────────────
echo
cd "$DIR" || exit 1

# ── AUCUNE SAISIE ────────────────────────────────────────────────────
# [2026-08-31] start.sh ouvre menu_interactif() (startup_scripts/06-banc-dsp-menu.sh) : Profil, Noeud SS7,
# Operateur, Inter-STP - quatre questions whiptail avant que quoi que ce soit
# ne demarre. Il les saute quand PERSONNE ne peut repondre, et il teste ca sur
# la presence d un terminal ([ -t 0 ]) - or launch.sh vient justement d en
# ouvrir un pour afficher la sortie. La garde ne se declenchait donc jamais
# ici, et un double-clic sur le telephone rouge s arretait sur un formulaire
# au lieu de lancer le banc.
#
# NO_MENU=1 est le drapeau prevu pour ca (startup_scripts/06-banc-dsp-menu.sh et 10-identite-noeud.sh) :
# les menus rendent la main sans rien demander et les valeurs par defaut
# s appliquent - noeud 1, operateur 1, hub local. C est exactement ce que
# start-docker.sh pose deja devant ses conteneurs.
#
# OSMO_LAUNCH_MENU=1 ./launch.sh redonne les questions quand on les veut.
if banc_unit_present; then
```

```
    banc_unit_restart "$@"
    rc=$?
    # Le banc tourne dans tmux : on s y attache si on a un terminal, comme
    # run.sh le faisait en fin de course. Ctrl-b d pour detacher ; le banc
    # continue sans ce terminal.
    if [ "$rc" -eq 0 ] && [ -t 1 ] && tmux has-session -t "${TMUX_SESSION:-calypso}" 2>/dev/null; then
        echo -e "  ${CYAN}connexion a la session tmux « ${TMUX_SESSION:-calypso} »  (Ctrl-b d pour detacher)${NC}"
        sleep 1
        verrou_liberer   # sinon le client tmux le garderait tant qu on reste attache
        exec tmux attach -t "${TMUX_SESSION:-calypso}"
    fi
elif [ "${OSMO_LAUNCH_MENU:-0}" = "1" ]; then
    "$TARGET" "$@"
    rc=$?
else
    NO_MENU=1 "$TARGET" "$@"
    rc=$?
fi

verrou_liberer

echo
if [ "$rc" -eq 0 ]; then
    echo -e "${GREEN}=== start.sh termine (code 0) ===${NC}"
else
    echo -e "${RED}=== start.sh a ECHOUE (code $rc) ===${NC}"
fi
# Uniquement dans une fenetre ouverte par nos soins : au terminal, rendre la
# main tout de suite est le comportement attendu.
if [ "$_own_window" = "1" ]; then
    echo "Fenetre maintenue ouverte - Entree pour fermer."
    read -r _ || true
fi
exit "$rc"
```

3.19 /opt/GSM/osmo-operator/reset

0 octets, 0 lignes → 0 lignes

3.20 /opt/GSM/osmo-operator/set_stp_ip.sh

28463 octets, 545 lignes → 545 lignes

```
#!/bin/bash
# ============================================================================
# set_stp_ip.sh - donne a CETTE VM son adresse SS7 sur le reseau en pont
#
# Une seule ISO, deux roles. Le script demande d'abord LEQUEL, puis les
# adresses - parce que ce ne sont pas les memes des deux cotes :
#
#   inter-STP (hub)      son IP de pont, + l'IP de chaque osmo-operator-N
#   osmo-operator-N      son IP de pont, + l'IP de l'inter-STP
#
#   osmo-operator-1  192.168.1.11 ─┐
#   osmo-operator-2  192.168.1.12 ─┼─ M3UA/SCTP 2908 ─► inter-STP 192.168.1.10
#   osmo-operator-3  192.168.1.13 ─┘                     PC 0.0.0
#
# POURQUOI LE PONT ET PAS L'HOTE-SEUL
# Les configs de l'image visent 192.168.56.1 (host-only VirtualBox) ou
# 172.20.0.10 (reseau docker). Ni l'un ni l'autre n'existe quand les VM sont en
```

```
# "acces par pont" : elles sont sur le LAN, en 192.168.1.X. Un ASP qui vise
# une adresse absente du segment ne dit rien d'autre que "connection
# refused" cote noeud - le hub, lui, ne voit jamais personne arriver.
#
# CE QUE LE SCRIPT ECRIT
#   hub       /etc/osmocom/osmo-stp-interop.cfg   local-ip = son IP de pont
#             /etc/osmo-wan.conf                  la table des operateurs
#             /etc/osmo-role                      OSMO_ROLE=interstp
#   operateur /etc/osmocom/osmo-stp.cfg           local-ip = la sienne
#                                                 remote-ip = celle du hub
#             /etc/osmocom/osmo-{msc,bsc}.cfg     point codes du plan WAN
#             /etc/osmo-wan.conf, /etc/osmo-role
#
# Idempotent : relancer avec d'autres adresses les remplace, il n'y a pas
# d'etat a nettoyer entre deux passages.
#
# Usage :
#   sudo ./set_stp_ip.sh                        menu : role, puis adresses
#   sudo ./set_stp_ip.sh --inter --ip 192.168.1.10 --ips "192.168.1.11 192.168.1.12"
#   sudo ./set_stp_ip.sh --operator --node 1 --ip 192.168.1.11 --hub-ip 192.168.1.10
#   sudo ./set_stp_ip.sh --show                 etat actuel, sans rien modifier
#
#   --inter | --operator   role (sinon : demande)
#   --ip ADRESSE           IP de pont de CETTE machine
#   --ips "a b c"          hub : IP des operateurs, dans l'ordre
#   --hub-ip ADRESSE       operateur : IP de l'inter-STP
#   --node N               operateur : son numero de noeud (1-9)
#   --op K                 COMBIEN d'operateurs (PLMN) ce noeud porte (defaut 1)
#   --op-index K           LEQUEL de ces operateurs est celui de ce noeud, dans
#                          le point code 1.<noeud><op>.<role> (1 a 9, pas plus
#                          que --op ; defaut 1)
#   --iface NOM            interface ou poser l'adresse (defaut : detectee)
#   --port N               port M3UA (defaut 2908)
#   --no-ip                ne touche pas a la configuration reseau
#   --defaut               la configuration du banc, sans une question : role
#                          deduit, IP de pont deduite, hub 192.168.1.49, noeud
#                          lu dans /etc/osmo-role ou la table WAN.
#   --restart              relance la pile apres reecriture
#   --dry-run              montre ce qui serait ecrit
# ═══════════════════════════════════════════════════════════════════════════
set -uo pipefail

RED='\033[0;31m'; GREEN='\033[0;32m'; YELLOW='\033[1;33m'
CYAN='\033[0;36m'; BOLD='\033[1m'; NC='\033[0m'

HERE="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
CONF_DIR="${CONF_DIR:-/etc/osmocom}"
ROLE_FILE="${ROLE_FILE:-/etc/osmo-role}"
INTEROP_CFG="${CONF_DIR}/osmo-stp-interop.cfg"

# wan_detect_local_ip, _wan_ask, la table WAN et sa validation vivent ici. Les
# redefinir localement, c'est se garantir qu'un jour ce script et les lanceurs
# ne designeront plus la meme carte ni la meme table.
# shellcheck source=network/wan-nodes.sh
. "$HERE/network/wan-nodes.sh"

ROLE=""
SELF_IP=""
HUB_IP=""
IPS=""
NODE=""
OPS=1
# --op COMPTE les operateurs : il dimensionne le pare-feu, les blocs "as" du hub
# et la table WAN. Il ne dit pas LEQUEL est celui de ce noeud - et c'est pourtant
# lui qui partait en --op de set-node-id.sh, ou l'option est un INDICE. Avec
# --op 2 sur le noeud 3, l'ecran annoncait STP 1.31.2 pendant que les fichiers
```

```
# recevaient 1.32.2 et le routing context 3250 : l'ASP se presentait au hub sous
# une identite que sa table ne connait pas.
OP_INDEX=1
IFACE=""
PORT=2908
SET_IP=1
RESTART=0
# ── La configuration du banc, sans une question ──────────────────────────
# --defaut repond a tout ce que ce script demande d'habitude, a partir de ce
# que la machine sait deja d'elle-meme :
#   role     interstp si son adresse EST celle du hub, operateur sinon ;
#   son IP   celle du pont, deduite ;
#   le hub   172.20.0.10 (osmo-inter-stp, backbone docker) ;
#   le noeud /etc/osmo-role, puis la table WAN, puis 1.
# Chaque valeur reste surchargeable : --defaut --node 3 gagne sur la deduction.
USE_DEFAULTS=0
DEFAULT_HUB_IP="172.20.0.10"
DRY=0
SHOW=0

while [ $# -gt 0 ]; do
    case "$1" in
        --inter|--interstp|--hub)  ROLE="interstp" ;;
        --operator|--op-node)      ROLE="operator" ;;
        --role)     ROLE="${2:-}"; shift ;;
        --role=*)   ROLE="${1#*=}" ;;
        --ip)       SELF_IP="${2:-}"; shift ;;
        --ip=*)     SELF_IP="${1#*=}" ;;
        --hub-ip)   HUB_IP="${2:-}"; shift ;;
        --hub-ip=*) HUB_IP="${1#*=}" ;;
        --ips)      IPS="${2:-}"; shift ;;
        --ips=*)    IPS="${1#*=}" ;;
        --node)     NODE="${2:-}"; shift ;;
        --node=*)   NODE="${1#*=}" ;;
        --op)       OPS="${2:-1}"; shift ;;
        --op=*)     OPS="${1#*=}" ;;
        --ops)      OPS="${2:-1}"; shift ;;
        --ops=*)    OPS="${1#*=}" ;;
        --op-index)   OP_INDEX="${2:-1}"; shift ;;
        --op-index=*) OP_INDEX="${1#*=}" ;;
        --iface)    IFACE="${2:-}"; shift ;;
        --iface=*)  IFACE="${1#*=}" ;;
        --port)     PORT="${2:-2908}"; shift ;;
        --port=*)   PORT="${1#*=}" ;;
        --conf-dir)   CONF_DIR="${2:-}"; INTEROP_CFG="${CONF_DIR}/osmo-stp-interop.cfg"; shift ;;
        --conf-dir=*) CONF_DIR="${1#*=}"; INTEROP_CFG="${CONF_DIR}/osmo-stp-interop.cfg" ;;
        --defaut|--default) USE_DEFAULTS=1 ;;
        --no-ip)    SET_IP=0 ;;
        --restart)  RESTART=1 ;;
        --dry-run)  DRY=1 ;;
        --show)     SHOW=1 ;;
        -h|--help)  sed -n '2,53p' "$0" | sed 's/^# \{0,1\}//'; exit 0 ;;
        *) echo -e "${RED}option inconnue : $1${NC}" >&2; exit 2 ;;
    esac
    shift
done

banner() {
    echo -e "${CYAN}${BOLD}"
    echo "╔══════════════════════════════════════════════════════════════╗"
    echo "║   set_stp_ip.sh - adressage SS7 sur le reseau en pont        ║"
    echo "╚══════════════════════════════════════════════════════════════╝"
    echo -e "${NC}"
}

is_ipv4() {
```

```
    local ip="$1" o
    [[ "$ip" =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]] || return 1
    for o in ${ip//./ }; do [ "$o" -le 255 ] || return 1; done
    return 0
}

need_ipv4() {  # $1=valeur  $2=ce que c'est
    is_ipv4 "$1" || { echo -e "${RED}$2 : '$1' n'est pas une IPv4${NC}" >&2; exit 2; }
}

# ── L'adresse de pont de cette VM ──────────────────────────────────────
# wan_detect_local_ip privilegie le host-only VirtualBox (192.168.56-63), qui
# est le bon choix pour le banc mais pas ici : en acces par pont, la VM est sur
# le LAN de l'hote. On cherche donc d'abord une adresse de LAN privee qui ne
# soit ni le NAT VirtualBox (10.0.2.x), ni un alias du plan docker (172.20.x),
# ni le host-only - et on ne retombe sur wan_detect_local_ip qu'a defaut.
bridge_ip() {
    local addrs a
    addrs="$(ip -4 -o addr show scope global 2>/dev/null | awk '{print $4}' | cut -d/ -f1)"
    for a in $addrs; do
        case "$a" in
            192.168.5[6-9].*|192.168.6[0-3].*) ;;   # host-only : pas le pont
            192.168.*) printf '%s' "$a"; return 0 ;;
        esac
    done
    for a in $addrs; do
        case "$a" in
            10.0.2.*|172.20.*|127.*|169.254.*) ;;
            10.*|172.1[6-9].*|172.2[0-9].*|172.3[01].*) printf '%s' "$a"; return 0 ;;
        esac
    done
    wan_detect_local_ip 2>/dev/null || return 1
}

# L'interface qui porte deja ce /24 : c'est celle du segment vise. A defaut, la
# premiere interface globale - poser l'adresse sur la mauvaise carte donne un
# ASP qui part par une route qui ne mene pas au pair.
iface_for() {  # $1=ip
    local ip="$1" f
    f="$(ip -4 -o addr show scope global 2>/dev/null \
         | awk -v pfx="${ip%.*}." '$4 ~ "^"pfx {print $2; exit}')"
    [ -z "$f" ] && f="$(ip -4 -o addr show scope global 2>/dev/null | awk 'NR==1{print $2}')"
    printf '%s' "$f"
}

ensure_ip() {  # $1=ip - la pose si elle manque
    local ip="$1" dev
    [ "$SET_IP" = "1" ] || { echo -e "  ${YELLOW}-${NC} --no-ip : reseau inchange"; return 0; }
    if ip -4 addr show 2>/dev/null | grep -q "inet ${ip}/"; then
        echo -e "  ${GREEN}✓${NC} ${CYAN}${ip}${NC} deja presente sur cette machine"
        return 0
    fi
    dev="${IFACE:-$(iface_for "$ip")}"
    if [ "$DRY" = "1" ]; then
        echo -e "  [dry-run] ip addr add ${ip}/24 dev ${dev:-?}"; return 0
    fi
    if [ -n "$dev" ] && ip addr add "${ip}/24" dev "$dev" 2>/dev/null; then
        echo -e "  ${GREEN}✓${NC} ${CYAN}${ip}/24${NC} posee sur ${dev}"
    else
        echo -e "  ${YELLOW}⚠${NC} impossible de poser ${ip} (interface ${dev:-?})"
        echo -e "     Tant qu'elle manque, aucun ASP ne peut s'attacher."
    fi
}

write_role() {  # $1=role  $2=hub-ip  $3=node (vide pour le hub)
    if [ "$DRY" = "1" ]; then
```

```
        echo -e "  [dry-run] ${ROLE_FILE} : OSMO_ROLE=$1 OSMO_HUB_IP=$2 ${3:+OSMO_WAN_NODE=$3}"
        return 0
    fi
    {
        printf '# %s - genere par set_stp_ip.sh\n' "$ROLE_FILE"
        printf 'OSMO_ROLE=%s\n' "$1"
        [ -n "${3:-}" ] && printf 'OSMO_WAN_NODE=%s\n' "$3"
        printf 'OSMO_HUB_IP=%s\n' "$2"
    } > "$ROLE_FILE"
    echo -e "  ${GREEN}✓${NC} ${CYAN}${ROLE_FILE}${NC}"
}

role_of_file() { awk -F= '/^OSMO_ROLE=/{gsub(/[ \r\t]/,"",$2); print $2}' "$ROLE_FILE" 2>/dev/null | tail -1; }
hub_of_file()  { awk -F= '/^OSMO_HUB_IP=/{gsub(/[ \r\t]/,"",$2); print $2}' "$ROLE_FILE" 2>/dev/null | tail -1; }
node_of_file() { awk -F= '/^OSMO_WAN_NODE=/{gsub(/[ \r\t]/,"",$2); print $2}' "$ROLE_FILE" 2>/dev/null | tail -1;
 ↪  }

# ═══════════════════════════════════════════════════════════════════
# --show : ce que cette machine croit etre, aujourd'hui
# ═══════════════════════════════════════════════════════════════════
if [ "$SHOW" = "1" ]; then
    banner
    r="$(role_of_file)"; h="$(hub_of_file)"; n="$(node_of_file)"
    echo -e "  ${BOLD}Role${NC}      : ${CYAN}${r:-inconnu}${NC}${n:+   noeud ${CYAN}${n}${NC}}"
    echo -e "  ${BOLD}IP de pont${NC}: ${CYAN}$(bridge_ip || echo '-')${NC}"
    echo -e "  ${BOLD}Inter-STP${NC} : ${CYAN}${h:--}${NC}"
    echo ""
    if [ -r "$INTEROP_CFG" ]; then
        echo -e "  ${BOLD}$(basename "$INTEROP_CFG")${NC}"
        awk '/^ *listen m3ua/{f=1} f&&/^ *(local-ip|listen)/{print "    "$0} f&&/^!/{f=0}' "$INTEROP_CFG"
    fi
    for f in "$CONF_DIR/osmo-stp.cfg" "$CONF_DIR/osmo-msc.cfg" "$CONF_DIR/osmo-bsc.cfg"; do
        [ -r "$f" ] || continue
        pc="$(awk '/^cs7 instance/{c=1} c && /^ *point-code /{print $2; exit}' "$f")"
        lo="$(awk '/asp asp-to-inter/{f=1} f&&/local-ip/{print $2; exit}' "$f")"
        re="$(awk '/asp asp-to-inter/{f=1} f&&/remote-ip/{print $2; exit}' "$f")"
        printf '  %-16s PC %-10s local-ip %-16s remote-ip %s\n' \
            "$(basename "$f")" "${pc:--}" "${lo:--}" "${re:--}"
    done
    echo ""
    [ -r "$WAN_CONF_FILE" ] && { echo -e "  ${BOLD}${WAN_CONF_FILE}${NC}"; sed 's/^/    /' "$WAN_CONF_FILE"; }
    exit 0
fi

if [ "$USE_DEFAULTS" = "1" ]; then
    [ -n "$HUB_IP" ]  || HUB_IP="$DEFAULT_HUB_IP"
    [ -n "$SELF_IP" ] || SELF_IP="$(bridge_ip || true)"
    if [ -z "$ROLE" ]; then
        if [ "$SELF_IP" = "$HUB_IP" ]; then ROLE="interstp"; else ROLE="operator"; fi
    fi
    if [ -z "$NODE" ] && [ "$ROLE" = "operator" ]; then
        NODE="$(node_of_file)"
        if [ -z "$NODE" ] && [ -r "$WAN_CONF_FILE" ]; then
            NODE="$(awk -F\" '/WAN_NODES=/{print $2}' "$WAN_CONF_FILE" 2>/dev/null \
                    | tr ' ' '\n' | awk -F: -v ip="$SELF_IP" '$2==ip{print $1; exit}')"
        fi
        [ -n "$NODE" ] || NODE=1
    fi
    if [ -z "$IPS" ] && [ "$ROLE" = "interstp" ] && [ -r "$WAN_CONF_FILE" ]; then
        IPS="$(awk -F\" '/WAN_NODES=/{print $2}' "$WAN_CONF_FILE" 2>/dev/null \
               | tr ' ' '\n' | awk -F: '{print $2}' | tr '\n' ' ')"
    fi
fi

banner
[ "$DRY" = "1" ] || [ "$(id -u)" -eq 0 ] \
```

```
    || { echo -e "${RED}Root requis (ecriture dans ${CONF_DIR} et sur l'interface).${NC}" >&2; exit 1; }

# ============================================================================
# 1. QUI EST CETTE MACHINE ?
# ============================================================================
# Le role decide de tout le reste : le hub ecoute et connait ses operateurs, un
# operateur s'attache et ne connait que le hub. Se tromper ici produit une
# machine qui ecoute sans que personne l'appelle, ou qui appelle une adresse
# qui n'ecoute pas.
if [ -z "$ROLE" ]; then
    def_role="$(role_of_file)"; [ -n "$def_role" ] || def_role="operator"
    if [ -t 0 ]; then
        echo -e "${BOLD}Role de cette machine${NC}"
        echo -e "    ${CYAN}1${NC}) inter-STP   - le hub SS7, PC 0.0.0, celui que tous appellent"
        echo -e "    ${CYAN}2${NC}) operateur   - osmo-operator-N, qui s'attache au hub"
        echo ""
        def_n=2; [ "$def_role" = "interstp" ] && def_n=1
        ans="$(_wan_ask "Role" "1 = inter-STP, 2 = operateur :" "$def_n")" || exit 1
        case "$ans" in
            1|inter|interstp|hub)   ROLE="interstp" ;;
            2|op|operator|operateur|operateur) ROLE="operator" ;;
            *) echo -e "${RED}reponse invalide : '$ans'${NC}" >&2; exit 2 ;;
        esac
    else
        ROLE="$def_role"
        echo -e "  ${YELLOW}Pas de terminal - role repris de ${ROLE_FILE} : ${ROLE}${NC}"
    fi
fi
case "$ROLE" in
    interstp|inter|hub) ROLE="interstp" ;;
    operator|op)        ROLE="operator" ;;
    *) echo -e "${RED}--role : interstp ou operator${NC}" >&2; exit 2 ;;
esac
[[ "$OPS" =~ ^[1-9]$ ]] || { echo -e "${RED}--op : un chiffre de 1 a 9${NC}" >&2; exit 2; }
[[ "$OP_INDEX" =~ ^[1-9]$ ]] || { echo -e "${RED}--op-index : un chiffre de 1 a 9${NC}" >&2; exit 2; }
# Un indice au-dela du compteur designe un operateur que ce noeud ne porte pas :
# le point code partirait dans une plage dont personne n'a la route.
[ "$OP_INDEX" -le "$OPS" ] \
    || { echo -e "${RED}--op-index ${OP_INDEX} : ce noeud n'annonce que ${OPS} operateur(s) (--op)${NC}" >&2; exit
     ↪  2; }

DETECTED="$(bridge_ip || true)"
echo -e "${BOLD}Role :${NC} ${CYAN}${ROLE}${NC}${DETECTED:+    IP de pont detectee : ${CYAN}${DETECTED}${NC}}"
echo ""

# ============================================================================
# 2A. INTER-STP - son IP de pont, et celles des osmo-operator-N
# ============================================================================
if [ "$ROLE" = "interstp" ]; then

    if [ -z "$SELF_IP" ]; then
        def="${DETECTED:-192.168.1.10}"
        SELF_IP="$(_wan_ask "Inter-STP" "IP de pont de CE hub :" "$def")" || exit 1
    fi
    need_ipv4 "$SELF_IP" "IP du hub"

    # Les IP des operateurs. Le hub n'en a pas besoin pour ROUTER - il ecoute et
    # les ASP viennent a lui, c'est le point code qui decide du reste. Il en a
    # besoin pour n'ouvrir son pare-feu qu'a eux, pour ecrire la table que
    # liront les noeuds, et pour que --status puisse dire QUI manque : sans la
    # liste, un noeud absent est indiscernable d'un noeud qui n'existe pas.
    WAN_IP=(); WAN_IND=(); WAN_NODE_LIST=(); i=0
    if [ -n "$IPS" ]; then
        for ip in ${IPS//,/ }; do
            i=$((i+1)); need_ipv4 "$ip" "IP de l'operateur $i"
            WAN_IP[$i]="$ip"; WAN_IND[$i]="$(wan_default_ind "$i")"; WAN_NODE_LIST+=("$i")
```

```
    done
else
    old_n=0
    [ -r "$WAN_CONF_FILE" ] && { wan_nodes_load "$WAN_CONF_FILE" >/dev/null 2>&1 && old_n="$WAN_NODE_COUNT"; }
    declare -A old_ip=()
    for k in ${WAN_NODE_LIST[@]+"${WAN_NODE_LIST[@]}"}; do old_ip[$k]="${WAN_IP[$k]}"; done
    WAN_IP=(); WAN_IND=(); WAN_NODE_LIST=()
    n="$(_wan_ask "Inter-STP" "Nombre d'osmo-operator (1-9) :" "$([ "$old_n" -ge 1 ] && echo "$old_n" || echo
     ↪  3)")" || exit 1
    [[ "$n" =~ ^[1-9]$ ]] || { echo -e "${RED}nombre invalide : $n${NC}" >&2; exit 2; }
    for i in $(seq 1 "$n"); do
        # Defaut aligne sur le plan du banc : osmo-operator-N a .1N sur le
        # meme /24 que le hub.
        def="${old_ip[$i]:-${SELF_IP%.*}.$((10 + i))}"
        ip="$(_wan_ask "osmo-operator-$i" "IP de pont de osmo-operator-$i :" "$def")" || exit 1
        need_ipv4 "$ip" "IP de osmo-operator-$i"
        WAN_IP[$i]="$ip"; WAN_IND[$i]="$(wan_default_ind "$i")"; WAN_NODE_LIST+=("$i")
    done
fi
NODES="${#WAN_NODE_LIST[@]}"
[ "$NODES" -ge 1 ] || { echo -e "${RED}aucun operateur saisi${NC}" >&2; exit 2; }
# Le hub ne porte aucun operateur : il n'est pas un noeud de la table. On note
# 0 - que wan_nodes_validate accepte - plutot qu'un numero qui le ferait
# ressembler a un noeud et lui volerait un point code.
WAN_NODE_ID=0; WAN_NODE_COUNT="$NODES"; WAN_OPS="$OPS"
wan_nodes_validate || exit 2

echo ""
echo -e "  Hub ${CYAN}${SELF_IP}:${PORT}${NC}   PC ${CYAN}0.0.0${NC}"
for i in "${WAN_NODE_LIST[@]}"; do
    printf '    osmo-operator-%s  %-15s  STP %s  MSC %s  BSC %s\n' \
        "$i" "${WAN_IP[$i]}" "1.${i}1.2" "1.${i}1.1" "1.${i}1.3"
done
echo ""

ensure_ip "$SELF_IP"

# ── La config du hub ──────────────────────────────────────────────
# local-ip 0.0.0.0 marche, mais annonce dans l'INIT SCTP toutes les adresses
# de la machine - NAT 10.0.2.15 et alias 172.20.x compris. Les noeuds tentent
# alors des chemins qui, vus d'eux, ne menent nulle part. Une adresse unique
# et juste vaut mieux qu'un multi-homing dont trois branches sur quatre sont
# mortes.
if [ "$DRY" = "1" ]; then
    echo -e "  [dry-run] ${INTEROP_CFG} : listen m3ua ${PORT} / local-ip ${SELF_IP}"
else
    mkdir -p "$CONF_DIR"
    # Regeneree a chaque passage, comme le fait start-interstp.sh : ce
    # fichier est entierement derive de (NODES, OPS): le garder tel quel
    # apres un changement de nombre d'operateurs laisserait des blocs "as"
    # pour des noeuds disparus et aucun pour le nouveau - un ASP qui arrive
    # sans "as" correspondant est refuse, sans que rien ne le dise.
    bash "$HERE/helpers/create_interop.sh" --wan "$NODES" "$OPS" "$INTEROP_CFG" || exit 1
    tmp="$(mktemp)"
    awk -v hub="$SELF_IP" -v port="$PORT" '
        /^ *listen m3ua/ { sub(/listen m3ua .*/, "listen m3ua " port); in_l=1; print; next }
        in_l && /^ *local-ip / { sub(/local-ip .*/, "local-ip " hub); print; next }
        in_l && /^(!|[a-z])/ { in_l=0 }
        { print }
    ' "$INTEROP_CFG" > "$tmp"
    # cat plutot que mv : sur l'ISO comme dans un conteneur ces fichiers
    # peuvent etre des points de montage, et rename() y echoue.
    cat "$tmp" > "$INTEROP_CFG"; rm -f "$tmp"
    echo -e "  ${GREEN}✓${NC} $(basename "$INTEROP_CFG") - ecoute ${CYAN}${SELF_IP}:${PORT}${NC}"
fi
```

```
    if [ "$DRY" = "1" ]; then
        echo -e "  [dry-run] ${WAN_CONF_FILE} : $(wan_nodes_spec)"
    else
        WAN_AUTO=0 wan_nodes_save "$WAN_CONF_FILE" \
            && echo -e "  ${GREEN}✓${NC} table ecrite dans ${CYAN}${WAN_CONF_FILE}${NC}"
        for i in "${WAN_NODE_LIST[@]}"; do
            bash "$HERE/network/firewall-wan.sh" "${WAN_IP[$i]}" "$OPS" >/dev/null 2>&1 || true
        done
        echo -e "  ${GREEN}✓${NC} pare-feu ouvert pour ${NODES} operateur(s)"
    fi
    write_role interstp "$SELF_IP" ""

    echo ""
    echo -e "  ${BOLD}Cote operateurs :${NC} sur chaque VM osmo-operator-N, lancer"
    echo -e "    ${CYAN}sudo ./set_stp_ip.sh --operator --node N --ip <son IP> --hub-ip ${SELF_IP}${NC}"

    if [ "$RESTART" = "1" ] && [ "$DRY" != "1" ]; then
        echo ""
        if [ -x "$HERE/launch/start-interstp.sh" ]; then
            "$HERE/launch/start-interstp.sh" --ip "$SELF_IP" --nodes "$NODES" --ops "$OPS" --port "$PORT" --no-ip
        else
            systemctl restart osmo-interstp 2>/dev/null \
                && echo -e "  ${GREEN}✓${NC} osmo-interstp relance" \
                || echo -e "  ${YELLOW}⚠${NC} relancez le hub a la main : ${CYAN}./launch/start-interstp.sh${NC}"
        fi
    else
        echo -e "  ${YELLOW}Le hub deja lance garde l'ancienne adresse :${NC}
         ↪  ${CYAN}./launch/start-interstp.sh${NC}"
    fi
    exit 0
fi

# ═══════════════════════════════════════════════════════════════════
# 2B. OSMO-OPERATOR-N - son IP de pont, et celle de l'inter-STP
# ═══════════════════════════════════════════════════════════════════
if [ -z "$NODE" ]; then
    def="$(node_of_file)"; [ -n "$def" ] || def=1
    NODE="$(_wan_ask "osmo-operator" "Numero de CE noeud operateur (1-9) :" "$def")" || exit 1
fi
[[ "$NODE" =~ ^[1-9]$ ]] || { echo -e "${RED}--node : un chiffre de 1 a 9${NC}" >&2; exit 2; }

if [ -z "$SELF_IP" ]; then
    def="${DETECTED:-192.168.1.$((10 + NODE))}"
    SELF_IP="$(_wan_ask "osmo-operator-$NODE" "IP de pont de CE noeud :" "$def")" || exit 1
fi
need_ipv4 "$SELF_IP" "IP de ce noeud"

if [ -z "$HUB_IP" ]; then
    def="$(hub_of_file)"
    # Un hub en 172.20.x (plan docker) ou 192.168.56.x (host-only) herite de
    # l'image n'existe pas sur le pont : le proposer comme defaut ferait
    # reconduire l'adresse meme qu'on est venu corriger.
    case "$def" in ""|172.20.*|192.168.5[6-9].*|192.168.6[0-3].*) def="${SELF_IP%.*}.10" ;; esac
    HUB_IP="$(_wan_ask "Inter-STP" "IP de pont de l'inter-STP :" "$def")" || exit 1
fi
need_ipv4 "$HUB_IP" "IP de l'inter-STP"

if [ "$SELF_IP" = "$HUB_IP" ]; then
    # Hub co-localise : le noeud 1 porte l'inter-STP sur sa propre adresse.
    # C'est valide tant que le hub ECOUTE (port 2908) et que l'ASP part d'un
    # autre port (2910) - le noeud ne s'attache pas a lui-meme, il s'attache
    # a un processus voisin. On ne refuse que si personne n'ecoute au hub.
    if pgrep -f 'osmo-stp -c /etc/osmocom/osmo-stp-interop\.cfg' >/dev/null 2>&1; then
        echo -e "  ${YELLOW}⚠${NC} inter-STP co-localise sur ${SELF_IP} (hub en service) - ASP vers le voisin, pas
         ↪  vers soi"
    else
```

```
        echo -e "${RED}Ce noeud et l'inter-STP portent la meme adresse (${SELF_IP}).${NC}" >&2
        echo -e "${RED}Aucun inter-STP n'ecoute ici : lancez start-interstp.sh, ou corrigez l'adresse.${NC}" >&2
        exit 2
    fi
fi
if [ "${SELF_IP%.*}" != "${HUB_IP%.*}" ]; then
    echo -e "  ${YELLOW}⚠${NC} ${SELF_IP} et ${HUB_IP} ne sont pas sur le meme /24 -"
    echo -e "     il faudra une route entre les deux pour que l'ASP monte."
fi

echo ""
echo -e "  ${BOLD}osmo-operator-${NODE}${NC}   ${CYAN}${SELF_IP}${NC}  ──M3UA──►  inter-STP
 ↪  ${CYAN}${HUB_IP}:${PORT}${NC}"
# Le <op> du plan est l'indice, pas le compteur. Ces trois point codes portaient
# un 1 en dur : ils annoncaient 1.<noeud>1.x quoi qu'on demande, pendant que
# set-node-id.sh en ecrivait un autre dans les fichiers.
echo -e "  STP ${CYAN}1.${NODE}${OP_INDEX}.2${NC}   MSC ${CYAN}1.${NODE}${OP_INDEX}.1${NC}   BSC
 ↪  ${CYAN}1.${NODE}${OP_INDEX}.3${NC}   (plan WAN 1.<noeud><op>.<role>)"
echo ""

ensure_ip "$SELF_IP"

# ── Point codes et adresses de l'ASP ──────────────────────────────────
# set-node-id.sh est le seul endroit qui sait tenir d'accord les TROIS fichiers
# (osmo-stp, osmo-msc, osmo-bsc) : son PC, son routing-key, et le PC du pair que
# chacun appelle. En rater un donne un coeur qui demarre et des appels qui
# n'aboutissent pas - d'ou la delegation plutot qu'un second sed ici.
# Son --op est un INDICE d'operateur, pas un nombre : lui passer le compteur
# revenait a lui demander le point code du DERNIER operateur du noeud.
SNI_ARGS=(--node "$NODE" --op "$OP_INDEX" --native
          --local-ip "$SELF_IP" --hub-ip "$HUB_IP" --conf-dir "$CONF_DIR")
[ "$DRY" = "1" ] && SNI_ARGS+=(--dry-run)
bash "$HERE/network/set-node-id.sh" "${SNI_ARGS[@]}" || exit 1

# ── La table WAN ──────────────────────────────────────────────────────
# On repart de celle qui existe pour ne pas perdre les autres noeuds : ce script
# ne connait que CE noeud et le hub, il n'a rien a dire sur les voisins.
WAN_IP=(); WAN_IND=(); WAN_NODE_LIST=()
[ -r "$WAN_CONF_FILE" ] && wan_nodes_load "$WAN_CONF_FILE" >/dev/null 2>&1 || true
if [ -z "${WAN_IP[$NODE]:-}" ]; then
    WAN_NODE_LIST+=("$NODE")
    WAN_IND[$NODE]="$(wan_default_ind "$NODE")"
fi
WAN_IP[$NODE]="$SELF_IP"
[ -n "${WAN_IND[$NODE]:-}" ] || WAN_IND[$NODE]="$(wan_default_ind "$NODE")"
WAN_NODE_ID="$NODE"; WAN_NODE_COUNT="${#WAN_NODE_LIST[@]}"; WAN_OPS="$OPS"

if [ "$DRY" = "1" ]; then
    echo -e "  [dry-run] ${WAN_CONF_FILE} : $(wan_nodes_spec)  (ce noeud : ${NODE})"
else
    if wan_nodes_validate; then
        wan_nodes_save "$WAN_CONF_FILE" \
            && echo -e "  ${GREEN}✓${NC} ${CYAN}${WAN_CONF_FILE}${NC} - ce noeud : ${NODE}"
    else
        echo -e "  ${YELLOW}⚠${NC} table WAN incoherente, non reecrite - ${CYAN}${WAN_CONF_FILE}${NC} inchange"
    fi
    bash "$HERE/network/firewall-wan.sh" "$HUB_IP" "$OPS" >/dev/null 2>&1 || true
fi
write_role operator "$HUB_IP" "$NODE"

echo ""
if [ "$RESTART" = "1" ] && [ "$DRY" != "1" ]; then
    if [ -x "$HERE/start-native.sh" ]; then
        echo -e "  Relance de la pile..."
        "$HERE/start-native.sh" --node "$NODE" || true
    else
```

```
        echo -e "  ${YELLOW}⚠${NC} start-native.sh absent - relancez la pile a la main."
    fi
else
    echo -e "  ${YELLOW}Les demons deja lances gardent l'ancienne adresse :${NC} relancez la pile"
    echo -e "    ${CYAN}sudo ./start-native.sh --node ${NODE}${NC}"
fi
echo -e "  ${BOLD}Verifier l'attachement, cote hub :${NC} ${CYAN}./launch/start-interstp.sh --status${NC}"
```

3.21 /opt/GSM/osmo-operator/ss7-console.py

307 octets, 10 lignes → 10 lignes

```
#!/usr/bin/env python3
# Raccourci vers la console SS7 : le code vit dans navigation/.
import os
import runpy
import sys

target = os.path.join(os.path.dirname(os.path.abspath(__file__)),
                      "navigation", "ss7-console.py")
sys.argv[0] = target
runpy.run_path(target, run_name="__main__")
```

3.22 /opt/GSM/osmo-operator/start-native.sh

2644 octets, 59 lignes → 59 lignes

```
#!/bin/bash
# ================================================================================
# start.sh - Prepare l'environnement Calypso puis lance run.sh
# ================================================================================
#
# Ce script ne demarre aucun demon.
#
# Il :
#   1. charge l'environnement ;
#   2. detecte le materiel / les binaires ;
#   3. choisit le mode (profil) ;
#   4. genere les fichiers mobile_*.cfg ;
#   5. exporte les variables necessaires ;
#   6. execute run.sh.
#
# Toute la logique GSM vit ensuite dans run.sh (et run_modules/).
#
# CHAINE DE CONFIGURATION - NE PAS LA CASSER
#     VAR=x ./start-native.sh   ← la ligne de commande gagne toujours
#       -> environment/load.env
#          -> paths / modes / domaines
#     puis exec run.sh avec le profil choisi.
# --------------------------------------------------------------------------------
set -uo pipefail

# ── [2026-08-14] GARDE "on doit etre dans Docker" RETIREE ──────────────────
# Elle exigeait /.dockerenv ET /etc/docker-entrypoint-cmd, et sortait en exit 1
# quand l'un manquait. Sur l'ISO il n'y a pas de Docker : les deux fichiers sont
# absents, la garde bloquait donc le seul lanceur utilisable la-bas. Retiree sur
# demande explicite - c'est le cas d'usage ISO qui prime.
#
# CE QU'ON PERD, et c'etait sa raison d'etre (constat du 12/08) : lance par
# erreur sur un HOTE qui a Docker, ce script repart sur un environnement a
# moitie construit - /opt/GSM/qosmo n'y existe pas, mais /tmp/osmo-nitb/logs,
# lui, se cree tout seul - et meurt sur un "run.sh introuvable" qui ne designe
```

```
# pas la vraie cause. Si ce symptome reapparait sur une machine avec Docker,
# c'est ca : la-bas le lanceur est ./launch/start-docker.sh (ou ./launch/start-nitb.sh), pas celui-ci.

HERE="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
cd "$HERE"

# ── --docker : le banc en conteneurs, par launch/start-docker.sh ──────────────
# Un seul point d'entree pour les deux lanceurs : `./start.sh --docker [args]`
# passe la main a start-docker.sh avec les autres arguments, tels quels. Le
# reste de ce script (startup_scripts/) est le lanceur NATIF et n'est pas lu.
_args=(); _docker=0
for _a in "$@"; do
    if [ "$_a" = --docker ]; then _docker=1; else _args+=("$_a"); fi
done
if [ "$_docker" = 1 ]; then
    exec bash "$HERE/launch/start-docker.sh" "${_args[@]+"${_args[@]}"}"
fi
unset _args _docker _a

# ── Le reste du lanceur : startup_scripts/*.sh, sourcees dans l'ordre (meme shell, memes variables) ──
for _f in "$HERE"/startup_scripts/*.sh; do
    . "$_f"
done
unset _f
```

3.23 /opt/GSM/osmo-operator/start.sh

identique à /opt/GSM/osmo-operator/start-native.sh

3.24 /opt/GSM/osmo-operator/test

10 octets, 3 lignes → 3 lignes

```
1 2
3 4
5
```

3.25 /opt/GSM/osmo-operator/tout-en-un.py

19496 octets, 391 lignes → 391 lignes

```
#!/usr/bin/env python3
"""tout-en-un.py — rassemble tous les fichiers texte (sources, scripts, configs,
docs, logs) d'un ou plusieurs dossiers dans UN fichier Markdown, raccourci SANS PERTE :

  * fichiers identiques (meme contenu) : une seule copie, les autres renvoient
    a la premiere ;
  * lignes consecutives identiques : « ligne  ×N » ;
  * lignes consecutives qui ne different que par des nombres (horodatage,
    compteur, fn...) : un gabarit ou les nombres constants restent en place et
    les nombres variables deviennent ⟨1⟩ ⟨2⟩..., suivi de la liste ordonnee des
    valeurs. On reconstruit chaque ligne en remettant les valeurs dans l'ordre.
  * codes couleur ANSI et retours chariot retires (seule perte, volontaire).

    ./tout-en-un.py [dossier...]      defaut : /opt/GSM/osmo-operator et le dernier /root/osmo-operator-*
    SORTIE=/chemin.md  EXT="sh py c"  (restreint aux extensions ; defaut : tout fichier texte)
    SEUIL=3 (taille mini d'un groupe)

  Sont ecartes : .git et caches, binaires (ELF, images, .pyc, archives), sauvegardes
  (.bak*, ~, .orig), LICENSE/COPYING et les sorties precedentes de ce script.
```

```
"""
import glob
import hashlib
import os
import re
import sys
import time

EXT = os.environ.get("EXT", "").split()             # vide : tout fichier texte
SEUIL = int(os.environ.get("SEUIL", "3"))
FORMAT = os.environ.get("FORMAT", "qmd")           # qmd (Quarto) ou md
SORTIE = os.environ.get("SORTIE") or "/root/osmo-operator-%s.%s" % (time.strftime("%Y%m%d-%H%M%S"), FORMAT)
AUTEUR = os.environ.get("AUTEUR", "Banc GSM émulé — banc-max")
IGNORES = {".git", "__pycache__", ".pytest_cache", ".mypy_cache", ".ruff_cache", "node_modules", ".venv", "venv",
          "rom", "roms",                            # dumps de ROM (DSP Calypso) : pas a nous, pas dans le dossier
           "osmo-operator-qmd"}                        # compilation du depot par un autre outil : tout en double
BINAIRES = set("png jpg jpeg gif bmp ico webp pdf zip gz tgz bz2 xz 7z tar bin elf o a so pyc pyo pyd "
               "wav mp3 ogg mp4 sqlite db img iso woff woff2 ttf otf".split())
SAUVEGARDES = re.compile(r"(~|\.orig|\.rej|\.swp|\.bak(-\w+)?)$")
TITRE_SORTIE = "title: \"Banc GSM émulé — dossier de run\"".encode()   # une sortie precedente de ce script
ANSI = re.compile(r"\x1b\[[0-9;]*[A-Za-z]")
TELNET = re.compile(rb"\xff[\xfb-\xfe].|\xff.")           # negociation telnet (IAC) des captures VTY
CONTROLE = re.compile("[\x00-\x08\x0b\x0c\x0e-\x1f\x7f\ufffd]")  # caracteres de commande, octets indecodables
NUM = re.compile(r"\d+")
# Preambule LaTeX du format pdf (xelatex). Sans lui : « Dimension too large » (framed
# mesure tout le bloc de code avant de le couper), lignes de code non repliees,
# diagrammes Mermaid plus larges que la page, glyphes manquants.
PDF_PREAMBULE = "".join("        %s\n" % l for l in r"""
% Code : police reduite, retour a la ligne (y compris dans les mots longs)
\usepackage{fvextra}
\fvset{fontsize=\small,breaklines,breakanywhere}
\RecustomVerbatimEnvironment{verbatim}{Verbatim}{fontsize=\small,breaklines,breakanywhere}
% Images (diagrammes Mermaid) : jamais plus larges que la ligne ni plus hautes que la page
\usepackage{adjustbox}
\usepackage{letltxmacro}
\LetLtxMacro\ORIGincludegraphics\includegraphics
\renewcommand{\includegraphics}[2][]{\adjustimage{#1,max width=\linewidth,max totalheight=.85\textheight}{#2}}
% Sans cadre : framed/snugshade mesure tout le bloc et plante ("Dimension too large")
\renewenvironment{Shaded}{\medskip}{\medskip}
% Glyphes absents de Latin Modern Roman -> DejaVu Sans
\usepackage{newunicodechar}
\newfontfamily\fallbackfont{DejaVu Sans}
\newunicodechar{↔}{{\fallbackfont ↔}}
\newunicodechar{⟨}{{\fallbackfont ⟨}}
\newunicodechar{⟩}{{\fallbackfont ⟩}}
\newunicodechar{≠}{{\fallbackfont ≠}}
\newunicodechar{∈}{{\fallbackfont ∈}}
\newunicodechar{ᵉ}{{\fallbackfont ᵉ}}
\newunicodechar{✓}{{\fallbackfont ✓}}
\newunicodechar{✗}{{\fallbackfont ✗}}
\newunicodechar{⚠}{{\fallbackfont ⚠}}
\newunicodechar{☎}{{\fallbackfont ☎}}
\newunicodechar{↩}{{\fallbackfont ↩}}
""".strip("\n").split("\n"))

LANG = {"sh": "bash", "bash": "bash", "py": "python", "md": "markdown", "qmd": "markdown", "mmd": "mermaid",
        "txt": "text", "log": "text", "c": "c", "h": "c", "cpp": "cpp", "cc": "cpp", "hpp": "cpp",
        "ini": "ini", "env": "ini", "service": "ini", "desktop": "ini", "toml": "toml", "yml": "yaml",
        "yaml": "yaml", "json": "json", "patch": "diff", "diff": "diff", "mak": "makefile", "mk": "makefile",
        "html": "html", "xml": "xml", "svg": "xml", "js": "javascript", "css": "css", "sql": "sql", "rs": "rust"}
NOMS_LANG = {"Makefile": "makefile", "GNUmakefile": "makefile", "Dockerfile": "dockerfile"}


def extension(nom):
    return nom.rsplit(".", 1)[-1].lower() if "." in nom[1:] else ""
```

```
def langage(chemin):
    """Langage de coloration : extension, puis nom du fichier, puis shebang."""
    nom = os.path.basename(chemin)
    e = extension(nom)
    if e in LANG:
        return LANG[e]
    if nom in NOMS_LANG:
        return NOMS_LANG[nom]
    try:
        premiere = open(chemin, "rb").readline().decode("utf-8", "replace")
    except OSError:
        premiere = ""
    if premiere.startswith("#!"):
        if "python" in premiere:
            return "python"
        if re.search(r"\b(ba|z|da|k)?sh\b", premiere):
            return "bash"
    return "text"


def texte_legitime(chemin, nom):
    """Fichier texte a nous : ni binaire, ni sauvegarde, ni licence, ni sortie de ce script."""
    if extension(nom) in BINAIRES or SAUVEGARDES.search(nom) or nom in ("LICENSE", "COPYING"):
        return False
    if nom.startswith("tout-en-un-"):
        return False
    try:
        with open(chemin, "rb") as f:
            debut = f.read(8192)
    except OSError:
        return False
    if b"\0" in debut:                          # ELF, images, .pyc, archives...
        return False
    if TITRE_SORTIE in debut:
        return False
    return True


def dossiers_par_defaut():
    runs = sorted(glob.glob("/root/osmo-operator-*"), key=os.path.getmtime)
    runs = [r for r in runs if os.path.isdir(r)]
    return ["/opt/GSM/osmo-operator"] + ([runs[-1]] if runs else [])


def fichiers(dossier):
    for racine, dirs, noms in os.walk(dossier):
        dirs[:] = sorted(d for d in dirs if d not in IGNORES)
        for n in sorted(noms):
            chemin = os.path.join(racine, n)
            if EXT and extension(n) not in EXT:
                continue
            if texte_legitime(chemin, n):
                yield chemin


LIGNE_MAX = int(os.environ.get("LIGNE_MAX", "1000"))


def couper(l):
    if len(l) <= LIGNE_MAX:
        return [l]
    return [l[i:i + LIGNE_MAX] + ("↩" if i + LIGNE_MAX < len(l) else "") for i in range(0, len(l), LIGNE_MAX)]


def compacter(lignes):
```

```
    """Rend (lignes de sortie, nb de groupes compactes)."""
    out, groupes, i, n = [], 0, 0, len(lignes)
    while i < n:
        cle = NUM.sub("\0", lignes[i])
        j = i + 1
        while j < n and NUM.sub("\0", lignes[j]) == cle:
            j += 1
        taille = j - i
        if taille < SEUIL:
            out.extend(lignes[i:j])
            i = j
            continue
        groupes += 1
        if "\0" not in cle:                      # repetition exacte
            out.append("%s  ×%d" % (lignes[i], taille))
            i = j
            continue
        valeurs = [NUM.findall(l) for l in lignes[i:j]]
        nb = len(valeurs[0])
        varie = [len({v[k] for v in valeurs}) > 1 for k in range(nb)]
        # gabarit : nombres constants remis en place, variables numerotes ⟨k⟩
        morceaux, k, idx = cle.split("\0"), 0, 0
        gab = morceaux[0]
        for m in morceaux[1:]:
            if varie[k]:
                idx += 1
                gab += "⟨%d⟩" % idx
            else:
                gab += valeurs[0][k]
            gab += m
            k += 1
        out.append("%s  ×%d" % (gab, taille))
        if idx:
            tuples = [",".join(v[k] for k in range(nb) if varie[k]) for v in valeurs]
            # lignes de valeurs de ~100 colonnes
            ligne, courant = [], "    ⟨⟩ ="
            for t in tuples:
                if len(courant) + len(t) + 3 > 110:
                    ligne.append(courant)
                    courant = "       "
                courant += " (" + t + ")"
            ligne.append(courant)
            out.extend(ligne)
        i = j
    return out, groupes


def synthese(tous):
    """Run, echelle, echecs, bilan de couverture : lus dans verdict.txt / couverture.txt."""
    lignes, vus = [], set()
    for f in tous:
        nom = os.path.basename(f)
        if nom not in ("verdict.txt", "couverture.txt"):
            continue
        run = os.path.basename(os.path.dirname(f))
        if (run, nom) in vus:                # meme run copie a deux endroits
            continue
        vus.add((run, nom))
        try:
            txt = ANSI.sub("", open(f, "rb").read().decode("utf-8", "replace"))
        except OSError:
            continue
        for l in txt.splitlines():
            l = re.sub(r"\s+", " ", l.strip())
            if l.startswith("ELEMENT MAX"):
                lignes.append((0, "| %s | échelle | %s |" % (run, l)))
```

```
            elif l.startswith("bilan :"):
                lignes.append((2, "| %s | couverture | %s |" % (run, l[8:])))
            elif l.startswith("couverture couche 1"):
                lignes.append((1, "| %s | mode, date | %s |" % (run, l.split("—", 1)[-1].strip())))
            else:
                m = re.match(r"(\d+) (\S+) (ECHEC|SAUTE) ?(.*)", l)
                if m:
                    lignes.append((3, "| %s | barreau %s %s | %s %s |" % (run, m.group(1), m.group(2), m.group(3),
↪   m.group(4))))
    return [l for _, l in sorted(lignes, key=lambda x: x[0])]


MERMAID_OUVRE = "```{mermaid}" if FORMAT == "qmd" else "```mermaid"


def rendre_mmd(lignes):
    return "%s\n%s\n```" % (MERMAID_OUVRE, "\n".join(lignes))


def rendre_md(lignes):
    out, dans_bloc, cloture = [], False, ""
    i = 0
    if lignes and lignes[0].strip() == "---":           # front matter YAML : montre en bloc, pas interprete
        j = next((k for k in range(1, len(lignes)) if lignes[k].strip() in ("---", "...")), None)
        if j:
            out.append("```yaml"); out.extend(lignes[1:j]); out.append("```")
            i = j + 1
    while i < len(lignes):
        l = lignes[i]; i += 1
        m = re.match(r"^(\s*)(`{3,}|~{3,})(.*)$", l)
        if m and not dans_bloc:
            dans_bloc, cloture = True, m.group(2)
            info = m.group(3).strip()
            if info.startswith("{mermaid}") or info.startswith("mermaid"):
                l = m.group(1) + MERMAID_OUVRE
            elif info.startswith("{"):                    # ```{r ...} ou {.bash filename=...} : langage seul,
                l = m.group(1) + m.group(2) + info.strip("{}").split()[0].split(",")[0].lstrip(".")  # ni
↪   execution ni flottant
            out.append(l); continue
        if m and dans_bloc and m.group(2)[0] == cloture[0] and len(m.group(2)) >= len(cloture) and not
         ↪  m.group(3).strip():
            dans_bloc = False; out.append(l); continue
        if not dans_bloc:
            h = re.match(r"^(#{1,6})\s", l)
            if h:
                l = "#" * min(6, len(h.group(1)) + 3) + l[len(h.group(1)):]
            elif l.strip() in ("---", "..."):            # pas de bloc YAML ni de regle au milieu du dossier
                l = "* * *"
        out.append(l)
    if dans_bloc:
        out.append(cloture)
    return "\n".join(out)


def main():
    dossiers = sys.argv[1:] or dossiers_par_defaut()
    vus, sections, table = {}, [], []
    total_in = total_out = 0
    # Ordre : les resultats des tests (.txt .md .mmd : verdicts, couverture, rapports, grafcets)
    # d'abord, les fichiers du banc (sources, scripts, configs...) ensuite, les .log en dernier
    CATEGORIES = (("Verdict et couverture", ()), ("Resultats des tests", ("txt", "md", "mmd", "tsv", "csv")),
                  ("Fichiers", None), ("Logs", ("log",)))          # Fichiers : tout le reste
    EN_TETE = ("verdict.txt", "couverture.txt")     # tout en haut, dans cet ordre
    tous = []
    for d in dossiers:
        if not os.path.isdir(d):
```

```
        table.append("| (absent) %s | | | | |" % d)
        continue
    tous.extend(fichiers(d))
def rang(f):
    if os.path.basename(f) in EN_TETE:
        return 0
    e = extension(os.path.basename(f))
    for i, (_, exts) in enumerate(CATEGORIES):
        if exts and e in exts:
            return i
    return 2                                    # Fichiers
categorie_courante = None
def cle(f):
    r = rang(f)
    return (r, EN_TETE.index(os.path.basename(f)) if r == 0 else 0, tous.index(f))
for f in sorted(tous, key=cle):
    if rang(f) != categorie_courante:
        categorie_courante = rang(f)
        nom = CATEGORIES[categorie_courante][0] if categorie_courante < len(CATEGORIES) else "Autres"
        sections.append("## %s\n" % nom)
    if True:
        try:
            brut = open(f, "rb").read()
        except OSError as e:
            table.append("| %s | | | | illisible : %s |" % (f, e))
            continue
        texte = ANSI.sub("", TELNET.sub(b"", brut).decode("utf-8", "replace")).replace("\r", "")
        texte = CONTROLE.sub("", texte)
        lignes = texte.split("\n")
        if lignes and lignes[-1] == "":
            lignes.pop()
        # lignes demesurees (blob base64, minifie...) : coupees en morceaux de LIGNE_MAX
        # termines par ↩ ; sinon xelatex mesure la ligne entiere et plante (Dimension too large)
        lignes = [m for l in lignes for m in couper(l)]
        h = hashlib.sha1(texte.encode()).hexdigest()
        total_in += len(brut)
        if h in vus:
            sections.append("### %s\n\nidentique à %s\n" % (f, vus[h]))
            continue
        vus[h] = f
        ext = extension(os.path.basename(f))
        if ext in ("md", "qmd", "mmd"):
            # Rendu, pas cite : les .mmd deviennent des diagrammes Mermaid, les .md
            # sont inclus tels quels (titres retrogrades sous le titre du fichier,
            # blocs mermaid rendus, blocs {r}/{python} neutralises). Pas de compactage.
            total_out += len(texte) + 1
            sections.append("### %s\n\n%d octets, %d lignes\n\n%s\n"
                            % (f, len(brut), len(lignes), rendre_md(lignes) if ext in ("md", "qmd") else
                             ↪  rendre_mmd(lignes)))
            continue
        compact, groupes = compacter(lignes)
        total_out += sum(len(l) + 1 for l in compact)
        note = "%d groupes compactés" % groupes if groupes else ""
        # cloture plus longue que toute suite de ` du contenu : un ``` dans un
        # .py ou un .md ne referme plus le bloc (pandoc lisait alors la suite
        # comme du Markdown : « Could not fetch resource »)
        plus_long = max((len(m) for m in re.findall(r"`+", "\n".join(compact))), default=0)
        cloture = "`" * max(3, plus_long + 1)
        sections.append("### %s\n\n%d octets, %d lignes → %d lignes%s\n\n%s%s\n%s\n%s\n"
                        % (f, len(brut), len(lignes), len(compact), (" (%s)" % note) if note else "",
                           cloture, langage(f), "\n".join(compact), cloture))
runs = sorted({os.path.basename(os.path.dirname(f)) for f in tous if os.path.basename(f) == "verdict.txt"})
with open(SORTIE, "w") as o:
    o.write("---\n")
    o.write("title: \"Banc GSM émulé — dossier de run\"\n")
    o.write("subtitle: \"%s\"\n" % (", ".join(runs) if runs else ", ".join(dossiers)))
```

```
        o.write("author: \"%s\"\n" % AUTEUR)
        o.write("date: \"%s\"\n" % time.strftime("%Y-%m-%d %H:%M"))
        o.write("lang: fr\n")
        if FORMAT == "qmd":
            o.write("engine: markdown\n")
            o.write("code-annotations: false\n")  # un "<100>" dans un commentaire n'est pas une annotation     #
↪  pandoc seul : knitr voyait des ```{r} dans les rapports inclus et s'arretait
        if FORMAT == "qmd":
            o.write("format:\n  html:\n    toc: true\n    toc-depth: 3\n    toc-location: left\n"
                    "    number-sections: true\n    embed-resources: true\n    theme: cosmo\n"
                    "    code-overflow: wrap\n    fontsize: 0.9em\n")
            o.write("  pdf:\n    toc: true\n    toc-depth: 3\n    number-sections: true\n"
                    "    shift-heading-level-by: -1\n"      # ## Synthese = section 1, pas 0.1
                    "    papersize: a4\n    geometry: margin=2cm\n    fontsize: 10pt\n"
                   "    monofont: DejaVu Sans Mono\n"      # trace de boites, ✓ ✗ ⟨⟩ ⚠ absents de Latin Modern Mono
                    "    include-in-header:\n      text: |\n" + PDF_PREAMBULE)
        o.write("---\n\n")
        o.write("## Synthèse\n\n")
        o.write("| run | élément | valeur |\n|---|---|---|\n")
        lignes_syn = synthese(tous)
        o.write("\n".join(lignes_syn) + "\n\n" if lignes_syn else "| (pas de verdict.txt) | | |\n\n")
        o.write("| dossier de run | contenu | |\n|---|---|---|\n")
        o.write("| Verdict et couverture | échelle des barreaux et tableau de couverture couche 1 | |\n")
        o.write("| Résultats des tests | captures VTY, diagnostics, rapports pytest, grafcets (.txt .md .mmd) |
↪  |\n")
        o.write("| Fichiers | sources, scripts, configs et docs du banc (tout fichier texte) | |\n")
        o.write("| Logs | journaux (.log), en dernier | |\n\n")
        o.write("::: {.callout-note collapse=\"true\"}\n## Méthode\n\n" if FORMAT == "qmd" else "### Méthode\n\n")
        o.write("Dossiers : %s. Extensions : %s. Entrée %d Ko, sortie %d Ko.\n\n"
                % (", ".join(dossiers), " ".join(EXT) if EXT else "tous les fichiers texte (hors .git, caches, "
                   "binaires, sauvegardes, LICENSE et sorties précédentes)", total_in // 1024, total_out // 1024))
        o.write("Compactage sans perte : fichiers identiques cités une fois ; lignes consécutives identiques "
                "« ×N » ; lignes ne différant que par des nombres : gabarit avec ⟨k⟩ puis la liste ordonnée "
                "des valeurs (k-uplets), chaque ligne se reconstruit en remettant les valeurs dans l'ordre. "
                "Seuls les codes couleur, la négociation telnet des captures VTY et les caractères de commande "
                "sont retirés ; les lignes de plus de %d caractères sont coupées en morceaux terminés par ↩. Les
                 ↪  .md sont inclus tels quels (titres rétrogradés) et les .mmd rendus en "
                "diagrammes Mermaid, sans compactage.\n" % LIGNE_MAX)
        o.write(":::\n\n" if FORMAT == "qmd" else "\n")
        for t in table:                      # ne reste que les dossiers absents / fichiers illisibles
            o.write(t.strip("| ").split(" |")[0] + "\n")
        o.write("\n")
        o.write("\n".join(sections))
    print("%s : %d fichiers (%d uniques), %d Ko -> %d Ko" % (SORTIE, len(tous), len(vus), total_in // 1024,
                                                            os.path.getsize(SORTIE) // 1024))


if __name__ == "__main__":
    main()
```

3.26 /opt/GSM/osmo-operator/update.sh

61256 octets, 1095 lignes → 1095 lignes

```
#!/bin/bash
# ============================================================================
# update.sh - l'animation SMS de l'ouverture de session. Rien d'autre.
#
# [2026-08-27] Ce fichier ne faisait pas ce que son nom dit : il posait un
# osmo-sync.sh qui, a CHAQUE demarrage, effacait puis reclonait osmo-operator et
# osmo-egprs-web depuis GitHub, resynchronisait qosmo-grgsm, installait socat a
# coups d'apt, et rearmait un declencheur sur la console. Trois consequences :
#
#   - ce qui tournait sur la machine n'etait plus ce que l'ISO portait, mais ce
```

```
#     que GitHub avait ce matin-la ;
#   - sans reseau au demarrage, les arbres effaces ne revenaient pas ;
#   - un paquet reinstalle a chaque boot, c'est un boot qui depend du reseau.
#
# Tout cela appartient a la CONSTRUCTION, pas au demarrage : c'est build-iso.sh
# qui embarque desormais les trois depots AVEC leur .git (et qosmo-grgsm avec son
# build/ compile), installe socat/nc/tcpdump/git dans le rootfs, et pose le
# service du dashboard. Une machine qui demarre n'a plus rien a aller chercher.
#
# Ce qu'il reste ici est ce qui ne peut pas etre fait a la construction, ou
# qui doit rattraper les machines DEJA installees :
#
#   - l'animation, qui a besoin d'un terminal et de quelqu'un devant ;
#   - la repose des ICONES DU BUREAU (voir le bloc dedie plus bas) : les
#     raccourcis dessines tombaient en page blanche generique, et le
#     correctif de build-iso.sh ne touche que les ISO a venir. Idempotent.
#
# Usage :
#   sudo ./update.sh            repose les icones, puis joue l'animation
#   sudo ./update.sh --quiet    repose les icones, sans l'animation (code 0)
#
# Sur l'ISO, /etc/profile.d/99-osmo-sms.sh l'appelle une fois par demarrage,
# apres le choix du clavier (ordre alphabetique de /etc/profile.d).
# ═══════════════════════════════════════════════════════════════════════════
set -u

# Meme raison que dans build.sh : ce script est appele depuis
# /etc/profile.d et depuis les lanceurs du bureau, ou le repertoire courant
# n'est pas le sien. On se place chez soi avant de toucher a quoi que ce soit.
cd "$(dirname "$(readlink -f "$0")")" || true

case "${1:-}" in
    -h|--help) sed -n '2,40p' "$0" | sed 's/^# \{0,1\}//'; exit 0 ;;
esac

# ── LES ICONES DU BUREAU, REPOSEES A CHAQUE DEMARRAGE ──────────────────────
# [2026-08-31] Les trois raccourcis dessines - "Lancer le banc GSM",
# "multi-operator", "Tutoriel" - s affichaient en PAGE BLANCHE generique sur le
# bureau. Les SVG etaient pourtant valides et bien poses dans
# /usr/share/icons/hicolor/scalable/apps/.
#
# La cause n est pas le fichier, c est la RESOLUTION DU NOM. "Icon=osmo-launch"
# n est pas un chemin : c est un nom que GTK va chercher dans le thème, via
# /usr/share/icons/hicolor/icon-theme.cache. Ce cache datait d AVANT l arrivee
# des icones - releve sur le banc :
#     strings /usr/share/icons/hicolor/icon-theme.cache | grep -c osmo  ->  0
#     cache 17:22:12   ·   icones 17:28:07
# Zero entree sur trois. Et un nom d icone qui ne resout pas ne provoque aucune
# erreur : GNOME/DING le remplace EN SILENCE par la page blanche. "Supplements"
# gardait la sienne parce que "system-software-install" vient de Yaru, deja
# dans le cache depuis l installation du systeme.
#
# build-iso.sh pose desormais le correctif dans l image. Le meme correctif est
# REJOUE ICI parce qu une machine deja installee ne repasse pas par la
# construction : sans ce bloc, elle garderait ses pages blanches jusqu a la
# prochaine ISO. Tout y est idempotent - on peut le rejouer a chaque session.
osmo_reposer_icones() {
    [ "$(id -u)" -eq 0 ] || return 0        # sans droits : on ne casse rien
    local src=/opt/GSM/osmo-operator/data
    local dst=/usr/share/osmo-operator/icons
    [ -d "$src" ] || return 0

    install -d "$dst" 2>/dev/null || return 0
    local ic
    for ic in osmo-launch osmo-multi osmo-tutorial; do
        [ -f "$src/$ic.svg" ] || continue
```

```
    cp -f "$src/$ic.svg" "$dst/$ic.svg" 2>/dev/null || continue
    chmod 644 "$dst/$ic.svg" 2>/dev/null || true
    # La copie du thème sert au MENU des applications, qui lui resout
    # encore par nom ; le bureau, lui, passe par le chemin absolu.
    install -d /usr/share/icons/hicolor/scalable/apps 2>/dev/null || true
    cp -f "$src/$ic.svg" \
          "/usr/share/icons/hicolor/scalable/apps/$ic.svg" 2>/dev/null || true
done
command -v gtk-update-icon-cache >/dev/null 2>&1 && \
    gtk-update-icon-cache -f -q /usr/share/icons/hicolor 2>/dev/null || true

# Icon= EN CHEMIN ABSOLU : plus de thème, plus de cache, plus de silence.
# Les raccourcis vivent en trois endroits (le menu, et les deux noms du
# bureau - Bureau en francais, Desktop en anglais, DING lit celui que la
# locale designe) ; les trois doivent porter la meme ligne.
# [2026-09-05] TOUS LES COMPTES, PAS « osmocom » EN DUR. La liste nommait
# /home/osmocom, le compte de la cle ; l installeur, lui, cree le compte que
# l utilisateur a choisi. Sur une machine installee, aucune de ces lignes ne
# designait donc son bureau, et les icones y gardaient l ancien Icon= (ou
# n etaient jamais reparees). Meme confusion racine/session que les blocs
# d addition.sh corriges le meme jour.
local f b d
for d in /usr/share/applications /root/Bureau /root/Desktop \
         /home/*/Bureau /home/*/Desktop; do
    [ -d "$d" ] || continue
    for b in osmo-launch osmo-multi osmo-tutorial; do
        f="$d/$b.desktop"
        [ -f "$f" ] || continue
        grep -q "^Icon=$dst/$b.svg\$" "$f" && continue
        sed -i "s|^Icon=.*|Icon=$dst/$b.svg|" "$f" 2>/dev/null || true
    done
done

# Un .desktop du bureau ne s affiche avec son nom et son icone que s il est
# executable ET porteur de metadata::trusted. Cet attribut vit dans les
# metadonnees gvfs de la SESSION, jamais dans le fichier : il se repose
# donc ici, sous la session, et pas a la construction.
# metadata::trusted vit dans les metadonnees gvfs DU PROPRIETAIRE : un `gio
# set` lance par root le pose pour root, jamais pour la session de
# l utilisateur. On repasse donc par son compte et son bus (runuser), comme
# _trust_desktop() d addition.sh - sans quoi les icones du compte installe
# restaient en pastille « fichier non fiable ».
local _own _uid _bus
for d in "$HOME/Bureau" "$HOME/Desktop" /root/Bureau /root/Desktop \
         /home/*/Bureau /home/*/Desktop; do
    [ -d "$d" ] || continue
    _own="$(stat -c '%U' "$d" 2>/dev/null)"; [ -n "$_own" ] || _own=root
    _uid="$(id -u "$_own" 2>/dev/null)" || continue
    _bus="/run/user/$_uid/bus"
    for f in "$d"/*.desktop; do
        [ -f "$f" ] || continue
        chown "$_own" "$f" 2>/dev/null || true
        chmod +x "$f" 2>/dev/null || true
        if [ -S "$_bus" ] && command -v runuser >/dev/null 2>&1; then
            runuser -u "$_own" -- env XDG_RUNTIME_DIR="/run/user/$_uid" \
                DBUS_SESSION_BUS_ADDRESS="unix:path=$_bus" \
                gio set -t string "$f" metadata::trusted true 2>/dev/null || true
        else
            gio set -t string "$f" metadata::trusted true 2>/dev/null || true
        fi
    done
    # DING ne relit pas les metadonnees a chaud : toucher le repertoire le
    # force a rebalayer, sinon la page blanche reste jusqu au login suivant.
    touch "$d" 2>/dev/null || true
done
update-desktop-database /usr/share/applications 2>/dev/null || true
```

```
    return 0
}
osmo_reposer_icones

# ── LE MULTI NE DOIT PAS SE TUER EN RECYCLANT LE NATIF ──────────────────────
# [2026-09-05] osmo-multi.service portait `Requires=osmo-banc.service`. Or
# start-multi.sh applique « un clic = un banc neuf » : il ARRETE osmo-banc a
# chaque lancement, puis le relance. Requires= propage l arret explicite d une
# dependance a l unite qui en depend -- ce stop du natif arretait osmo-multi
# LUI-MEME en plein ExecStart (code=killed, status=15/TERM). Le correctif est
# `Wants=` : meme ordre au boot (After=), mais plus de propagation de l arret.
osmo_corriger_multi_natif() {
    [ "$(id -u)" -eq 0 ] || return 0
    local unit=/etc/systemd/system/osmo-multi.service
    [ -f "$unit" ] || return 0
    grep -q '^Requires=osmo-banc\.service$' "$unit" || return 0
    sed -i 's/^Requires=osmo-banc\.service$/Wants=osmo-banc.service/' "$unit"
    if command -v systemctl >/dev/null 2>&1 && [ -d /run/systemd/system ]; then
        systemctl daemon-reload 2>/dev/null || true
    fi
    echo "[OK] osmo-multi.service : Requires=osmo-banc -> Wants (plus de suicide au recyclage du natif)."
    return 0
}
osmo_corriger_multi_natif

# ── LE MOBILE QEMU DOIT CHERCHER LA CELLULE SUR L ARFCN DU PLAN ─────────────
# [2026-10-24] Dans les conteneurs du multi, la BSC emet sur PLAN_ARFCN (1,
# globals.conf ARFCN="1") mais le mobile QEMU de l instance 1 lit
# c54x_exe/mobile_pont.cfg tel quel : `stick 514`, le gabarit. run.sh ne
# reecrit le stick que pour les instances 2+ : le mobile cherchait 514 et
# ne trouvait jamais la cellule. On aligne le gabarit sur le plan radio
# (radio-plan.env, puis ARFCN de globals.conf, puis 1). Idempotent.
osmo_aligner_arfcn_qemu() {
    local cfg arfcn
    arfcn="$( . /etc/osmocom/radio-plan.env 2>/dev/null; echo "${PLAN_ARFCN:-}")"
    [ -n "$arfcn" ] || arfcn="$(sed -n 's/^ARFCN="\{0,1\}\([0-9]*\)"\{0,1\}$/\1/p' \
        "$(dirname "$(readlink -f "$0")")/globals.conf" 2>/dev/null | tail -1)"
    [[ "$arfcn" =~ ^[0-9]+$ ]] || arfcn=1
    for cfg in /opt/GSM/c54x_exe/mobile_pont.cfg; do
        [ -w "$cfg" ] || continue
        grep -qE "^[[:space:]]*stick ${arfcn}\$" "$cfg" \
            && ! grep -qE "^[[:space:]]*no (p|e)-gsm\$" "$cfg" && continue
        # [2026-10-24] le gabarit ne supporte que DCS (no p-gsm/no e-gsm) : un
        # stick sur une ARFCN 900 non supportee fait boucler gsm322_cs_powerscan
        # a 100 % CPU (« goto again » sans fin), le mobile ne dit plus rien.
        sed -i -e "s|^\([[:space:]]*\)stick [0-9]*\$|\1stick ${arfcn}|" \
            -e "s|^\([[:space:]]*\)no \([pe]-gsm\)\$|\1\2|" "$cfg"
        echo "[OK] $cfg : stick ${arfcn} (ARFCN du plan radio), bandes P/E-GSM actives."
    done
    return 0
}
osmo_aligner_arfcn_qemu

# ── L OUTIL DE PEINTURE (overlay) SUR LES MACHINES DEJA INSTALLEES ──────────
# [2026-09-05] tools/overlay-draw.py + son lanceur. build-iso l embarque ; ici
# on rattrape les machines deja posees : symlink, icone, chemin absolu. Idempotent.
osmo_poser_peinture() {
    [ "$(id -u)" -eq 0 ] || return 0
    local d=/opt/GSM/osmo-operator
    local py="$d/tools/overlay-draw.py"
    [ -f "$py" ] || return 0
    chmod 755 "$py" 2>/dev/null || true
    ln -sf "$py" /usr/local/bin/overlay-draw 2>/dev/null || true
    if [ -f "$d/data/osmo-paint.svg" ]; then
        install -d /usr/share/osmo-operator/icons \
                   /usr/share/icons/hicolor/scalable/apps 2>/dev/null || true
```

```
        cp -f "$d/data/osmo-paint.svg" /usr/share/osmo-operator/icons/osmo-paint.svg 2>/dev/null || true
        cp -f "$d/data/osmo-paint.svg" /usr/share/icons/hicolor/scalable/apps/osmo-paint.svg 2>/dev/null || true
        chmod 644 /usr/share/osmo-operator/icons/osmo-paint.svg 2>/dev/null || true
    fi
    if [ -f "$d/data/desktop/osmo-paint.desktop" ]; then
        install -m644 "$d/data/desktop/osmo-paint.desktop" \
                /usr/share/applications/osmo-paint.desktop 2>/dev/null || true
        sed -i "s|^Icon=.*|Icon=/usr/share/osmo-operator/icons/osmo-paint.svg|" \
            /usr/share/applications/osmo-paint.desktop 2>/dev/null || true
    fi
    command -v gtk-update-icon-cache >/dev/null 2>&1 && \
        gtk-update-icon-cache -f -q /usr/share/icons/hicolor 2>/dev/null || true
    update-desktop-database /usr/share/applications 2>/dev/null || true
    return 0
}
osmo_poser_peinture

# ── LE TELEPHONE postmarketOS SUR LES MACHINES DEJA INSTALLEES ──────────────
# [2026-09-07] Le telephone du banc n est plus Android (Waydroid, abandonne)
# mais une VM postmarketOS/Phosh : tools/osmo-pmos.sh. L ISO et le supplement
# (addition.sh, par tools/osmo-extras-install.sh) posent son lanceur et son
# icone ; une machine deja installee, elle, ne repasse ni par l un ni par
# l autre - d ou ce rattrapage. On enleve au passage l ancien lanceur Waydroid,
# qui pointerait sur un script disparu du depot. Idempotent.
osmo_poser_pmos() {
    [ "$(id -u)" -eq 0 ] || return 0
    local d=/opt/GSM/osmo-operator
    [ -f "$d/tools/osmo-pmos.sh" ] || return 0
    chmod 755 "$d/tools/osmo-pmos.sh" 2>/dev/null || true
    cat > /usr/local/bin/osmo-pmos <<'PM'
#!/bin/bash
# osmo-pmos - le telephone postmarketOS du banc (voir tools/osmo-pmos.sh).
exec "${OSMO_REPO:-/opt/GSM/osmo-operator}/tools/osmo-pmos.sh" "$@"
PM
    chmod 755 /usr/local/bin/osmo-pmos 2>/dev/null || true
    cat > /usr/share/applications/osmo-pmos.desktop <<'PMD'
[Desktop Entry]
Type=Application
Name=Telephone (postmarketOS)
Comment=Le telephone du banc - Phosh, ModemManager, appels et SMS reels
Exec=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py --tty --root --title "Telephone (postmarketOS)" --
 ↪  /usr/local/bin/osmo-pmos up
Icon=phone
Terminal=false
Categories=Network;Telephony;
Keywords=postmarketos;phosh;telephone;sms;appel;modem;
Actions=Etat;Arreter;

[Desktop Action Etat]
Name=Etat du telephone
Exec=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py --root --keep-open --title "Etat du telephone" --
 ↪  /usr/local/bin/osmo-pmos status

[Desktop Action Arreter]
Name=Arreter le telephone
Exec=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py --root --close-on-exit --title "Arret du telephone" --
 ↪  /usr/local/bin/osmo-pmos stop
PMD
    chmod 644 /usr/share/applications/osmo-pmos.desktop 2>/dev/null || true
    # [2026-09-07] LES DEUX LANCEURS PAR pmbootstrap vivent dans le depot :
    # tools/osmo-pmos-qemu.sh (la VM, format smartphone|tablette en argument)
    # et tools/osmo-pmos-setup.sh (le modem et la voix, VM demarree). Le
    # patch pmbootstrap qui va avec (port serie du modem, cartes son, taille
    # d ecran) est patches/pmbootstrap-osmo-bench-qemu.patch - il s applique
    # dans le pmbootstrap de l utilisateur, pas ici.
    local s
```

```
    for s in qemu setup; do
        [ -f "$d/tools/osmo-pmos-$s.sh" ] \
            && install -m 755 "$d/tools/osmo-pmos-$s.sh" "/usr/local/bin/osmo-pmos-$s" 2>/dev/null
    done
    # [2026-09-07] DEUX FORMATS, DEUX ICONES. Phosh se met en page d apres
    # l ecran qu on lui donne : haut et etroit, c est un telephone ; large,
    # c est une tablette - meme image, meme session. C est la carte graphique
    # de QEMU qui porte cette taille (OSMO_PMOS_RES, lu par le patch
    # pmbootstrap), et le systeme demarre dedans : le format se choisit donc
    # AU LANCEMENT et ne change pas sous une session deja ouverte. D ou deux
    # entrees plutot qu un reglage.
    local f n r
    for f in smartphone:720x1440 tablette:1280x800; do
        n="${f%%:*}"; r="${f#*:}"
        cat > "/usr/share/applications/osmo-pmos-$n.desktop" <<PMD
[Desktop Entry]
Type=Application
Name=postmarketOS - $n (banc)
Comment=Le telephone du banc, modem et voix branches tout seuls, en format $n ($r) - modem branche sur le banc GSM
Exec=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py --tty --title "Telephone du banc (VM)" -- env OSMO_PMOS_RES=$r
 ↪  /usr/local/bin/osmo-pmos-qemu
Icon=$([ "$n" = tablette ] && echo video-display || echo phone)
Terminal=false
Categories=Network;Telephony;System;
Keywords=postmarketos;pmos;qemu;modem;gsm;telephone;$n;
Actions=Arreter;Setup;SansModem;

[Desktop Action Arreter]
Name=Arreter le telephone (osmo-pmos-qemu stop)
Exec=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py --close-on-exit --title "Arret du telephone" --
 ↪  /usr/local/bin/osmo-pmos-qemu stop

[Desktop Action Setup]
Name=Rebrancher le modem et la voix (osmo-pmos-setup)
Exec=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py --tty --title "Modem et voix du telephone" --
 ↪  /usr/local/bin/osmo-pmos-setup

[Desktop Action SansModem]
Name=Demarrer sans modem (VM nue)
Exec=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py --tty --title "Telephone du banc (VM)" -- env OSMO_PMOS_RES=$r
 ↪  OSMO_PMOS_MODEM=0 /usr/local/bin/osmo-pmos-qemu
PMD
        chmod 644 "/usr/share/applications/osmo-pmos-$n.desktop" 2>/dev/null || true
    done
    rm -f /usr/local/bin/osmo-waydroid /usr/share/applications/osmo-waydroid.desktop 2>/dev/null || true
    update-desktop-database /usr/share/applications 2>/dev/null || true
    echo "  [telephone] osmo-pmos pose (VM postmarketOS ; « osmo-pmos install » la premiere fois)"
    echo "  [telephone] deux formats au lancement : smartphone (720x1440) et tablette (1280x800)"
    echo "  [telephone] « osmo-pmos-qemu stop » (ou le clic droit de l icone) arrete la VM et debranche le modem"
    return 0
}
osmo_poser_pmos

# ── DEKA : UN STOP AU CLIC DROIT DE L ICONE ──────────────────────────────
# [2026-10-25] L icone deka ne savait que DEMARRER (clic = osmo-deka-anim ->
# deka-start.sh). Pour arreter (tuer les workers, demonter les tables,
# desactiver le LVM) il fallait un terminal et « deka-start.sh --stop ». On
# ajoute donc une ACTION de menu (clic droit) « Arreter deka », exactement
# comme le telephone pmOS expose son arret. Le clic principal reste le
# demarrage ; deka-start.sh fait deja un --stop implicite avant de remonter.
#
# addition.sh (--opencl) POSE le .desktop ; ici on ne fait que le REECRIRE pour
# lui ajouter l action, et seulement si deka est installe - sinon on ne touche
# a rien.
osmo_poser_deka() {
    local dsk=/usr/share/applications/deka.desktop
```

```
    [ -x /usr/local/bin/osmo-deka-anim ] || [ -f "$dsk" ] || return 0
    local icon=/usr/share/osmo-operator/icons/deka.svg
    [ -f "$icon" ] || icon=deka
    local gtk=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py
    cat > "$dsk" <<DEKADSK
[Desktop Entry]
Type=Application
Name=deka
Name[fr]=deka
Comment=Monte les tables et lance les workers deka
Comment[fr]=Monte les tables et lance les workers deka
Exec=/usr/local/bin/osmo-deka-anim
Icon=$icon
Terminal=false
Categories=System;Utility;
Keywords=deka;tables;workers;opencl;
Actions=Arreter;

[Desktop Action Arreter]
Name=Arreter deka
Name[fr]=Arreter deka
Exec=$gtk --root --title "Arret deka" -- /root/deka/deka-start.sh --stop
DEKADSK
    chmod 644 "$dsk" 2>/dev/null || true
    update-desktop-database /usr/share/applications 2>/dev/null || true
    echo "  [deka] icone : clic = demarrer, clic droit « Arreter deka » = stop (workers + demontages)"
}
osmo_poser_deka

# ── LA 4G ET L UI SMARTPHONE : le rattrapage ──────────────────────────
# [2026-09-08] Meme source que addition.sh et l ISO (tools/osmo-lte-install.sh,
# tools/osmo-pmos-install.sh). Ici on ne compile rien : les configs du depot
# (sans ecraser celles qui existent), les lanceurs osmo-lte / osmo-epc /
# open5gs-epc.sh, le layout /opt/user_interface/pmos et son noyau PPP (du
# cache .deb ou de GitHub). Les binaires, s ils manquent, viennent des .deb
# du cache ; sinon « osmo-lte-install --build ».
osmo_poser_lte() {
    local d=/opt/GSM/osmo-operator
    if [ -f "$d/tools/osmo-lte-install.sh" ]; then
        # shellcheck source=tools/osmo-lte-install.sh
        DIR="$d" . "$d/tools/osmo-lte-install.sh"
        osmo_lte_install --debs --configs --launchers 2>&1 | sed 's/^/  [4G]/'
    fi
    if [ -f "$d/tools/osmo-pmos-install.sh" ]; then
        # shellcheck source=tools/osmo-pmos-install.sh
        DIR="$d" . "$d/tools/osmo-pmos-install.sh"
        osmo_pmos_install 2>&1 | sed 's/^/  [telephone]/'
    fi
}
osmo_poser_lte

# ── WIRESHARK : UNE ICONE QUI ECOUTE DEJA LE BANC ─────────────────────
# [2026-09-08] tools/osmo-wireshark-root.sh : root par pkexec (invite de mot de
# passe graphique), capture immediate LTE + SCTP + GSM, filtre d affichage 2G +
# 4G + telephone (signalisation, appel, SMS, data ; les battements de coeur
# retranches) et boutons de filtre dans le profil de root. « wireshark » au
# clavier est un lien dessus. Le favori du dock (org.wireshark.Wireshark, le
# Wireshark nu qui refuse de capturer) est remplace par le notre, pour chaque
# session ouverte - les favoris sont dans le dconf de l utilisateur, pas de
# root (voir la famille root / session). L ISO fait pareil dans 80-chroot.
osmo_poser_wireshark() {
    [ "$(id -u)" -eq 0 ] || return 0
    local d=/opt/GSM/osmo-operator
    [ -f "$d/tools/osmo-wireshark-root.sh" ] || return 0
    install -m 755 "$d/tools/osmo-wireshark-root.sh" /usr/local/bin/osmo-wireshark-root
    ln -sfn /usr/local/bin/osmo-wireshark-root /usr/local/bin/wireshark
```

```
    cat > /usr/share/applications/osmo-wireshark-root.desktop <<'WSD'
[Desktop Entry]
Type=Application
Name=Wireshark (banc)
GenericName=Analyseur reseau
Comment=Wireshark en root, capture immediate du banc, filtre 2G + 4G + telephone (signalisation, appel, SMS, data)
 ↪  sans les battements de coeur ; boutons Banc / 2G / 4G / Appel / SMS / Data / Attach
Exec=/usr/local/bin/osmo-wireshark-root %f
Icon=org.wireshark.Wireshark
Terminal=false
StartupNotify=true
Categories=Network;Monitor;
Keywords=wireshark;capture;pcap;reseau;lte;sctp;gsm;gsmtap;2g;4g;sms;appel;
WSD
    chmod 644 /usr/share/applications/osmo-wireshark-root.desktop
    update-desktop-database /usr/share/applications 2>/dev/null || true
    # Le favori, dans chaque session GNOME ouverte.
    local u uid bus
    for u in $(loginctl list-users --no-legend 2>/dev/null | awk '{print $2}'); do
        uid="$(id -u "$u" 2>/dev/null)" || continue
        bus="/run/user/$uid/bus"; [ -S "$bus" ] || continue
        sudo -u "$u" DBUS_SESSION_BUS_ADDRESS="unix:path=$bus" python3 - <<'PY' 2>/dev/null || true
import subprocess, ast
g = ["gsettings", "get", "org.gnome.shell", "favorite-apps"]
cur = ast.literal_eval(subprocess.run(g, capture_output=True, text=True).stdout.strip() or "[]")
neu = [a for a in cur if a not in ("org.wireshark.Wireshark.desktop", "wireshark.desktop",
 ↪  "wireshark-root.desktop")]
if "osmo-wireshark-root.desktop" not in neu:
    i = next((k for k, a in enumerate(cur) if a in ("org.wireshark.Wireshark.desktop", "wireshark.desktop")),
 ↪  len(neu))
    neu.insert(min(i, len(neu)), "osmo-wireshark-root.desktop")
if neu != cur:
    subprocess.run(["gsettings", "set", "org.gnome.shell", "favorite-apps", repr(neu)], check=False)
PY
    done
    echo "  [wireshark] osmo-wireshark-root pose (LTE + SCTP + GSM, root par pkexec) ; « wireshark » y mene ;
     ↪  favori du dock remplace"
    return 0
}
osmo_poser_wireshark

# ── LES LANCEURS /usr/local/bin (osmo-dino-play, osmo-youtube...) ──────────
# [2026-09-07] osmo-dino-play est CUIT dans /usr/local/bin par
# tools/osmo-extras-install.sh (addition.sh et l ISO) ; quand il change dans
# le depot - le plein ecran pilote par la page, ce jour-la - une machine deja
# installee garde l ancien. On rejoue la seule fonction qui pose les lanceurs :
# idempotente, sans apt, sans reseau.
osmo_reposer_lanceurs() {
    [ "$(id -u)" -eq 0 ] || return 0
    local ex=/opt/GSM/osmo-operator/tools/osmo-extras-install.sh
    [ -f "$ex" ] || return 0
    # shellcheck source=tools/osmo-extras-install.sh
    ( . "$ex" && _osmo_extras_lanceurs ) >/dev/null 2>&1 \
        && echo "  [bureau] lanceurs /usr/local/bin reposes (osmo-dino-play, osmo-pmos...)" \
        || true
    return 0
}
osmo_reposer_lanceurs

# ── start-direct.sh DEVENU start-native.sh (2026-10-18) ────────────────────
# Le lien /usr/local/bin/osmo-start-direct d une machine deja installee vise
# encore start-direct.sh, que git pull vient de faire disparaitre. On le
# repointe, et on pose le nouveau nom a cote. Idempotent.
osmo_reposer_lien_start_native() {
    [ "$(id -u)" -eq 0 ] || return 0
    local t=/opt/GSM/osmo-operator/start-native.sh l
```

```
    [ -f "$t" ] || return 0
    for l in /usr/local/bin/osmo-start-native /usr/local/bin/osmo-start-direct; do
        [ "$(readlink "$l" 2>/dev/null)" = "$t" ] || ln -sf "$t" "$l"
    done
    return 0
}
osmo_reposer_lien_start_native

# ── osmo-banc.service : L UNITE INSTALLEE SUIT CELLE DU DEPOT (2026-10-23) ──
# La copie de /etc/systemd/system visait encore start-direct.sh : sa
# ConditionPathExists etait fausse, systemd sautait l unite SANS erreur et
# l icone du telephone rouge (launch.sh -> systemctl restart osmo-banc) ne
# lancait rien. On repose services/osmo-banc.service (start.sh) quand elle
# differe, sans toucher a une unite absente (machine sans le service).
osmo_reposer_unite_banc() {
    [ "$(id -u)" -eq 0 ] || return 0
    local src=/opt/GSM/osmo-operator/services/osmo-banc.service dst=/etc/systemd/system/osmo-banc.service
    [ -f "$src" ] && [ -f "$dst" ] || return 0
    cmp -s "$src" "$dst" && return 0
    install -m644 "$src" "$dst" && systemctl daemon-reload 2>/dev/null || true
    return 0
}
osmo_reposer_unite_banc

# ── AUCUN SERVICE TELECOM AU BOOT : RATTRAPAGE, UNE SEULE FOIS (2026-10-23) ──
# Les machines installees avant la politique (tools/osmo-telecom-boot.sh) la
# recoivent ici, UNE fois (marqueur) : un service que l operateur reactive
# ensuite (systemctl enable osmo-banc) ne doit pas etre redesactive a chaque
# mise a jour.
osmo_politique_telecom_boot() {
    [ "$(id -u)" -eq 0 ] || return 0
    local m=/var/lib/osmo-operator/telecom-boot-v1 t=/opt/GSM/osmo-operator/tools/osmo-telecom-boot.sh
    [ -e "$m" ] || [ ! -f "$t" ] && return 0
    bash "$t" && mkdir -p "${m%/*}" && date -Is > "$m"
    return 0
}
osmo_politique_telecom_boot

# ── LE « ? » DU DOCK : PAGE D AIDE SERVIE EN HTTP, EN TETE (2026-10-23) ──────
# L icone ouvrait file:///usr/share/osmo-operator/info.html, que Firefox (snap)
# ne lit pas : page vide. On pose /usr/local/bin/osmo-info (dashboard 8080, repli
# local), la page a jour aux deux endroits, et osmo-info.desktop en tete des
# favoris de chaque session ouverte.
osmo_poser_aide() {
    [ "$(id -u)" -eq 0 ] || return 0
    local R=/opt/GSM/osmo-operator u uid bus
    [ -f "$R/configs/info.html" ] || return 0
    install -m755 "$R/tools/osmo-info" /usr/local/bin/osmo-info 2>/dev/null || true
    install -D -m644 "$R/configs/info.html" /usr/share/osmo-operator/info.html
    [ -d /opt/GSM/osmo-egprs-web/web ] && cp -f "$R/configs/info.html" /opt/GSM/osmo-egprs-web/web/info.html
    install -m644 "$R/data/desktop/osmo-info.desktop" /usr/share/applications/osmo-info.desktop
    update-desktop-database /usr/share/applications 2>/dev/null || true
    for u in $(loginctl list-users --no-legend 2>/dev/null | awk '{print $2}'); do
        uid="$(id -u "$u" 2>/dev/null)" || continue
        bus="/run/user/$uid/bus"; [ -S "$bus" ] || continue
        sudo -u "$u" DBUS_SESSION_BUS_ADDRESS="unix:path=$bus" python3 - <<'PY' 2>/dev/null || true
import subprocess, ast
g = ["gsettings", "get", "org.gnome.shell", "favorite-apps"]
cur = ast.literal_eval(subprocess.run(g, capture_output=True, text=True).stdout.strip() or "[]")
neu = ["osmo-info.desktop"] + [a for a in cur if a != "osmo-info.desktop"]
if neu != cur:
    subprocess.run(["gsettings", "set", "org.gnome.shell", "favorite-apps", repr(neu)], check=False)
PY
    done
    return 0
}
```

```
osmo_poser_aide

# ── BUREAU INTERACTIF : RATTRAPAGE DU WRAPPER osmo-desktop-panel ──────────────
# Le lanceur des widgets (/usr/local/bin/osmo-desktop-panel) est CUIT dans
# l image par iso_modules/85-installeur-bureau.sh. Les widgets eux-memes
# (osmo-topzone.py, osmo-launcher.py, osmo-ts-probe.py) vivent dans le depot et
# arrivent par git pull - mais une machine deja installee garde son ANCIEN
# wrapper, qui ne les lance pas. On re-pose donc le wrapper quand il ignore
# encore la zone haute vivante - ou, depuis le 2026-09-10, quand il compte pour
# vivant un widget rescape d une session X precedente (osmo_vivant : le Conky
# ne revenait jamais apres un redemarrage de GNOME). Miroir exact de 85.
osmo_reposer_bureau_interactif() {
    [ "$(id -u)" -eq 0 ] || return 0
    local w=/usr/local/bin/osmo-desktop-panel
    [ -f "$w" ] || return 0
    grep -q 'osmo_vivant' "$w" 2>/dev/null && return 0
    cat > "$w" <<'PANEL'
#!/bin/sh
# osmo-desktop-panel - tient l encart et Conky en vie pour toute la session.
REPO=/opt/GSM/osmo-operator
export OSMO_REPO="$REPO"
# La carte LAB GSM n est plus cuite dans le fond : c est osmo-topzone.py qui la
# rend, vivante. On le dit au rendu du fond.
export OSMO_LIVE_BANNER=1

# ── UN WIDGET RESCAPE DE LA SESSION D AVANT N EST PAS UN WIDGET ──────────────
# [2026-09-10] Le Conky avait disparu du coin haut droit, et le gardien ne le
# relancait JAMAIS. Pourquoi : un conky de la session X precedente tournait
# encore, avec DISPLAY=:0, alors que GNOME etait reparti sur :1 (le socket
# /tmp/.X11-unix/X0 avait disparu avec l ancien serveur). Il ne peut plus rien
# peindre - « Invalid MIT-MAGIC-COOKIE-1 key / can't open display: :0 » dans
# /tmp/osmo-conky.log - mais il EXISTE, et « pgrep -x conky » ne sait dire que
# ca. Le gardien le comptait pour vivant et n en relancait pas d autre : plus
# de tableau de bord pour toute la session. Meme piege pour les widgets python
# (osmo-ts-probe.py etait dans le meme etat).
#
# On ne compte donc que les processus qui regardent NOTRE display, et on tue
# les autres : ils ne peuvent plus servir personne.
osmo_vivant() {
    _trouve=1
    for _p in $(pgrep "$@" 2>/dev/null); do
        _d="$(tr '\0' '\n' < "/proc/$_p/environ" 2>/dev/null | sed -n 's/^DISPLAY=//p' | head -1)"
        # On ne tue QUE ce qu on a lu et qui regarde ailleurs. Un environ
        # illisible, ou sans DISPLAY, ne prouve rien : ce processus compte
        # pour vivant - une sonde qui confond « absent » et « invisible »
        # ferait ici pousser un doublon de chaque widget a chaque tour.
        if [ -n "$_d" ] && [ "$_d" != "$DISPLAY" ]; then kill "$_p" 2>/dev/null
        else _trouve=0
        fi
    done
    return $_trouve
}
# Et le gardien ne survit pas a son propre serveur X : sans cette sortie,
# l ancien tourne en parallele du neuf et relance dans le vide, pour toujours
# (on en avait deux, l un sur :0 et l autre sur :1).
osmo_display_vivant() {
    [ -n "$DISPLAY" ] || return 0          # pas de display nomme : on ne juge pas
    _n="${DISPLAY#*:}"; _n="${_n%%.*}"
    [ -S "/tmp/.X11-unix/X$_n" ]
}

# ── LES FLECHES DU CONKY, AU CLAVIER : Ctrl+AltGr+Gauche / Droite ────────────
# [2026-09-10] Le raccourci est cuit dans un override de schema
# (/usr/share/glib-2.0/schemas/98-osmo-keys.gschema.override, pose par
# iso_modules/80-chroot.sh) - sauf qu un override ne prend PAS sur un schema
# RELOCALISABLE : « gsettings get ...custom-keybinding:/...osmo-op-next/
```

```
# binding » rendait '' sur un systeme ou le fichier etait pourtant bien
# compile. Le raccourci n a donc jamais rien lie. On l ecrit ici, dans la
# session, une fois par demarrage : c est le seul endroit qui tourne SOUS le
# compte de l utilisateur, et gsettings ecrit dans son dconf.
# <Mod5> est ISO_Level3_Shift, c est-a-dire AltGr (xmodmap -pm le confirme).
osmo_raccourci() {
    command -v gsettings >/dev/null 2>&1 || return 0
    _k=org.gnome.settings-daemon.plugins.media-keys.custom-keybinding
    _p=/org/gnome/settings-daemon/plugins/media-keys/custom-keybindings
    gsettings set org.gnome.settings-daemon.plugins.media-keys custom-keybindings \
        "['$_p/osmo-op-next/','$_p/osmo-op-prev/']" 2>/dev/null || return 0
    gsettings set "$_k:$_p/osmo-op-next/" name 'osmo-operator : arret suivant (Conky)' 2>/dev/null
    gsettings set "$_k:$_p/osmo-op-next/" command '/usr/local/bin/osmo-op --next' 2>/dev/null
    gsettings set "$_k:$_p/osmo-op-next/" binding '<Control><Mod5>Right' 2>/dev/null
    gsettings set "$_k:$_p/osmo-op-prev/" name 'osmo-operator : arret precedent (Conky)' 2>/dev/null
    gsettings set "$_k:$_p/osmo-op-prev/" command '/usr/local/bin/osmo-op --prev' 2>/dev/null
    gsettings set "$_k:$_p/osmo-op-prev/" binding '<Control><Mod5>Left' 2>/dev/null
}
osmo_raccourci

# Conky doit trouver le bureau GNOME deja peint (own_window_type desktop),
# sinon il se pose derriere le fond d ecran.
sleep 6
while osmo_display_vivant; do
    osmo_vivant -f "$REPO/tools/osmo-panel.py" || \
        "$REPO/tools/osmo-panel.py" >>/tmp/osmo-panel.log 2>&1 &
    osmo_vivant -f "$REPO/tools/osmo-dino.py" || \
        "$REPO/tools/osmo-dino.py" >>/tmp/osmo-dino.log 2>&1 &
    # la sonde d activite des timeslots (conf osmo-bsc + VTY) pour la banniere
    osmo_vivant -f "$REPO/tools/osmo-ts-probe.py" || \
        "$REPO/tools/osmo-ts-probe.py" >>/tmp/osmo-ts-probe.log 2>&1 &
    # la zone haute LAB GSM vivante (timeslots clignotants + accueil d appli)
    osmo_vivant -f "$REPO/tools/osmo-topzone.py" || \
        "$REPO/tools/osmo-topzone.py" >>/tmp/osmo-topzone.log 2>&1 &
    # la barre de lancement categorisee du bas
    osmo_vivant -f "$REPO/tools/osmo-launcher.py" || \
        "$REPO/tools/osmo-launcher.py" >>/tmp/osmo-launcher.log 2>&1 &
    osmo_vivant -x conky || \
        conky --daemonize -c "$REPO/configs/conky/osmo-conky.conf" >>/tmp/osmo-conky.log 2>&1
    sleep 5
done
PANEL
    chmod 755 "$w"
    # relancer : tuer l ancien wrapper, l autostart (ou la boucle) reprend le neuf
    pkill -f /usr/local/bin/osmo-desktop-panel 2>/dev/null || true
    echo "  [bureau] wrapper osmo-desktop-panel remis a jour (zone LAB GSM vivante, barre de lancement, widgets
     ↪  suivis par display)"
    return 0
}
osmo_reposer_bureau_interactif

# ── LES DEUX CONSOLES WEB, ACTIVES PAR DEFAUT (COTE NATIF) ──────────────────
# [2026-09-10] L ISO les active au build (iso_modules/60-dashboard.sh pour le
# tableau de bord 2G, iso_modules/82-services.sh pour les deux) ; une machine
# deja installee, elle, gardait ce qu elle avait - et pour open5gs-webui c est
# RIEN : l unite n existait pas avant aujourd hui. On pose ce qui manque et on
# active ce qui ne l est pas.
#
# LE MOYEN DE DIRE NON, C EST `systemctl mask`. Un simple `disable` serait
# remis a l endroit au prochain update - c est le sens de « par defaut ». Une
# unite masquee est laissee tranquille, ici comme partout.
#
# Aucune des deux n est DEMARREE ici : osmo-egprs-web suit le banc (PartOf=
# osmo-banc.service) et open5gs-webui n a rien a servir tant que le WebUI n est
# pas construit (« osmo-lte-install --webui »). L activation suffit : elles
# partiront au prochain demarrage, ou tout de suite avec `systemctl start`.
```

```
osmo_activer_consoles_web() {
    [ "$(id -u)" -eq 0 ] || return 0
    command -v systemctl >/dev/null 2>&1 || return 0
    local u src dst etat
    for u in osmo-egprs-web open5gs-webui; do
        src="$REPO/services/$u.service"
        dst="/etc/systemd/system/$u.service"
        [ -f "$src" ] || continue
        etat="$(systemctl is-enabled "$u" 2>/dev/null)"
        [ "$etat" = masked ] && continue          # coupee a la main : on respecte
        if [ ! -f "$dst" ] || ! cmp -s "$src" "$dst"; then
            install -m644 "$src" "$dst"
            systemctl daemon-reload 2>/dev/null || true
            echo "  [web] unite $u.service posee depuis le depot"
        fi
        if [ "$etat" != enabled ]; then
            systemctl enable "$u" >/dev/null 2>&1 \
                && echo "  [web] $u.service active au demarrage" \
                || echo "  [web] $u.service : activation impossible"
        fi
    done
    return 0
}
osmo_activer_consoles_web

# -- FIREFOX ------------------------------------------------------------------
# Le dashboard et fft-web s ouvrent dans un navigateur. Les images du
# 2026-09-04 et apres embarquent le .deb de Mozilla : il est deja la, apt le
# tient a jour, et cette fonction ne fait rien.
#
# Elle ne sert donc qu aux machines plus anciennes, ou aux images ou le depot
# Mozilla etait injoignable au build. Elle repose ce depot puis installe le
# paquet - le "firefox" des depots Ubuntu n est PAS une solution de repli : il
# rappelle snapd, et c est ce montage-la qui laissait des bancs sans navigateur.
osmo_installer_firefox() {
    [ "$(id -u)" -eq 0 ] || return 0
    command -v firefox >/dev/null 2>&1 && return 0
    command -v apt-get >/dev/null 2>&1 || return 0
    echo "[*] Installation de Firefox (deb Mozilla)..."
    if [ ! -s /etc/apt/keyrings/packages.mozilla.org.asc ]; then
        install -d -m0755 /etc/apt/keyrings
        curl -fsSL --retry 3 https://packages.mozilla.org/apt/repo-signing-key.gpg \
             -o /etc/apt/keyrings/packages.mozilla.org.asc || {
            echo "[WARN] packages.mozilla.org injoignable - Firefox non installe."
            return 0
        }
    fi
    echo "deb [signed-by=/etc/apt/keyrings/packages.mozilla.org.asc] https://packages.mozilla.org/apt mozilla
    ↪  main" \
        > /etc/apt/sources.list.d/mozilla.list
    printf "Package: *\nPin: origin packages.mozilla.org\nPin-Priority: 1000\n" \
        > /etc/apt/preferences.d/mozilla
    apt-get update -qq 2>/dev/null || true
    if DEBIAN_FRONTEND=noninteractive apt-get install -y firefox >/dev/null 2>&1; then
        echo "[OK] Firefox installe (deb Mozilla)."
    else
        echo "[WARN] Firefox non installe (reseau ?)."
    fi
    return 0
}
osmo_installer_firefox

# -- LE COMPTE DE LA SESSION DOIT VOIR DOCKER ---------------------------------
# [2026-09-05] RATTRAPAGE : le Conky (tools/conky-osmo-status.sh) et l'encart
# (tools/osmo-panel.py) tournent sous le compte de la SESSION et sondent chaque
# operateur en conteneur par "docker exec". Sans le groupe docker, la sonde rend
```

```
# "permission denied while trying to connect to the docker API", que le code lit
# comme "operateur arrete" : Conky annoncant « banc a l arret » et sections
# Coeur GSM / Radio / Abonnes VIDES, avec les trois operateurs bien vivants.
# Le groupe est desormais pose a l'installation (users.conf, defaultGroups) et
# par addition.sh (_docker_groupe_session) ; ici on rattrape les machines deja
# installees, qui ne repassent par aucun des deux.
# Meme detection qu'addition.sh : SUDO_USER, puis PKEXEC_UID, puis le
# proprietaire d'un bus de session actif - update.sh est lance par une icone
# (pkexec) autant que par sudo.
osmo_docker_groupe_session() {
    [ "$(id -u)" -eq 0 ] || return 0
    getent group docker >/dev/null 2>&1 || return 0
    local u uid bus vus=""
    for u in "${SUDO_USER:-}" \
             "$([ -n "${PKEXEC_UID:-}" ] && getent passwd "$PKEXEC_UID" | cut -d: -f1)"; do
        [ -n "$u" ] && [ "$u" != root ] && getent passwd "$u" >/dev/null 2>&1 && { vus="$u"; break; }
    done
    if [ -z "$vus" ]; then
        for bus in /run/user/*/bus; do
            [ -S "$bus" ] || continue
            uid="${bus#/run/user/}"; uid="${uid%/bus}"
            [ "$uid" = 0 ] && continue
            u="$(getent passwd "$uid" 2>/dev/null | cut -d: -f1)"
            [ -n "$u" ] && vus="$vus $u"
        done
    fi
    for u in $vus; do
        id -nG "$u" 2>/dev/null | tr ' ' '\n' | grep -qx docker && continue
        usermod -aG docker "$u" 2>/dev/null || continue
        echo "[OK] $u ajoute au groupe docker (le Conky et l encart voient les conteneurs)."
        echo "     Effectif au prochain login de $u."
    done
}
osmo_docker_groupe_session

# ── UNITES osmo-* : User=osmocom -> compte de la SESSION (rattrapage) ──────
# Miroir d'addition.sh (_osmo_unites_user_session) pour les machines DEJA
# installees qui ne repassent pas par addition.sh. Meme cause : les .service
# amont (osmo-bsc/bts-trx/bts-virtual/msc) posent User=osmocom / Group=osmocom,
# et le compte osmocom n'existe pas en natif (l'image le supprime, cf
# iso_modules/52-qemu.sh) -> 217/USER, crash-loop Restart=always MUET, run.sh
# abandonne ("OsmoMSC started but never ready"). On rend les unites au compte de
# session via un drop-in /etc/systemd/system/<unit>.d/ (le .service amont reste
# intact). Meme detection d'user qu'osmo_docker_groupe_session.
osmo_unites_user_session() {
    [ "$(id -u)" -eq 0 ] || return 0
    command -v systemctl >/dev/null 2>&1 || return 0
    local u uid bus grp unit base d n=0 vus=""
    for u in "${SUDO_USER:-}" \
             "$([ -n "${PKEXEC_UID:-}" ] && getent passwd "$PKEXEC_UID" | cut -d: -f1)"; do
        [ -n "$u" ] && [ "$u" != root ] && getent passwd "$u" >/dev/null 2>&1 && { vus="$u"; break; }
    done
    if [ -z "$vus" ]; then
        for bus in /run/user/*/bus; do
            [ -S "$bus" ] || continue
            uid="${bus#/run/user/}"; uid="${uid%/bus}"
            [ "$uid" = 0 ] && continue
            u="$(getent passwd "$uid" 2>/dev/null | cut -d: -f1)"
            [ -n "$u" ] && { vus="$u"; break; }
        done
    fi
    [ -n "$vus" ] && getent passwd "$vus" >/dev/null 2>&1 || vus=root
    grp="$(id -gn "$vus" 2>/dev/null || echo "$vus")"
    for unit in /lib/systemd/system/osmo-*.service /etc/systemd/system/osmo-*.service; do
        [ -f "$unit" ] || continue
        grep -qE '^(User|Group)=osmocom$' "$unit" || continue
```

```
        base="$(basename "$unit")"; d="/etc/systemd/system/$base.d"
        mkdir -p "$d"
        printf '[Service]\nUser=%s\nGroup=%s\n' "$vus" "$grp" > "$d/00-session-user.conf"
        n=$((n+1))
    done
    [ "$n" -gt 0 ] || return 0
    systemctl daemon-reload 2>/dev/null || true
    # repertoires d'etat/log osmocom -> compte de session, sinon "Unable to
    # create file /var/log/osmocom/*.log" -> parse .cfg en echec -> status=1.
    for _d in /var/log/osmocom /var/lib/osmocom /run/osmocom; do
        [ -e "$_d" ] && chown -R "$vus:$grp" "$_d" 2>/dev/null || true
    done
    for base in osmo-msc osmo-bsc osmo-bts-trx osmo-bts-virtual; do
        systemctl cat "$base" >/dev/null 2>&1 || continue
        systemctl reset-failed "$base" 2>/dev/null || true
        systemctl try-restart "$base" 2>/dev/null || true
    done
    echo "[OK] unites osmo-* rendues au compte de session ($vus) - sinon User=osmocom -> 217/USER."
    return 0
}
osmo_unites_user_session

# ── LE 600 QU ON ENTEND MAL : LE SUPERVISEUR GAPK ──────────────────────
# [2026-09-09] scripts/gapk-start.sh (mode auto) lancait, a chaque appel, un
# osmo-gapk qui encodait le micro (gsm_in) et l ENVOYAIT au pair de la premiere
# connexion MGW - le port RTP d osmo-bts (127.0.0.1:16384). Le BTS recevait deux
# flux RTP entremeles sur un seul jitter-buffer : son hache, double, le micro
# renvoye en direct dans l oreille. Le mobile (io-handler gapk) porte deja le
# montant par la radio ; cette patte etait un doublon. Le correctif est dans le
# depot (git pull), mais lib/audio.sh prefere /etc/osmocom/gapk-start.sh - une
# COPIE posee par le Dockerfile/l ISO, que git pull ne touche pas. On la repose
# ici, on tue un eventuel emetteur qui trainerait, et on relance le superviseur
# (session tmux « gapk ») pour qu il lise la nouvelle version : sans cela le
# doublon survivait a la mise a jour jusqu au prochain redemarrage.
osmo_poser_correctif_audio() {
    [ "$(id -u)" -eq 0 ] || return 0
    # NB: deux 'local' separes - sous 'set -u', bash evalue TOUS les membres
    # droits d un meme 'local' AVANT d affecter le premier, donc "$d/..." sur la
    # meme ligne que d=... plantait ("d: unbound variable").
    local d=/opt/GSM/osmo-operator
    local src="$d/scripts/gapk-start.sh" dst=/etc/osmocom/gapk-start.sh
    [ -f "$src" ] || return 0
    if [ -d /etc/osmocom ] && ! cmp -s "$src" "$dst"; then
        install -m 755 "$src" "$dst" && echo "[OK] $dst repose depuis scripts/ (emetteur RTP en doublon retire)"
    fi
    # Un osmo-gapk en cours = l ancien TX (le mobile n en lance jamais) : on l arrete.
    pkill -x osmo-gapk 2>/dev/null && echo "[OK] osmo-gapk en doublon arrete"
    rm -f /var/run/gapk-tx.pid /var/run/gapk-rx.pid
    # Le superviseur ne se relance que s il tourne deja : hors banc, rien a faire.
    if tmux has-session -t gapk 2>/dev/null; then
        local log_dir; log_dir="$(tmux list-panes -t gapk -F '#{pane_start_command}' 2>/dev/null \
                                  | grep -oE 'tee [^ ]+/gapk-auto.log' | awk '{print $2}' | xargs -r dirname)"
        [ -n "$log_dir" ] || log_dir="${LOG_DIR:-/run/user/0/osmo-nitb/logs}"
        local sock="${PULSE_SOCK:-/var/run/pulse/native}"
        tmux kill-session -t gapk 2>/dev/null || true
        rm -f /var/run/gapk-auto.lock
        [ -x "$dst" ] || dst="$src"
        tmux new-session -d -s gapk \
            "GAPK_ALSA_DEV=gsm_out GAPK_ALSA_DEV_IN=gsm_in PULSE_SERVER=unix:${sock} bash '$dst' auto gsm gsm_out
             ↪  gsm_in 2>&1 | tee ${log_dir}/gapk-auto.log" \
            && echo "[OK] superviseur gapk relance (tmux gapk, journal ${log_dir}/gapk-auto.log)"
    fi
    # ── Le 600 depuis le telephone : pas d annonce avant l echo ──
    # [2026-09-09] Deuxieme cause du son mauvais « pendant une partie de
    # l appel » : avec l enregistrement (sub-record) sur la jambe Local du banc,
    # le passage Playback(demo-echotest) -> Echo faisait lacher a Asterisk une
```

```
    # rafale de milliers de paquets RTP de silence vers le BTS. Le gabarit
    # configs/extensions.conf saute l annonce quand ECHO_NO_PROMPT=1 (pose par
    # osmo-phonesim-banc.py) ; la copie vivante /etc/asterisk/extensions.conf
    # n est regeneree qu au demarrage : on la retouche ici et on recharge.
    local ext=/etc/asterisk/extensions.conf
    if [ -f "$ext" ] && ! grep -q 'ECHO_NO_PROMPT' "$ext"; then
        sed -i '/^exten => 600,1,NoOp(=== ECHO TEST/,/^ same => n,Hangup()/{
            s|^ same => n,Playback(demo-echotest)$| same => n,GotoIf($["${ECHO_NO_PROMPT}" = "1"]?echo)\n same =>
↪   n,Playback(demo-echotest)|
            s|^ same => n,Echo()$| same => n(echo),Echo()|
        }' "$ext" && echo "[OK] $ext : le 600 saute son annonce pour le banc (ECHO_NO_PROMPT)"
    fi
    # L annonce de l echo test, rejouee sur la patte du mobile (Dial ...rT${DIAL_OPTS},
    # le banc pose DIAL_OPTS=A(demo-echotest) pour le 600) : contexte [internal] seulement.
    if [ -f "$ext" ] && ! grep -q 'rT\${DIAL_OPTS}' "$ext"; then
        sed -i '/^\[internal\]/,/^\[/{s|^ same => n,Dial(PJSIP/${EXTEN}@gsm_msc,,rT)$| same =>
        ↪  n,Dial(PJSIP/${EXTEN}@gsm_msc,,rT${DIAL_OPTS})|}' "$ext" \
            && echo "[OK] $ext : annonce de l echo test sur la patte du mobile (DIAL_OPTS)"
    fi
    command -v asterisk >/dev/null 2>&1 && asterisk -rx "dialplan reload" >/dev/null 2>&1 || true
    # Le modem du banc (osmo-phonesim-banc.py --connect) tourne depuis le depot
    # mais ne relit pas son code : on le relance tel qu osmo-pmos-setup.sh l a
    # lance, sinon il continue d originer sans ECHO_NO_PROMPT et de marteler
    # « call 1 answer » (29 x « % No alerting call » par appel).
    local bpid
    bpid="$(pgrep -f '[o]smo-phonesim-banc.py --connect' | head -1)"
    if [ -n "$bpid" ]; then
       local bargs; bargs="$(tr '\0' ' ' <"/proc/$bpid/cmdline" 2>/dev/null | sed 's/.*osmo-phonesim-banc.py //')"
        kill "$bpid" 2>/dev/null; sleep 1
        setsid nohup env OSMO_AT_LOG_OFONO=/tmp/osmo-at-vm.log \
            "$d/tools/osmo-phonesim-banc.py" $bargs >>/tmp/osmo-phonesim-vm-root.log 2>&1 </dev/null &
        sleep 3
        pgrep -f '[o]smo-phonesim-banc.py --connect' >/dev/null 2>&1 \
            && echo "[OK] modem du banc relance ($bargs) - journal /tmp/osmo-phonesim-vm-root.log" \
            || echo "[!!] le modem du banc ne s est pas relance : osmo-pmos-setup pour le rebrancher"
    fi
    # ── Micro « sature, grave » : l AGC analogique de l annuleur d echo ──
    # [2026-09-09] module-echo-cancel (webrtc) pilotait le gain du micro et
    # le faisait pomper entre 0 et 100 % (voir tools/osmo-pmos-setup.sh). Si
    # l annuleur tourne encore sans aec_args, on le recharge a l identique
    # avec l AGC analogique coupe, et on rebranche les flux de la VM dessus.
    export PULSE_SERVER="${PULSE_SERVER:-unix:/var/run/pulse/native}"
    # [2026-09-23] Seulement un annuleur WEBRTC sans aec_args : speex (le defaut
    # desormais, lib/audio.sh) n en a pas et ne doit pas etre ramene a webrtc.
    if command -v pactl >/dev/null 2>&1 && pactl list short modules 2>/dev/null | grep module-echo-cancel | grep
    ↪  -q aec_method=webrtc \
       && ! pactl list short modules 2>/dev/null | grep module-echo-cancel | grep -q analog_gain_control=0; then
        local ecm mic hp so si
        ecm="$(pactl list short modules | awk '/module-echo-cancel/ {print $1; exit}')"
        mic="$(pactl list short modules | grep module-echo-cancel | grep -oE 'source_master=[^ ]+' | cut -d= -f2)"
        hp="$(pactl list short modules | grep module-echo-cancel | grep -oE 'sink_master=[^ ]+' | cut -d= -f2)"
        if [ -n "$mic" ] && [ -n "$hp" ]; then
            pactl unload-module "$ecm" 2>/dev/null
            pactl load-module module-echo-cancel aec_method=webrtc source_master="$mic" sink_master="$hp" \
                source_name=osmo_mic_ec sink_name=osmo_hp_ec \
                'aec_args="analog_gain_control=0 digital_gain_control=1 noise_suppression=1 high_pass_filter=1"' \
                source_properties=device.description=Micro_sans_echo
↪   sink_properties=device.description=HP_sans_echo >/dev/null 2>&1 \
                && echo "[OK] annuleur d echo recharge sans AGC analogique (micro fixe a 25 %)"
            pactl set-default-source osmo_mic_ec 2>/dev/null; pactl set-default-sink osmo_hp_ec 2>/dev/null
            for so in $(pactl list source-outputs 2>/dev/null | awk '/Source Output #/{id=$3} /media.name =
            ↪  "combine"/{print id}' | tr -d '#'); do
                pactl move-source-output "$so" osmo_mic_ec 2>/dev/null; done
            for si in $(pactl list sink-inputs 2>/dev/null | awk '/Sink Input #/{id=$3} /media.name =
            ↪  "combine"/{print id}' | tr -d '#'); do
                pactl move-sink-input "$si" osmo_hp_ec 2>/dev/null; done
```

```
            pactl set-source-mute "$mic" 0 2>/dev/null; pactl set-source-volume "$mic" 25% 2>/dev/null
            pactl set-source-mute osmo_mic_ec 0 2>/dev/null; pactl set-source-volume osmo_mic_ec 25% 2>/dev/null
        fi
    fi
    # ── Le message factice de speech-dispatcher dans les haut-parleurs ──
    # [2026-09-09] Voir lib/audio.sh (osmo_tts_off) : on cree le sink poubelle
    # s il manque et on y envoie tout flux speech-dispatcher deja ouvert.
    if command -v pactl >/dev/null 2>&1 && pactl info >/dev/null 2>&1; then
        pactl list short sinks 2>/dev/null | grep -qw osmo_tts_off \
            || pactl load-module module-null-sink sink_name=osmo_tts_off format=s16le rate=8000 channels=1
             ↪  sink_properties=device.description=TTS_off >/dev/null 2>&1
        for si in $(pactl list sink-inputs 2>/dev/null | awk '/Sink Input #/{id=$3} /application.name =
         ↪  "speech-dispatcher/{print id}' | tr -d '#'); do
            pactl move-sink-input "$si" osmo_tts_off 2>/dev/null && echo "[OK] flux speech-dispatcher detourne des
             ↪  haut-parleurs (osmo_tts_off)"
        done
    fi
    return 0
}
osmo_poser_correctif_audio

# ── LA CONSOLE srsUE DU BUREAU : DES JOURNAUX 4G ──────────────────────────
# [2026-09-09] L encart (tools/osmo-fft-snap.py) cherchait /tmp/ue.log, jamais
# ecrit : osmo-lte.sh impose --log.filename=/tmp/osmo-lte-ue.log et met la
# console dans /tmp/osmo-lte-ue.console. Le depot est corrige (git pull) ; ici on
# porte le NAS en info (sans hexa) dans le ue.conf DEJA pose (osmo-lte-install --configs ne
# l ecrase pas sans --force) et on relance l encart, qui lit ses chemins au
# demarrage. Le UE lui-meme n est pas relance : les niveaux valent au prochain
# « osmo-lte restart ».
osmo_poser_journaux_lte() {
    [ "$(id -u)" -eq 0 ] || return 0
    local f
    for f in /root/.config/srsran/ue.conf /home/*/.config/srsran/ue.conf; do
        [ -f "$f" ] || continue
        grep -q '^nas_hex_limit *= *0' "$f" && continue
        sed -i -e '/^rrc_level *= *info$/d' -e '/^nas_level *= *info$/d' \
               -e '/^\[log\]/,/^\[/{s/^\(all_level *= *warning\)$/\1\nnas_level = info\nnas_hex_limit = 0/}' \
               -e 's|^filename *= */tmp/ue\.log$|filename = /tmp/osmo-lte-ue.log|' "$f" \
            && echo "[OK] $f : NAS en info (console srsUE du bureau)"
    done
    if systemctl is-active --quiet osmo-fft-snap 2>/dev/null; then
       systemctl try-restart osmo-fft-snap 2>/dev/null && echo "[OK] encart osmo-fft-snap relance (console srsUE)"
    fi
    return 0
}
osmo_poser_journaux_lte

# ── LE FORMAT DE TRAME TCH ET LE TAMPON DU MOBILE, DANS LES FORKS ──────────
# [2026-09-09] DEUX REGLAGES VITAUX VIVENT DANS DES DEPOTS QU ON REFETCHE.
# La voix du banc depend de deux lignes qui ne sont PAS dans osmo-operator :
#
#  1. « io-tch-format rtp » dans cfgs/mobile_group1.cfg des forks. C est le
#     gabarit que run.sh (run_modules/20-mobile-cfg.sh) RECOPIE par-dessus
#     ~/.osmocom/bb/mobile_group1.cfg a chaque demarrage, ecrasant ce que
#     start-native.sh avait genere. Sans cette ligne, le mobile lit et ecrit
#     ses trames TCH dans la disposition Texas Instruments des vrais Calypso,
#     alors que pont.py les code avec libosmocoding, en disposition
#     RTP/RFC3551. Les deux bouts ne se comprennent DANS AUCUN SENS, aucun
#     compteur ne bronche, et la voix sort robotisee. Constate le 09/09 : les
#     TROIS configurations du banc tournaient sans cette ligne.
#
#  2. CALYPSO_PULSE_LATENCY_MSEC dans run_modules/70-l2.sh et
#     68-sidecar-mobile.sh. Le mobile ecrit une trame GSM (20 ms) par
#     snd_pcm_writei ; si le tampon du greffon ALSA-PulseAudio est trop court,
#     chaque ecriture rend -EPIPE et pq_alsa.c y repond par un snd_pcm_prepare()
#     muet, qui DEMONTE ET REMONTE le flux PulseAudio - 50 fois par seconde
```

```
#     pendant un appel. Mesure du 09/09, ecrivain cadence a 20 ms sur 10 s :
#     80 ms -> 16 remontages, 120 -> 19, 160 -> 1, 200 -> 0, 240 -> 0.
#     start-native.sh exporte 320, ce qui suffit ; on aligne quand meme le
#     defaut des forks, pour qui lance run.sh directement.
#
# git pull sur osmo-operator ne peut rien pour ces fichiers-la : ils
# appartiennent a qosmo-grgsm et qosmo-dsp, que la mise a jour resynchronise
# (« git reset --hard FETCH_HEAD »). Le COMMENT est dans scripts/voix-forks.sh,
# seul a savoir poser ces deux reglages ; ici on ne fait que l appeler, comme
# le service osmo-update (juste apres son fetch) et iso_modules/70-scripts.sh
# (sur le ROOTFS). Un seul endroit a corriger le jour ou ca rebouge.
# JUGE : `grep io-tch-format ~/.osmocom/bb/mobile_group1.cfg` apres un
# demarrage, et l index du sink-input du mobile qui doit rester FIXE pendant
# un appel (`pactl list short sink-inputs`, deux releves a 1 s).
osmo_poser_voix_forks() {
    [ "$(id -u)" -eq 0 ] || return 0
    local s=/opt/GSM/osmo-operator/scripts/voix-forks.sh
    [ -x "$s" ] || return 0
    "$s" | sed 's/^  \[voix\]/[OK]/'
    return 0
}
osmo_poser_voix_forks

# ── gnuarm, ALTERNATIVES GCC, c123 ET banc-* : RATTRAPAGE (2026-10-24) ────────
# Ce que build-iso.sh (70-scripts, 80-chroot) et l install native posent desormais,
# rejoue sur une machine deja installee. Sans apt ni reseau : seuls les compilateurs
# deja presents entrent dans les alternatives.
#   - /etc/profile.d/97-osmo-gnuarm.sh : /root/gnuarm/install/bin dans le PATH
#     (arm-elf-gcc 4.5.2 du supplement --gnuarm), sans effet tant qu il est absent ;
#   - update-alternatives gcc : gcc-9 (90), gcc-11 (110), gcc-13 (130). Un choix deja
#     fait (update-alternatives --config gcc) est garde ; a la creation, gcc-13, le
#     compilateur du noyau, pour DKMS ;
#   - c123 / osmo-c123 et les commandes du banc (banc-rssi, banc-trx, c123-1...)
#     dans /usr/local/bin, 98-osmo-banc-aliases.sh dans profile.d ;
#   - /usr/local/bin/osmo-glava (une copie, pas un lien) aligne sur tools/osmo-glava.sh.
osmo_poser_outils_banc() {
    [ "$(id -u)" -eq 0 ] || return 0
    local r=/opt/GSM/osmo-operator v
    [ -d "$r" ] || return 0
    [ -f "$r/scripts/osmo-gnuarm-profile.sh" ] \
        && install -m644 "$r/scripts/osmo-gnuarm-profile.sh" /etc/profile.d/97-osmo-gnuarm.sh
    [ -f "$r/scripts/osmo-banc-aliases.sh" ] \
        && install -m644 "$r/scripts/osmo-banc-aliases.sh" /etc/profile.d/98-osmo-banc-aliases.sh
    # WireGuard du WAN (« wg 1 », « wg N ... ») ; le paquet, lui, demande apt : install native / ISO
    [ -f "$r/scripts/osmo-wg-profile.sh" ] \
        && install -m644 "$r/scripts/osmo-wg-profile.sh" /etc/profile.d/97-osmo-wg.sh \
        && ln -sf "$r/tools/osmo-wg.sh" /usr/local/bin/osmo-wg
    [ -x "$r/tools/osmo-banc-cmd.sh" ] && bash "$r/tools/osmo-banc-cmd.sh" --install /usr/local/bin >/dev/null
    for v in c123 osmo-c123; do
        [ "$(readlink "/usr/local/bin/$v" 2>/dev/null)" = "$r/tools/osmo-c123-gtk.py" ] \
            || ln -sf "$r/tools/osmo-c123-gtk.py" "/usr/local/bin/$v"
    done
    [ -f /usr/local/bin/osmo-glava ] && [ -f "$r/tools/osmo-glava.sh" ] \
        && ! cmp -s "$r/tools/osmo-glava.sh" /usr/local/bin/osmo-glava \
        && install -m755 "$r/tools/osmo-glava.sh" /usr/local/bin/osmo-glava \
        && echo "  [bureau] osmo-glava aligne sur le depot (systemctl --user restart glava)"
    # alternatives gcc-9 / gcc-11 / gcc-13 (une seule definition : tools/osmo-gcc-alternatives.sh)
    [ -x "$r/tools/osmo-gcc-alternatives.sh" ] && bash "$r/tools/osmo-gcc-alternatives.sh"
    return 0
}
osmo_poser_outils_banc

case "${1:-}" in
    --quiet) exit 0 ;;
esac
```

```
# [2026-09-03] Le "git config --global http.version HTTP/1.1" qui etait ici est
# retire (voir Dockerfile) : on defait meme le reglage s il traine encore.
git config --global --unset http.version 2>/dev/null || true


# ── SANS TERMINAL : UNE FENETRE GTK ──────────────────────────────────────
# [2026-09-03] Ici, le script sortait en silence des qu il n avait pas de tty.
# C etait juste pour l animation -- mais lance depuis l ICONE DU BUREAU, il n a
# jamais de tty : la repose des icones se faisait, et l utilisateur ne voyait
# RIEN. Aucun retour, aucune erreur, rien : le double-clic ne repondait pas.
#
# On garde la regle qui a motive la sortie (pas de sequences de curseur hors
# terminal, elles rendent un journal illisible) et on repond en GTK quand il y a
# un serveur graphique. Sans terminal NI graphique -- cron, ssh sans X,
# /etc/profile.d en console -- on sort comme avant, en silence et en code 0.
#
# zenity vient de la variante desktop (build-iso.sh l installe) ; l absence du
# binaire n est donc pas une erreur, c est une image sans bureau.
if [ ! -t 1 ]; then
    if [ -n "${DISPLAY:-}${WAYLAND_DISPLAY:-}" ] && command -v zenity >/dev/null 2>&1; then
        # --pulsate et non un pourcentage : les etapes ci-dessus sont deja
        # faites quand on arrive ici, il n y a pas de progression a mesurer --
        # une barre chiffree mentirait. --auto-close pour que la fenetre parte
        # avec le flux, sans bouton a cliquer.
        {
            echo "# Recherche de la cellule (ARFCN)..." ; sleep 0.6
            echo "# Raccourcis du bureau reposes."      ; sleep 0.6
            echo "# Envoi du SMS de test..."            ; sleep 0.6
        } | zenity --progress --pulsate --auto-close --no-cancel \
                   --title="osmo-operator - mise a jour" \
                   --width=420 --text="Mise a jour en cours..." 2>/dev/null
        zenity --info --title="osmo-operator" --width=420 \
               --text="<b>SMS delivered</b> - MT end-to-end\n\nMessage : Bastien phone home\n\nLes raccourcis du
                ↪  bureau ont ete reposes." \
               2>/dev/null
    fi
    exit 0
fi

printf '\033[?25l'
trap 'printf "\033[?25h"' EXIT

ph='\033[1;33m☎\033[0m'
bars=('\033[2m___\033[0m' '\033[1;32m▄\033[0m\033[2m__\033[0m' '\033[1;32m▄▆\033[0m\033[2m_\033[0m'
 ↪  '\033[1;32m▄▆█\033[0m')
for b in "${bars[@]}"; do
    printf '\r  %b %b  \033[36mscanning ARFCN...\033[0m   ' "$ph" "$b"
    sleep 0.12
done
for ((p=0; p<=20; p++)); do
    printf '\r\033[K  %b %*s\033[1;36m✉\033[0m%*s %b' "$ph" "$p" '' "$((20-p))" '' "$ph"
    sleep 0.04
done
printf '\r\033[K  %b%21s%b  \033[1;32m✓ SMS delivered - MT end-to-end Message : Bastien phone home\033[0m\n'
 ↪  "$ph" '' "$ph"
```

3.27 /opt/GSM/osmo-operator/.deb-cache/.keep

identique à /opt/GSM/osmo-operator/reset

3.28 /opt/GSM/osmo-operator/.github/actions/docker-disk/action.yml

3397 octets, 74 lignes → 74 lignes

```yaml
name: "Disque pour docker"
description: >
  Fait de la place sur le runner, puis installe /var/lib/docker sur le plus
  grand disque disponible. A utiliser dans tout job qui manipule une image
  osmocom-nitb (11 Go).

inputs:
  need:
    description: "Go libres exiges pour /var/lib/docker"
    required: false
    default: "40"

runs:
  using: composite
  steps:
    # Le runner arrive avec ~14 Go d'outils dont aucun build de ce depot n'a
    # l'usage, et plusieurs images deja tirees dans /var/lib/docker.
    - name: Menage du runner
      shell: bash
      run: |
        sudo rm -rf /usr/share/dotnet /usr/local/lib/android \
          /opt/ghc /opt/hostedtoolcache/CodeQL \
          /usr/local/share/boost /usr/share/swift \
          /usr/local/graalvm /usr/local/share/chromium \
          /usr/local/lib/node_modules
        sudo apt-get remove -y '^dotnet-.*' 'php.*' 'mongodb.*' \
          'mysql.*' azure-cli google-cloud-cli 2>/dev/null || true
        sudo apt-get autoremove -y && sudo apt-get clean
        docker system prune -af --volumes
        df -h / /mnt || true

    # / n'offre qu'une trentaine de Go libres. /mnt est un second disque
    # d'environ 70 Go quasiment vide : c'est la que docker doit vivre.
    #
    # [2026-09-13] Le repli silencieux « on reste sur / » est SUPPRIME. Sur /,
    # le build va au bout de ses 75 etapes puis meurt en « exporting layers »,
    # SIGTERM du runner a court d'espace : exit 143, une heure et demie perdue,
    # et pour seule trace une ligne noyee ici. S'il n'y a nulle part la place,
    # l'echec est certain : autant le dire en trois secondes, avec son motif.
    - name: Docker data-root sur /mnt
      shell: bash
      env:
        NEED: ${{ inputs.need }}
      run: |
        AVAIL=$(df -BG --output=avail /mnt 2>/dev/null | tail -1 | tr -dc '0-9')
        ROOT_AVAIL=$(df -BG --output=avail / | tail -1 | tr -dc '0-9')
        if [ "${AVAIL:-0}" -lt "$NEED" ]; then
          if [ "${ROOT_AVAIL:-0}" -ge "$NEED" ]; then
            echo "::warning::/mnt inutilisable (${AVAIL:-0} Go) mais / offre ${ROOT_AVAIL} Go - on reste sur /."
            df -h /; exit 0
          fi
          echo "::error::Aucun disque de ${NEED} Go pour /var/lib/docker (/mnt ${AVAIL:-0} Go, / ${ROOT_AVAIL:-0}
↪ Go). L'image de base en pese 11 : ce job echouerait en exit 143 apres plus d'une heure."
          df -h / /mnt || true
          exit 1
        fi
        sudo systemctl stop docker.service docker.socket
        sudo mkdir -p /mnt/docker
        sudo mv /var/lib/docker/* /mnt/docker/ 2>/dev/null || true
        sudo jq '. + {"data-root":"/mnt/docker"}' /etc/docker/daemon.json \
          > /tmp/daemon.json 2>/dev/null \
          || echo '{"data-root":"/mnt/docker"}' > /tmp/daemon.json
        sudo mv /tmp/daemon.json /etc/docker/daemon.json
        sudo systemctl start docker.service
        # Le demon a pu redemarrer sur l'ancienne racine (daemon.json refuse,
        # mv partiel) : sans cette verification on ne l'apprendrait qu'a
```

```
        # l'export, sous la forme d'un 143 muet.
        for _ in $(seq 30); do docker info >/dev/null 2>&1 && break; sleep 2; done
        DR=$(docker info -f '{{.DockerRootDir}}')
        if [ "$DR" != "/mnt/docker" ]; then
          echo "::error::docker a redemarre sur ${DR}, pas /mnt/docker."
          exit 1
        fi
        echo "data-root = $DR"
        df -h / /mnt
```

3.29 /opt/GSM/osmo-operator/.github/workflows/build-arm.yml

13854 octets, 305 lignes → 305 lignes

```
name: Build images SD (arm64 / Armbian - Raspberry Pi 4, Orange Pi 5)

# La MEME pile que les ISO x86, construite pour arm64 et livree en IMAGES SD
# par le framework officiel github.com/armbian/build : c'est le chemin --arm de
# build-iso.sh, qui passe la main a armbian/build-armbian.sh. Armbian fabrique
# noyau, BSP, u-boot/firmware, rootfs et image pour chaque carte ; la pile y
# est posee par armbian/userpatches/customize-image.sh, depuis l'image Docker
# arm64 publiee par docker-arm.yml (job amont).
#
# [2026-10-24] Une image PAR CARTE, un job par carte (matrice) : Armbian ne
# construit qu'une carte par compile.sh, et sept cartes a la suite ne tiendraient
# pas dans les 6 h d'un job heberge. Les cartes :
#   rpi4b                      Raspberry Pi 4
#   orangepi5, orangepi5b, orangepi5-plus, orangepi5pro,
#   orangepi5-max, orangepi5-ultra      toutes les Orange Pi 5 (la Max et
#                              l'Ultra ont leur propre fichier de carte)
# La branche noyau est choisie par carte (build-armbian.sh : current, sinon la
# premiere proposee - edge pour la 5B).
#
# ── POURQUOI UN RUNNER arm64 ET PAS L'EMULATION ──────────────────────────
# Sur un hote x86, le chroot de customize-image.sh tourne sous qemu-user :
# c'est lent, et un job heberge est TUE a 6 heures. Les runners arm64 heberges
# (ubuntu-24.04-arm) font tout NATIVEMENT. C'est le defaut ; l'entree `runner`
# permet de demander un hote x86 pour verifier le chemin emule.
#
# ── LE BUREAU ────────────────────────────────────────────────────────────
# desktop=true : GNOME (DESKTOP_ENVIRONMENT=gnome, DESKTOP_TIER=minimal, via
# armbian/configng) + le bureau du banc (fond, Conky, encart, icones, c123).
# GLava n'y est pas : le cercle NCS veut OpenGL >= 4.2, absent des GPU de ces
# cartes sous Mesa. Pas de 4G non plus (srsRAN/Open5GS non compiles en aarch64).

on:
  push:
    tags: ["rpi-*", "arm-*", "opi-*"]
  workflow_dispatch:
    inputs:
      version:
        description: "Release tag (v0.1-8 kept in reserve)"
        type: string
        default: "v0.1-9"
      boards:
       description: "Cartes : all, rpi4, opi5, ou des noms Armbian separes par des virgules (orangepi5-max,rpi4b)"
        type: string
        default: "all"
      desktop:
        description: "Bureau GNOME + bureau du banc (--desktop)"
        type: boolean
        default: true
      runner:
        description: "Hote de construction"
```

```
        type: choice
        default: ubuntu-24.04-arm
        options:
          - ubuntu-24.04-arm   # natif aarch64 : le seul qui tienne dans les 6 h
          - ubuntu-24.04       # x86 + qemu-user : lent, sera coupe a 6 h
      release:
        description: "Create GitHub Release"
        type: boolean
        default: true
      no_cache:
        description: "Rebuild Docker images without cache (--no-cache)"
        type: boolean
        default: false

concurrency:
  group: build-arm-${{ github.ref }}
  cancel-in-progress: false

jobs:
  # ── EN AMONT : l'image Docker arm64 de la pile (docker-arm.yml) ──────────
  docker-arm:
    permissions:
      contents: read
      packages: write
    uses: ./.github/workflows/docker-arm.yml
    with:
      rebuild: false
      no_cache: ${{ github.event.inputs.no_cache == 'true' }}

  # ── La liste des cartes -> la matrice ────────────────────────────────────
  # Memes groupes que build-armbian.sh (--board=rpi4|opi5|all). Sur un push de
  # tag les entrees sont VIDES : toutes les cartes, avec bureau.
  boards:
    runs-on: ubuntu-24.04
    outputs:
      matrix: ${{ steps.m.outputs.matrix }}
      desktop: ${{ steps.m.outputs.desktop }}
    steps:
      - id: m
        run: |
          IN='${{ github.event.inputs.boards }}'; IN=${IN:-all}
          OPI5="orangepi5 orangepi5b orangepi5-plus orangepi5pro orangepi5-max orangepi5-ultra"
          L=""
          for b in ${IN//,/ }; do
            case "$b" in
              rpi4|rpi4b) L="$L rpi4b" ;;
              opi5)       L="$L $OPI5" ;;
              all)        L="$L rpi4b $OPI5" ;;
              *)          L="$L $b" ;;
            esac
          done
          M=$(echo $L | tr ' ' '\n' | awk 'NF && !vu[$0]++' | jq -R . | jq -cs .)
          echo "matrix=$M" >> "$GITHUB_OUTPUT"
          D='${{ github.event.inputs.desktop }}'
          echo "desktop=${D:-true}" >> "$GITHUB_OUTPUT"
          echo "cartes : $M ; bureau : ${D:-true}"

  build:
    needs: [docker-arm, boards]
    runs-on: ${{ github.event.inputs.runner || 'ubuntu-24.04-arm' }}
    timeout-minutes: 360
    strategy:
      fail-fast: false            # une carte qui casse n'emporte pas les autres
      matrix:
        board: ${{ fromJSON(needs.boards.outputs.matrix) }}
    permissions:
```

```
    contents: read
    packages: read

  steps:
    - name: Checkout
      uses: actions/checkout@v5
      with:
        fetch-depth: 0

    - name: Architecture de l'hote
      run: |
        HOST=$(dpkg --print-architecture)
        echo "hote : $HOST  ($(uname -m), $(nproc) coeurs, $(free -g | awk '/^Mem:/{print $2}') Go) ; carte ${{
↪ matrix.board }}"
        if [ "$HOST" = "arm64" ]; then
          echo "::notice::Hote aarch64 : construction Armbian NATIVE."
        else
          echo "::warning::Hote $HOST : le chroot arm64 sera EMULE (qemu-user-static). Ce job est coupe a 6 h."
        fi

    # Menage + /var/lib/docker sur le plus grand disque.
    - uses: ./.github/actions/docker-disk
      with:
        need: "40"

    # Le clone armbian/build (cache noyau, rootfs, image) sur le disque qui a
    # le plus de place : une image Armbian avec bureau en demande ~30 Go.
    - name: Repertoire Armbian
      run: |
        pick() { df -P "$1" 2>/dev/null | awk 'NR==2{print $4}'; }
        BASE=/; [ "$(pick /mnt || echo 0)" -gt "$(pick /)" ] 2>/dev/null && BASE=/mnt
        D="$BASE/armbian-build"
        sudo mkdir -p "$D"
        echo "ARMBIAN_DIR=$D" >> "$GITHUB_ENV"
        echo "armbian/build -> $D ($(df -h --output=avail "$BASE" | tail -1 | tr -d ' ') libres)"

    - name: Swap anti-OOM
      run: |
        pick() { df -P "$1" 2>/dev/null | awk 'NR==2{print $4}'; }
        AV_ROOT=$(pick /); AV_MNT=$(pick /mnt)
        BASE=/; [ -n "$AV_MNT" ] && [ "${AV_MNT:-0}" -gt "${AV_ROOT:-0}" ] && BASE=/mnt
        FREE_GB=$(( $(pick "$BASE") / 1024 / 1024 ))
        SWAP_GB=8; THIRD=$(( FREE_GB / 4 ))
        [ "$SWAP_GB" -gt "$THIRD" ] && SWAP_GB=$THIRD
        [ "$SWAP_GB" -lt 2 ] && SWAP_GB=2
        SF="$BASE/osmo-swapfile"
        sudo swapoff "$SF" 2>/dev/null || true; sudo rm -f "$SF"
        sudo fallocate -l ${SWAP_GB}G "$SF" 2>/dev/null \
          || sudo dd if=/dev/zero of="$SF" bs=1M count=$((SWAP_GB*1024)) status=none
        sudo chmod 600 "$SF"; sudo mkswap "$SF" >/dev/null; sudo swapon "$SF"
        echo "swap +${SWAP_GB} Go sur $BASE"

    # La base arm64 vient de GHCR (job docker-arm) : tiree ICI, sous le compte
    # runner qui porte les identifiants GHCR (build-iso.sh tourne sous sudo).
    - name: Log in to GHCR
      uses: docker/login-action@v4
      with:
        registry: ghcr.io
        username: ${{ github.actor }}
        password: ${{ secrets.GITHUB_TOKEN }}

    - name: Image de base arm64 depuis GHCR
      run: |
        REF='${{ needs.docker-arm.outputs.image }}'
        docker pull --platform linux/arm64 "$REF"
        echo "BASE_REF=$REF" >> "$GITHUB_ENV"
```

```
        echo "::notice::Base arm64 : $REF"

    # build-iso.sh --arm -> armbian/build-armbian.sh (iso_modules/00-options.sh).
    # --skip-build=REF : la pile vient de l'image tiree, rien n'est compile ici.
    - name: Build image Armbian (${{ matrix.board }})
      run: |
        chmod +x build-iso.sh armbian/build-armbian.sh
        ARGS="--arm --board=${{ matrix.board }} --skip-build=$BASE_REF"
        [ "${{ needs.boards.outputs.desktop }}" = "true" ] && ARGS="$ARGS --desktop"
        echo "::notice::build-iso.sh $ARGS"
        # sudo env ... : sudoers est en env_reset sans SETENV sur un runner.
        sudo env ARMBIAN_DIR="$ARMBIAN_DIR" ./build-iso.sh $ARGS
        df -h / /mnt || true

    # Le nom d'Armbian (Armbian-unofficial_<version>_<Carte>_noble_<branche>_<noyau>[_gnome_desktop].img)
    # devient un nom stable par carte, pour les sommes et la Release.
    - name: Renommer, sommes, compresser
      run: |
        IMG=$(ls -t "$ARMBIAN_DIR"/output/images/*.img 2>/dev/null | head -1)
        [ -n "$IMG" ] || { echo "::error::pas d image dans $ARMBIAN_DIR/output/images/"; exit 1; }
        SUF=""; [ "${{ needs.boards.outputs.desktop }}" = "true" ] && SUF="-desktop"
        OUT="osmo-operator-${{ matrix.board }}${SUF}.img"
        sudo mv "$IMG" "$OUT"; sudo chown "$(id -u):$(id -g)" "$OUT"
        echo "$(basename "$IMG") -> $OUT ($(du -h "$OUT" | cut -f1))"
        sha256sum "$OUT" > "SHA256SUMS-${{ matrix.board }}"
        md5sum    "$OUT" > "MD5SUMS-${{ matrix.board }}"
        zstd -T0 -9 --sparse --rm -q -o "$OUT.zst" "$OUT"
        ls -lh "$OUT.zst"

    # Une Release GitHub refuse tout fichier de plus de 2 Gio : on ne decoupe
    # QUE ce qui depasse. SHA256SUMS porte sur l'image brute.
    - name: Split over 2 GiB
      run: |
        LIMIT=$((2 * 1024 * 1024 * 1024))
        : > "PARTS-${{ matrix.board }}.sha256"
        for z in ./osmo-operator-*.img.zst; do
          [ -f "$z" ] || continue
          if [ "$(stat -c%s "$z")" -gt "$LIMIT" ]; then
            split -b 1900M -d -a 2 "$z" "${z}.part-"
            sha256sum "${z}".part-* >> "PARTS-${{ matrix.board }}.sha256"
            rm -f "$z"
          fi
        done
        ls -lh ./osmo-operator-*.img.zst* || true

    - name: Upload artifacts
      uses: actions/upload-artifact@v6
      with:
        name: osmo_egprs-${{ matrix.board }}
        path: |
          osmo-operator-*.img.zst*
          SHA256SUMS-*
          MD5SUMS-*
          PARTS-*.sha256
        retention-days: 14
        compression-level: 0

# ── La Release : toutes les cartes reussies, sommes fusionnees ──────────
release:
  needs: [boards, build]
  if: always() && needs.build.result != 'cancelled' && (startsWith(github.ref, 'refs/tags/') ||
↪ (github.event_name == 'workflow_dispatch' && github.event.inputs.release == 'true'))
  runs-on: ubuntu-24.04
  permissions:
    contents: write
  steps:
```

```
      - uses: actions/download-artifact@v5
        with:
          pattern: osmo_egprs-*
          merge-multiple: true
      - name: Fusionner les sommes
        run: |
          ls osmo-operator-*.img.zst* >/dev/null 2>&1 || { echo "::error::aucune image produite"; exit 1; }
          cat SHA256SUMS-* > SHA256SUMS; cat MD5SUMS-* > MD5SUMS; cat PARTS-*.sha256 > PARTS.sha256 2>/dev/null ||
↪ : > PARTS.sha256
          cat SHA256SUMS
      - name: Create GitHub Release
        uses: softprops/action-gh-release@v3
        with:
          # Pas github.ref_name tout seul : sur un workflow_dispatch il vaut « main »
          # et l'action creerait un TAG main, ambigu avec la branche.
          tag_name: ${{ startsWith(github.ref, 'refs/tags/') && github.ref_name || github.event.inputs.version ||
↪ 'v0.1-9' }}
          name: "osmo_egprs — images SD arm64 (Armbian : Raspberry Pi 4, Orange Pi 5) — ${{ startsWith(github.ref,
↪ 'refs/tags/') && github.ref_name || github.event.inputs.version || 'v0.1-9' }}"
          draft: false
          prerelease: ${{ !(startsWith(github.ref, 'refs/tags/v') || startsWith(github.event.inputs.version, 'v'))
↪ }}
          body: |
            ## osmo_egprs — images SD arm64 (Armbian)

            La meme pile que les ISO x86, construite pour aarch64 par le framework
            [armbian/build](https://github.com/armbian/build) (Ubuntu 24.04 noble).
            Une image **persistante** par carte, ${{ needs.boards.outputs.desktop == 'true' && 'avec le bureau
↪ GNOME du banc (fond GSM LAB, Conky, icones, `c123`)' || 'sans bureau (console et ssh)' }}.

            | Image | Carte |
            |---|---|
            | `osmo-operator-rpi4b*.img.zst` | Raspberry Pi 4 |
            | `osmo-operator-orangepi5*.img.zst` | Orange Pi 5 |
            | `osmo-operator-orangepi5b*.img.zst` | Orange Pi 5B (noyau edge) |
            | `osmo-operator-orangepi5-plus*.img.zst` | Orange Pi 5 Plus |
            | `osmo-operator-orangepi5pro*.img.zst` | Orange Pi 5 Pro |
            | `osmo-operator-orangepi5-max*.img.zst` | Orange Pi 5 Max |
            | `osmo-operator-orangepi5-ultra*.img.zst` | Orange Pi 5 Ultra |

            ### Graver

            ```bash
            zstd -dc osmo-operator-<carte>.img.zst \
              | sudo dd of=/dev/sdX bs=4M status=progress conv=fsync
            ```

            Au premier demarrage Armbian etend la racine a la taille de la carte.
            Comptes : `root:osmo`, `osmocom:osmo`.

            ### Verifier

            `SHA256SUMS` porte sur l'image **brute** : `zstd -d <image>.zst && sha256sum -c SHA256SUMS
↪ --ignore-missing`.
            Ce qui depasse 2 Gio est decoupe (`PARTS.sha256`) : `cat <image>.zst.part-* > <image>.zst`.

            ### Ce que ces images n'ont pas

            Pas de 4G (srsRAN et Open5GS ne sont pas compiles pour aarch64) ni de
            GLava (OpenGL >= 4.2 requis). La 2G/EDGE est entiere.
          files: |
            osmo-operator-*.img.zst*
            SHA256SUMS
            MD5SUMS
            PARTS.sha256
```

3.30 /opt/GSM/osmo-operator/.github/workflows/build-iso.yml

25285 octets, 462 lignes → 462 lignes

```yaml
name: Build ISO

# Trois images, une seule chaine :
#   interstp.iso            le hub SS7 (PC 0.0.0), aucun operateur
#   osmo-operator.iso       un noeud - UNE image pour les 9 : le numero se
#                           choisit au demarrage (start-native.sh --node N)
#   osmo-operator-lite.iso  le meme noeud, construit depuis l'image d'execution
#                           elaguee (Dockerfile.lite) : les ateliers de
#                           compilation de /opt/GSM ne partent pas dans le
#                           squashfs. Ce qui tourne est identique.
#
# Les trois dans UN SEUL job, et dans cet ordre : la lite se greffe sur l'image
# osmocom-run que la passe operateur vient de construire. Dans un job separe, le
# runner repartirait de zero - deux heures de compilation pour le meme resultat.
#
# --kb=us : build-iso.sh pose sa question du clavier AU DEBUT. Sans terminal il
# prend deja son defaut sans attendre - le job ne peut pas se bloquer dessus -
# mais on l'ecrit pour que l'image de CI ne dependent pas de ce defaut.

on:
  push:
    # [2026-10-01] "v*" et pas "v*.*.*" : les versions du depot s ecrivent
    # v0.1-6 (UN point) - l ancien motif ne declenchait jamais sur leur push.
    tags: ["v*", "iso-*"]
  workflow_dispatch:
    inputs:
      # [2026-10-18] Version de la release, en entree : on la choisit au lancement
      # au lieu de modifier le workflow. v0.1-8 est gardee en reserve : on passe
      # directement de v0.1-7 a v0.1-9 (la saisir ici si besoin).
      version:
        description: "Release tag (v0.1-8 kept in reserve)"
        type: string
        default: "v0.1-9"
      release:
        description: "Create GitHub Release"
        type: boolean
        default: true
      no_cache:
        description: "Rebuild Docker images without cache (--no-cache)"
        type: boolean
        default: true
      build_docker:
        description: "Construire l'image de base ici, au lieu de la tirer de GHCR (--build-docker)"
        type: boolean
        default: false
      mode:
        description: "Quelles images construire"
        type: choice
        default: all
        options:
          - all       # les quatre en une passe
          - stp       # le hub SS7 seul (interstp.iso)
          - base      # le noeud complet (osmo-operator.iso)
          - lite      # le noeud sans les ateliers (osmo-operator-lite.iso)
          - desktop   # le noeud avec GNOME (osmo-operator-desktop.iso)
      pmos_image:
        description: "Embed the postmarketOS VM image in the DESKTOP ISO only (~1.7 GB, from the pmos-image
↪  release)"
        type: boolean
        default: false

jobs:
```

```yaml
build-iso:
  # [2026-09-13] noble, PAS jammy. L'ISO et le Dockerfile sont en 24.04
  # (ISO_UBUNTU par defaut dans iso_modules/00-options.sh, FROM ubuntu:24.04),
  # mais le runner etait en 22.04 : l'hote du build etait donc d'une suite
  # differente de ce qu'il construit. Deux consequences concretes -
  #   - le debootstrap de jammy ne connait pas « noble » : 40-rootfs.sh s'en
  #     tire en liant le script gutsy, mais c'est un contournement ;
  #   - packaging/osmo-deb.sh lit VERSION_CODENAME de l'HOTE pour nommer les
  #     .deb : tout ce qui se compile cote hote sortait en « ~jammy », et un
  #     .deb jammy n'est jamais pose sur un rootfs noble (osmo-deb.sh:43).
  # Un runner noble aligne les trois.
  runs-on: ubuntu-24.04
  timeout-minutes: 360
  permissions:
    contents: write
    packages: read

  steps:
    - name: Checkout
      uses: actions/checkout@v5
      with:
        fetch-depth: 0

    # L'EMPREINTE DU CONTEXTE. La meme commande que docker.yml (job `base`) :
    # elle nomme l'image de base publiee sur GHCR, et elle sert de cle au
    # cache .deb. Calculee ici, une seule fois, parce que les deux en ont
    # besoin et qu'elles doivent parler de la meme chose.
    - name: Empreinte du contexte
      id: key
      run: |
        KEY=$(git ls-files -- Dockerfile packaging configs scripts services \
                              helpers opt-gsm tools patches \
              | sort | xargs sha256sum | sha256sum | cut -c1-16)
        echo "key=$KEY" >> "$GITHUB_OUTPUT"
        echo "Empreinte du contexte : $KEY"

    # Les trois ISO plus les images Docker intermediaires tiennent tout juste
    # dans un runner : on fait la place avant, pas apres l'echec.
    - name: Free disk space
      run: |
        sudo rm -rf /usr/share/dotnet /usr/local/lib/android \
          /opt/ghc /opt/hostedtoolcache/CodeQL \
          /usr/local/share/boost /usr/share/swift \
          /usr/local/graalvm /usr/local/share/chromium \
          /usr/local/lib/node_modules
        sudo apt-get remove -y '^dotnet-.*' 'php.*' 'mongodb.*' \
          'mysql.*' azure-cli google-cloud-cli 2>/dev/null || true
        sudo apt-get autoremove -y
        sudo apt-get clean
        sudo docker image prune -af || true
        df -h / /mnt

    # [2026-09-13] Le job mourait en « exporting layers » sur exit 143 : pas
    # une erreur de docker (il n'ecrit rien), un SIGTERM recu de l'exterieur -
    # la partition / du runner arrivait a saturation et le runner lui-meme
    # tombait. Trois gros consommateurs partageaient / :
    #   /var/lib/docker    base ~11 Go, et on en construit QUATRE variantes
    #   /var/tmp/iso-build-$$   le rootfs decompresse (build-iso.sh:33)
    #   le workspace       les quatre .iso
    # Le runner a un SECOND disque, /mnt, ~70 Go, inutilise. On y met le store
    # docker : le rootfs et les ISO gardent / pour eux seuls.
    - name: Docker data-root sur /mnt
      run: |
        sudo systemctl stop docker docker.socket containerd || true
        # Deplacer, pas recopier : le store contient deja les images de base
        # tirees par le runner. Si /mnt/docker existe (re-run sur le meme
```

```
    # runner), on le garde plutot que d'ecraser un store valide.
    [ -d /mnt/docker ] || sudo mv /var/lib/docker /mnt/docker
    echo '{ "data-root": "/mnt/docker" }' | sudo tee /etc/docker/daemon.json
    sudo systemctl start docker
    docker info 2>/dev/null | grep -i 'docker root dir'
    df -h / /mnt

# Un exit 143 ne dit rien de ce qui manquait. Ce temoin ecrit disque et
# memoire toutes les 30 s dans un fichier relu en fin de job (if: always),
# y compris quand l'etape de build est tuee sans un mot.
- name: Temoin disque/memoire
  run: |
    nohup bash -c 'while true; do
      printf "[%s] / %s  /mnt %s  mem %s\n" \
        "$(date -u +%H:%M:%S)" \
        "$(df -h --output=avail / | tail -1 | tr -d " ")" \
        "$(df -h --output=avail /mnt | tail -1 | tr -d " ")" \
        "$(free -m | awk "/^Mem:/{print \$3\"/\"\$2\" Mo\"}")"
      sleep 30
    done' > "$RUNNER_TEMP/watchdog.log" 2>&1 &
    echo "temoin demarre -> $RUNNER_TEMP/watchdog.log"

# ── Le cache .deb entre deux runs ──────────────────────────────────
# Ce que le Dockerfile compile (libosmocore, osmo-*, gnuradio, gr-gsm,
# QEMU, osmocom-bb, pmbootstrap...) sort en paquet .deb dans
# /var/cache/osmo-debs (packaging/osmo-deb.sh). build.sh recopie ce
# dossier dans .deb-cache/ AVANT le build - le Dockerfile le prend par
# `COPY .deb-cache/ /var/cache/osmo-debs/` - et y remet APRES les paquets
# que l'image vient de produire (debs_from_image). Sur l'hote c'est ce
# qui fait passer un rebuild de 1h23 a quelques minutes ; un runner,
# lui, repart vierge a chaque fois. On lui rend la memoire.
#
# Le dossier appartient a root (build.sh tourne sous sudo) : il faut le
# creer AVANT que actions/cache y ecrive, et le rendre au runner APRES
# le build pour que le post-step puisse le relire.
# SUR /mnt, PAS SUR /. Ce job est le seul des deux ou / est la partition
# serree : le rootfs decompresse, le squashfs et les quatre ISO y tiennent
# tout juste. Les 2-4 Go du cache .deb iraient s'ajouter a ca, alors que
# /mnt en a 68 de libres. build.sh, build-iso.sh et 50-injection-image.sh
# lisent tous OSMO_DEB_CACHE : il suffit de le leur dire.
# Repli sur / si le deplacement de docker plus haut a renonce a /mnt.
- name: Preparer le cache .deb
  run: |
    # Pas de repli conditionnel : `path:` du step suivant est statique,
    # un chemin qui divergerait ici ne serait pas celui qu'on sauvegarde.
    # /mnt existe sur tout runner heberge ; ailleurs, mkdir -p le cree sur
    # / et on retombe simplement sur l'ancien comportement.
    CACHE=/mnt/osmo-debs
    sudo mkdir -p "$CACHE"
    sudo chown -R "$(id -u):$(id -g)" "$CACHE"
    echo "OSMO_DEB_CACHE=$CACHE" >> "$GITHUB_ENV"
    echo "cache .deb : $CACHE"

- name: Cache .deb (osmo-deb)
  uses: actions/cache@v5
  with:
    path: /mnt/osmo-debs
    # CLE = L'EMPREINTE DU CONTEXTE, pas le numero du run.
    # [2026-09-13] C'etait `${{ github.run_id }}`, donc une entree NEUVE a
    # chaque passage. A 2-4 Go l'entree et 10 Go de quota par depot,
    # l'eviction LRU detruisait le cache au bout de trois runs. Les .deb
    # portent leur nom et leur version, osmo-deb.sh trie : le prefixe
    # reprend la plus recente, et ce qui n'a pas bouge reste valable.
    # [2026-09-14] LA GENERATION (-g2-) EXISTE POUR JETER UN CACHE DEVENU
    # FAUX. `restore-keys` repeche la derniere entree du prefixe meme
    # quand la cle exacte a change : un osmo-build-srsran compile en
```

```
    # -march=native AVANT le 14/09 revenait donc a chaque run, passait le
    # `osmo-deb install srsran`, et l ISO repartait avec un srsenb qui
    # fait « Illegal instruction » ailleurs - jusqu a ce que le controle
    # objdump de iso_modules/88-lte-pmos.sh l ecarte et recompile une
    # heure, A CHAQUE FOIS. La version des .deb porte desormais son ISA
    # (tools/osmo-lte-install.sh, srs_deb_version), ce qui empeche la
    # confusion pour de bon ; ce marqueur coupe en plus le lien avec les
    # entrees d avant. L INCREMENTER quand un cache doit etre abandonne.
    key: osmo-debs-noble-g2-${{ steps.key.outputs.key }}
    restore-keys: |
      osmo-debs-noble-g2-

- name: Log in to GHCR
  uses: docker/login-action@v4
  with:
    registry: ghcr.io
    username: ${{ github.actor }}
    password: ${{ secrets.GITHUB_TOKEN }}

# ── L'image de base vient de GHCR quand elle y est deja ──────────────
# Le workflow Docker (docker.yml) publie osmocom-nitb:base-<empreinte>,
# l'empreinte etant le SHA de tout ce que le Dockerfile prend dans le
# contexte. MEME COMMANDE ici, au caractere pres : si elle donne la meme
# valeur, l'image publiee correspond exactement a ce commit, et il n'y a
# aucune raison de recompiler 1h23 pour la refabriquer a l'identique.
# build-iso.sh la tague osmocom-nitb et la chaine continue sans rien
# savoir de sa provenance (--skip-build, iso_modules/21-docker-build.sh).
#
# Le pull se fait ICI, sous le compte runner qui porte les identifiants
# GHCR - build-iso.sh tourne sous sudo, et le ~/.docker de root n'en a
# pas. L'image etant alors locale, le module ne retire rien.
- name: Image de base (Docker Hub, puis GHCR)
  run: |
    if [ "${{ github.event.inputs.build_docker }}" = "true" ]; then
      echo "ISO_DOCKER_SRC=--build-docker" >> "$GITHUB_ENV"
      echo "::notice::--build-docker demande : l'image de base sera construite ici."
      exit 0
    fi
    KEY='${{ steps.key.outputs.key }}'
    REPO=$(echo '${{ github.repository }}' | tr '[:upper:]' '[:lower:]')
    # [2026-10-10] ORDRE : Docker Hub (bastienbaranoff/norf_gsm:latest)
    # d'abord, l'image GHCR de l'empreinte ensuite, la compilation en
    # dernier. L'image retenue est taguee osmocom-nitb.
    HUB="bastienbaranoff/norf_gsm:latest"
    GHCR="ghcr.io/${REPO}/osmocom-nitb:base-${KEY}"
    REF=""
    if docker pull "$HUB"; then
      REF="$HUB"
      echo "::notice::Base tiree de Docker Hub : $HUB - ne suit pas forcement ce commit."
    elif docker manifest inspect "$GHCR" >/dev/null 2>&1 && docker pull "$GHCR"; then
      REF="$GHCR"
      echo "::notice::Docker Hub indisponible, base tiree de GHCR : $GHCR - pas de compilation."
    fi
    if [ -n "$REF" ]; then
      docker tag "$REF" osmocom-nitb
      echo "ISO_DOCKER_SRC=--skip-build=$REF" >> "$GITHUB_ENV"
      # Les .deb de l'image rejoignent le cache de l'hote : c'est la que
      # 50-injection-image.sh va les chercher pour poser la pile dans le
      # rootfs (il sait retomber sur l'image, mais un dpkg vaut mieux
      # qu'un docker cp integral).
      mkdir -p /tmp/debs-out
      cid=$(docker create "$REF" /bin/true)
      docker cp "$cid:/var/cache/osmo-debs/." /tmp/debs-out/ 2>/dev/null || true
      docker rm "$cid" >/dev/null
      cp -f /tmp/debs-out/osmo-build-*.deb "$OSMO_DEB_CACHE/" 2>/dev/null || true
      rm -rf /tmp/debs-out
```

```
      echo "cache .deb : $(find "$OSMO_DEB_CACHE" -name 'osmo-build-*.deb' | wc -l) paquets"
    else
      # --build-docker EXPLICITE, et pas une chaine vide. [2026-09-14]
      # build-iso.sh tire desormais par defaut, et son dernier recours est
      # l'image Docker Hub, qui ne suit pas le commit : laisser vide
      # publierait ici une ISO batie sur une pile qui n'est pas celle de
      # ce commit, sans que rien ne le dise. Quand l'empreinte n'a pas
      # d'image, la seule reponse juste est de compiler.
      echo "ISO_DOCKER_SRC=--build-docker" >> "$GITHUB_ENV"
      echo "::notice::Ni Docker Hub ni base GHCR pour l'empreinte $KEY - construction locale."
    fi
    df -h / /mnt

# Les paquets hote (squashfs-tools, xorriso, grub*, mtools, debootstrap...)
# sont installes par build-iso.sh lui-meme.

# Le hub d'abord : il ne depend ni de build.sh ni de l'image osmocom-run,
# donc un echec de son cote se voit en minutes plutot qu'en heures.
# --all : interstp + operator + operator-lite + operator-desktop en UNE
# passe. build-iso.sh greffe lite et desktop sur l'image osmocom-run que la
# passe operateur vient de produire, et retire --no-cache pour ces passes
# greffees : inutile de separer en plusieurs etapes. On rend le cache buildx
# avant - QUATRE ISO plus les images intermediaires, le disque du runner
# decide si le job passe.
- name: Build ISOs
  run: |
    chmod +x build-iso.sh
    df -h /
    # ── QUELLES IMAGES ──────────────────────────────────────
    # Sur un push de tag les entrees de workflow_dispatch sont VIDES :
    # le defaut reste "all", c'est-a-dire le comportement d'avant ce
    # choix. Les autres modes servent a reprendre UNE image sans
    # repayer les trois autres - un theme plymouth a revoir, un
    # installeur a retester, le hub qui sort en minutes.
    #
    # CE QUE COUTE UN MODE SEUL : hors "all", build-iso.sh n'a pas de
    # rootfs frere a reprendre (le greffage lite/desktop sur le rootfs
    # de la normale n'existe que dans la passe --all). Chaque image
    # refait donc son rootfs de zero. Pour les quatre, "all" reste
    # nettement moins cher que quatre runs.
    case "${{ github.event.inputs.mode }}" in
      ""|all)  ISO_ARGS="--all" ;;
      stp)     ISO_ARGS="--role=interstp" ;;
      base)    ISO_ARGS="--role=operator --output=osmo-operator.iso" ;;
      lite)    ISO_ARGS="--role=operator --lite --output=osmo-operator-lite.iso" ;;
      desktop) ISO_ARGS="--role=operator --desktop --output=osmo-operator-desktop.iso" ;;
      *) echo "::error::mode inconnu : ${{ github.event.inputs.mode }}"; exit 2 ;;
    esac
    echo "::notice::mode ${{ github.event.inputs.mode || 'all' }} -> build-iso.sh $ISO_ARGS"
    # [2026-09-13] DEFAUT INVERSE. C'etait NO_CACHE="--no-cache" avec un
    # repli sur "" quand l'entree valait "false" - mais sur un push les
    # entrees de workflow_dispatch sont VIDES, jamais "false" : chaque
    # push forcait donc --no-cache, donc OSMO_DEB_REFRESH=1, donc les 51
    # RUN de compilation refaits integralement. 1h23 a chaque fois, pour
    # des sources qui n'avaient pas bouge. Le plein rebuild reste
    # accessible, mais il se demande.
    NO_CACHE=""
    [ "${{ github.event.inputs.no_cache }}" = "true" ] && NO_CACHE="--no-cache"
    # L'image de la VM postmarketOS (qemu-amd64.img.zst, 1,7 Go) n'existe
    # pas sur un runner : 88-lte-pmos.sh la telecharge depuis l'asset de
    # Release quand on lui donne l'URL. [2026-10-09] Elle n'est embarquee
    # que dans l'ISO desktop (88-lte-pmos.sh gate sur ISO_DESKTOP), pas
    # dans interstp, operator ni lite. Sans elle, le premier clic sur le
    # telephone fait un "pmbootstrap install" complet.
    # sudo env ... : sur un runner, sudoers est en env_reset sans SETENV,
    # « sudo -E » et « sudo VAR=x » peuvent etre refuses ; env(1) lance en
```

```
          # root, lui, passe toujours.
          sudo env OSMO_PMOS_IMAGE_URL="$OSMO_PMOS_IMAGE_URL" \
                   OSMO_HOST_DOCKER="$OSMO_HOST_DOCKER" \
                   OSMO_DEB_CACHE="$OSMO_DEB_CACHE" \
               ./build-iso.sh $NO_CACHE $ISO_DOCKER_SRC --kb=us $ISO_ARGS
          # `|| true` : hors mode "all" les autres .iso n'existent pas, et un
          # ls qui ne trouve pas un de ses arguments sort en statut 2 - le
          # shell des steps tourne avec -e, la derniere commande decide.
          ls -lh ./*.iso 2>/dev/null || true
          df -h /
        env:
          OSMO_PMOS_IMAGE_URL: ${{ github.event.inputs.pmos_image == 'true' && (vars.PMOS_IMAGE_URL ||
↪  'https://github.com/bbaranoff/osmo-operator/releases/download/pmos-image/qemu-amd64.img.zst') || '' }}
          # Le runner a deja docker, compose v2 et buildx (paquets moby-*), et
          # docker.io entre en conflit avec eux : apt repond « pkgProblemResolver
          # generated breaks » et build.sh s'arretait la. Rien a installer ici.
          OSMO_HOST_DOCKER: "0"

      # build.sh a tourne sous sudo : les .deb qu'il vient de deposer sont a
      # root, et le post-step d'actions/cache tourne en runner. Sans ce chown,
      # la sauvegarde du cache echoue en silence et le run suivant recompile.
      - name: Rendre le cache .deb au runner
        if: always()
        run: |
          CACHE="${OSMO_DEB_CACHE:-/var/cache/osmo-debs}"
          sudo chown -R "$(id -u):$(id -g)" "$CACHE" 2>/dev/null || true
          echo "$(find "$CACHE" -name 'osmo-build-*.deb' | wc -l) paquets, $(du -sh "$CACHE" | cut -f1)"

      # Apres le build, tue ou non : les dernieres mesures avant la fin.
      - name: Temoin - dernieres mesures
        if: always()
        run: tail -40 "$RUNNER_TEMP/watchdog.log" 2>/dev/null || echo "(pas de temoin)"

      - name: Checksums
        run: |
          # Aucune image n'est tenue pour acquise : avec --role=... une seule
          # est produite, et nommer les trois autres ferait echouer sha256sum.
          ISOS=""
          for iso in interstp.iso osmo-operator.iso \
                     osmo-operator-lite.iso osmo-operator-desktop.iso; do
              [ -f "$iso" ] && ISOS="$ISOS $iso"
          done
          [ -n "$ISOS" ] || { echo "::error::aucune ISO produite"; exit 1; }
          sha256sum $ISOS > SHA256SUMS
          md5sum    $ISOS > MD5SUMS
          cat SHA256SUMS

      # Une Release GitHub refuse tout fichier de plus de 2 Gio. On ne decoupe
      # QUE ce qui depasse : decouper une image de 1,3 Go obligerait a la
      # recombiner pour rien. Les sommes ci-dessus portent sur les ISO ENTIERES,
      # elles restent donc verifiables apres recombinaison.
      - name: Split ISOs over 2 GiB
        run: |
          LIMIT=$((2 * 1024 * 1024 * 1024))
          for iso in interstp.iso osmo-operator.iso osmo-operator-lite.iso osmo-operator-desktop.iso; do
            [ -f "$iso" ] || continue
            size=$(stat -c%s "$iso")
            if [ "$size" -gt "$LIMIT" ]; then
              echo "$iso : $size octets > 2 Gio -> decoupe"
              split -b 1900M -d -a 2 "$iso" "${iso}.part-"
              sha256sum "${iso}".part-* >> PARTS.sha256
              rm -f "$iso"          # seules les parties sont publiees
            else
              echo "$iso : $size octets -> publie tel quel"
            fi
          done
```

```
        touch PARTS.sha256
        ls -lh ./*.iso ./*.iso.part-* 2>/dev/null || true

    - name: Upload artifacts
      uses: actions/upload-artifact@v6
      with:
        name: osmo_egprs-iso
        path: |
          interstp.iso*
          osmo-operator.iso*
          osmo-operator-lite.iso*
          osmo-operator-desktop.iso*
          SHA256SUMS
          MD5SUMS
        retention-days: 14
        compression-level: 0

    - name: Create GitHub Release
      if: startsWith(github.ref, 'refs/tags/') || (github.event_name == 'workflow_dispatch' &&
↪ github.event.inputs.release == 'true')
      uses: softprops/action-gh-release@v3
      with:
        # [2026-09-14] PAS github.ref_name tout seul. Sur un workflow_dispatch
        # depuis la branche, ref_name vaut « main » : l'action creait donc un
        # TAG nomme « main », a cote de la branche du meme nom. D'ou le
        # « warning: refname 'main' is ambiguous » que git sort ensuite a
        # chaque commande - il ne sait plus si « main » designe la branche ou
        # le tag - et un release qui ne disait pas quelle version il portait.
        # Un push de tag garde son propre tag ; tout le reste sort sous la
        # version saisie au lancement (entree version, defaut ci-dessous).
        tag_name: ${{ startsWith(github.ref, 'refs/tags/') && github.ref_name || github.event.inputs.version ||
↪ 'v0.1-9' }}
        # [2026-09-25] Le nom et le texte sont ceux des releases publiees a la
        # main (« Osmo-Operator v0.1-3 ») : le texte vit dans
        # .github/release-notes.md, qui se modifie sans toucher au workflow.
        name: "Osmo-Operator ${{ startsWith(github.ref, 'refs/tags/') && github.ref_name ||
↪ github.event.inputs.version || 'v0.1-9' }}"
        # [2026-10-09] Brouillon le temps des envois : sans cela la release
        # devient publique avec quelques fichiers seulement (un timeout d'upload
        # laissait ainsi v0.1-7 a moitie complete). Publiee par l'etape suivante.
        draft: true
        prerelease: ${{ !(startsWith(github.ref, 'refs/tags/v') || startsWith(github.event.inputs.version, 'v'))
↪ }}
        body_path: .github/release-notes.md

    # [2026-10-09] softprops envoyait tous les fichiers en parallele ; les
    # parties de 1 a 1,7 Go coupaient avec « Upload body timed out due to
    # inactivity ». Ici : un fichier a la fois, relances sur echec.
    - name: Upload release assets (one by one, retried)
      if: startsWith(github.ref, 'refs/tags/') || (github.event_name == 'workflow_dispatch' &&
↪ github.event.inputs.release == 'true')
      env:
        GH_TOKEN: ${{ secrets.GITHUB_TOKEN }}
        TAG: ${{ startsWith(github.ref, 'refs/tags/') && github.ref_name || github.event.inputs.version ||
↪ 'v0.1-9' }}
      run: |
        set -u
        shopt -s nullglob
        files=(interstp.iso* osmo-operator.iso* osmo-operator-lite.iso* osmo-operator-desktop.iso* SHA256SUMS
↪ MD5SUMS PARTS.sha256)
        for f in "${files[@]}"; do
          [ -f "$f" ] || { echo "absent: $f"; continue; }
          ok=0
          for attempt in 1 2 3 4 5; do
            echo "== $f (tentative $attempt)"
            if gh release upload "$TAG" "$f" --clobber; then ok=1; break; fi
```

```
            sleep $((attempt * 20))
          done
          [ "$ok" = 1 ] || { echo "ECHEC definitif : $f"; exit 1; }
        done

    - name: Publish release
      if: startsWith(github.ref, 'refs/tags/') || (github.event_name == 'workflow_dispatch' &&
↪  github.event.inputs.release == 'true')
      env:
        GH_TOKEN: ${{ secrets.GITHUB_TOKEN }}
        TAG: ${{ startsWith(github.ref, 'refs/tags/') && github.ref_name || github.event.inputs.version ||
↪  'v0.1-9' }}
        PRE: ${{ !(startsWith(github.ref, 'refs/tags/v') || startsWith(github.event.inputs.version, 'v')) }}
      run: |
        gh release edit "$TAG" --draft=false --prerelease="$PRE"
```

3.31 /opt/GSM/osmo-operator/.github/workflows/docker-arm.yml

7661 octets, 191 lignes → 191 lignes

```
name: Docker arm64 (GHCR)

# L'image de BASE (Dockerfile -> osmocom-nitb) construite NATIVEMENT en aarch64
# et publiee sur GHCR sous osmocom-nitb:base-arm64-<empreinte>. C'est le pendant
# arm64 du job `base` de docker.yml (qui ne fait que de l'amd64).
#
# Pourquoi un workflow a part, en amont de build-arm.yml : jusqu'ici l'image SD
# compilait toute la pile DANS son propre job (« rien n'est publie en arm64 »,
# donc pas de --skip-build) - c'etait l'essentiel des 6 h. Ici la compilation
# sort dans un job dedie, avec ses caches (couches via :cache-arm64, .deb via
# actions/cache), et build-arm.yml n'a plus qu'a TIRER l'image. Une base
# inchangee (meme empreinte) n'est pas reconstruite.
#
# L'empreinte est la MEME commande que docker.yml / build-iso.yml : memes
# fichiers, donc memes contenus. Le prefixe `base-arm64-` evite qu'elle
# collisionne avec la base amd64 du meme commit.

on:
  workflow_call:
    inputs:
      rebuild:
        description: "Reconstruire meme si l'empreinte existe deja"
        type: boolean
        default: false
      no_cache:
        description: "Sans cache de couches (--no-cache)"
        type: boolean
        default: false
    outputs:
      image:
        description: "Reference GHCR de la base arm64"
        value: ${{ jobs.base-arm64.outputs.image }}
  workflow_dispatch:
    inputs:
      rebuild:
        description: "Reconstruire meme si l'empreinte existe deja"
        type: boolean
        default: false
      no_cache:
        description: "Sans cache de couches (--no-cache)"
        type: boolean
        default: false

env:
```

```yaml
  REGISTRY: ghcr.io

jobs:
  base-arm64:
    runs-on: ubuntu-24.04-arm
    timeout-minutes: 360
    permissions:
      contents: read
      packages: write
    outputs:
      image: ${{ steps.key.outputs.image }}
      hit:   ${{ steps.probe.outputs.hit }}
    steps:
      - name: Checkout
        uses: actions/checkout@v5

      - name: Empreinte du contexte
        id: key
        run: |
          REPO=$(echo '${{ github.repository }}' | tr '[:upper:]' '[:lower:]')
          KEY=$(git ls-files -- Dockerfile packaging configs scripts services \
                               helpers opt-gsm tools patches \
                | sort | xargs sha256sum | sha256sum | cut -c1-16)
          echo "key=$KEY" >> "$GITHUB_OUTPUT"
          echo "image=${REGISTRY}/${REPO}/osmocom-nitb:base-arm64-${KEY}" >> "$GITHUB_OUTPUT"
          echo "cache=${REGISTRY}/${REPO}/osmocom-nitb:cache-arm64"       >> "$GITHUB_OUTPUT"
          echo "Empreinte du contexte : $KEY"

      - name: Log in to GHCR
        uses: docker/login-action@v4
        with:
          registry: ${{ env.REGISTRY }}
          username: ${{ github.actor }}
          password: ${{ secrets.GITHUB_TOKEN }}

      - name: La base arm64 existe-t-elle deja ?
        id: probe
        run: |
          HIT=false
          if [ "${{ inputs.rebuild }}" != "true" ] \
             && docker manifest inspect '${{ steps.key.outputs.image }}' >/dev/null 2>&1; then
            HIT=true
          fi
          echo "hit=$HIT" >> "$GITHUB_OUTPUT"
          if [ "$HIT" = true ]; then
            echo "::notice::Base arm64 inchangee, deja publiee : ${{ steps.key.outputs.image }}"
          else
            echo "::notice::Base arm64 absente ou forcee : construction native."
          fi

      - name: Disque pour docker
        if: steps.probe.outputs.hit != 'true'
        uses: ./.github/actions/docker-disk
        with:
          need: "40"

      - name: Buildx
        if: steps.probe.outputs.hit != 'true'
        uses: docker/setup-buildx-action@v4

      - name: Preparer le cache .deb
        if: steps.probe.outputs.hit != 'true'
        run: |
          sudo mkdir -p /var/cache/osmo-debs
          sudo chown -R "$(id -u):$(id -g)" /var/cache/osmo-debs
```

```
    # Cle distincte de celle de build-arm.yml (osmo-debs-noble-arm64-) : meme
    # contenu, mais ce job est desormais celui qui l'alimente.
    - name: Cache .deb (osmo-deb, arm64)
      if: steps.probe.outputs.hit != 'true'
      uses: actions/cache@v5
      with:
        path: /var/cache/osmo-debs
        key: osmo-debs-noble-arm64-${{ steps.key.outputs.key }}
        restore-keys: |
          osmo-debs-noble-arm64-

    # Meme filet anti-OOM que build-arm.yml : buildkit batit plusieurs stages
    # a la fois, le pic cumule depasse la RAM du runner arm64.
    - name: Swap anti-OOM
      if: steps.probe.outputs.hit != 'true'
      run: |
        pick() { df -P "$1" 2>/dev/null | awk 'NR==2{print $4}'; }
        AV_ROOT=$(pick /); AV_MNT=$(pick /mnt)
        BASE=/; [ -n "$AV_MNT" ] && [ "${AV_MNT:-0}" -gt "${AV_ROOT:-0}" ] && BASE=/mnt
        FREE_GB=$(( $(pick "$BASE") / 1024 / 1024 ))
        SWAP_GB=12; THIRD=$(( FREE_GB / 3 ))
        [ "$SWAP_GB" -gt "$THIRD" ] && SWAP_GB=$THIRD
        [ "$SWAP_GB" -lt 2 ] && SWAP_GB=2
        SF="$BASE/osmo-swapfile"
        sudo swapoff "$SF" 2>/dev/null || true; sudo rm -f "$SF"
        sudo fallocate -l ${SWAP_GB}G "$SF" 2>/dev/null \
          || sudo dd if=/dev/zero of="$SF" bs=1M count=$((SWAP_GB*1024)) status=none
        sudo chmod 600 "$SF"; sudo mkswap "$SF" >/dev/null; sudo swapon "$SF"
        echo "swap +${SWAP_GB} Go sur $BASE"

    - name: Temoin disque/memoire
      if: steps.probe.outputs.hit != 'true'
      run: |
        nohup bash -c 'while true; do
          echo "$(date -u +%H:%M:%S)  / $(df -h --output=avail / | tail -1 | tr -d " ")  /mnt $(df -h
↪ --output=avail /mnt 2>/dev/null | tail -1 | tr -d " ")  mem $(free -m | awk "/^Mem:/{print \$7}")Mo"
          sleep 30
        done' > "$RUNNER_TEMP/watchdog.log" 2>&1 &

    # Hote aarch64 : buildx construit pour la plateforme native, sans
    # --platform ni qemu. --push comme docker.yml : rien dans le store local.
    - name: Build et push osmocom-nitb (base arm64)
      if: steps.probe.outputs.hit != 'true'
      env:
        IMAGE: ${{ steps.key.outputs.image }}
        CACHE: ${{ steps.key.outputs.cache }}
        NO_CACHE: ${{ inputs.no_cache }}
      run: |
        mkdir -p .deb-cache
        cp -f /var/cache/osmo-debs/osmo-build-*_arm64.deb .deb-cache/ 2>/dev/null || true
        echo "cache .deb en entree : $(find .deb-cache -name 'osmo-build-*.deb' | wc -l) paquets"

        CACHE_ARGS=(--cache-from "type=registry,ref=$CACHE")
        REFRESH=0
        if [ "$NO_CACHE" = true ]; then CACHE_ARGS=(--no-cache); REFRESH=1; fi

        docker buildx build \
          --platform linux/arm64 \
          --target osmocom-nitb \
          --build-arg "OSMO_DEB_REFRESH=$REFRESH" \
          "${CACHE_ARGS[@]}" \
          --cache-to type=inline \
          -t "$IMAGE" -t "$CACHE" --push \
          -f Dockerfile .

        # Recolte des .deb : seconde passe sur `debs`, tout en cache.
```

```
          rm -rf /tmp/debs-out && mkdir -p /tmp/debs-out
          docker buildx build \
            --platform linux/arm64 \
            --target debs \
            --build-arg "OSMO_DEB_REFRESH=$REFRESH" \
            --cache-from "type=registry,ref=$CACHE" \
            --output "type=local,dest=/tmp/debs-out" \
            -f Dockerfile .
          cp -f /tmp/debs-out/osmo-build-*.deb /var/cache/osmo-debs/ 2>/dev/null || true
          rm -rf /tmp/debs-out .deb-cache/*.deb
          echo "cache .deb en sortie : $(find /var/cache/osmo-debs -name 'osmo-build-*.deb' | wc -l) paquets"
          df -h / /mnt || true

      - name: Temoin - dernieres mesures
        if: always()
        run: tail -40 "$RUNNER_TEMP/watchdog.log" 2>/dev/null || echo "(pas de temoin)"
```

3.32 /opt/GSM/osmo-operator/.github/workflows/docker.yml

19026 octets, 411 lignes → 411 lignes

```
name: Docker (GHCR)

# Quatre images publiees :
#
#   osmocom-nitb        l'image de BASE (Dockerfile) : toute la compilation,
#                       ~11 Go. Elle n'est PLUS rebatie a chaque push - voir le
#                       job `base` plus bas.
#   interstp            le hub SS7 seul (Dockerfile.stp) - osmo-stp et sa table
#                       de routage, rien d'autre. Independante : elle ne part
#                       pas de osmocom-nitb, donc elle sort en minutes.
#   osmo-operator       un noeud complet (Dockerfile.run, qui part de
#                       osmocom-nitb) : HLR, MSC, BSC, PCU, BTS, Asterisk, QEMU.
#   osmo-operator-lite  la meme pile, sans les arbres de sources de /opt/GSM
#                       (Dockerfile.lite) - ~4 Go de moins sur 11.
#
# ── Pourquoi cette structure ──────────────────────────────────────────
# [2026-09-13] Avant, un seul job enchainait les trois builds sur un runner, et
# recompilait donc l'image de base a chaque push : 1h23, plus un disque sature
# en « exporting layers » (exit 143, un SIGTERM du runner a bout d'espace).
# Maintenant :
#   1. `base` calcule une EMPREINTE de tout ce que le Dockerfile prend dans le
#      contexte. Si ghcr.io/<depot>/osmocom-nitb:base-<empreinte> existe deja,
#      il ne construit RIEN et se contente de la publier comme reference.
#   2. Sinon il construit, avec deux caches : les couches de la derniere base
#      publiee (--cache-from registry + --cache-to inline) et les .deb deja
#      compiles (actions/cache sur /var/cache/osmo-debs, le mecanisme de
#      packaging/osmo-deb.sh). Un changement en fin de Dockerfile ne recompile
#      plus que sa queue.
#   3. `operator` et `lite` partent de cette image, en parallele, chacun sur
#      son runner - donc chacun avec un disque propre.
#
# ── Et pourquoi buildx --push partout ─────────────────────────────────
# [2026-09-13] Les trois jobs poussent leur image SANS jamais la charger dans
# le docker du runner. `docker build -t X . && docker push X` demande a
# buildkit d'exporter l'image dans le store local : il detient alors ses
# snapshots ET ecrit par-dessus les archives de couches, soit deux fois 11 Go.
# C'est la, en « exporting layers » - build reussi, 75 etapes sur 75 - que le
# runner saturait et se faisait tuer (exit 143). Avec
# `docker buildx build --push`, les couches sont compressees et poussees au
# fil de l'eau ; il ne reste sur le disque que le cache du builder.
#
# Consequence a ne pas perdre de vue : plus aucune image locale apres le
# build, donc plus de `docker create` + `docker cp` pour recuperer les .deb
```

```
# compiles. C'est le role de l'etape `debs` du Dockerfile (une seconde passe
# `--target debs --output type=local`, tout en cache). Et parce que `debs` est
# desormais la DERNIERE etape du Dockerfile, tout build de la base doit dire
# `--target osmocom-nitb` : ici, dans build.sh, dans compose.yaml et dans
# iso_modules/21-docker-build.sh.
#
# Dockerfile.run et Dockerfile.lite ont un `ARG BASE` : en local ils valent
# toujours `osmocom-nitb` (l'image du docker de l'hote), en CI ils recoivent la
# reference GHCR. build.sh n'est pas concerne.
#
# Sur une pull_request, rien n'est publie (le GITHUB_TOKEN d'un fork n'a pas
# packages:write) : `base` construit pour valider le Dockerfile, et les deux
# jobs derives ne tournent pas.

on:
  push:
    branches: ["main"]
    tags: ["v*.*.*"]
  pull_request:
    branches: ["main"]
  workflow_dispatch:
    inputs:
      rebuild_base:
        description: "Reconstruire l'image de base meme si l'empreinte existe deja"
        type: boolean
        default: false

env:
  REGISTRY: ghcr.io

jobs:
  # ── L'image de base : construite seulement si son contexte a bouge ──────────
  base:
    runs-on: ubuntu-24.04
    timeout-minutes: 360
    permissions:
      contents: read
      packages: write
      id-token: write
    outputs:
      repo:  ${{ steps.img.outputs.repo }}
      image: ${{ steps.key.outputs.image }}
      hit:   ${{ steps.probe.outputs.hit }}
    steps:
      - name: Checkout
        uses: actions/checkout@v5

      # GHCR n'accepte que des noms d'image en minuscules ; github.repository
      # garde la casse du proprietaire.
      - name: Image name
        id: img
        run: echo "repo=$(echo '${{ github.repository }}' | tr '[:upper:]' '[:lower:]')" >> "$GITHUB_OUTPUT"

      # L'EMPREINTE. Tout ce que Dockerfile prend dans le contexte de build -
      # lui-meme, packaging/, patches/, et les dossiers qu'il COPY (configs,
      # scripts, services, helpers, opt-gsm, tools). Pas .deb-cache/ : c'est un
      # cache, son contenu ne definit pas l'image.
      # `git ls-files` et non `find` : ce qui n'est pas versionne n'entre pas
      # dans l'image de toute facon, et l'ordre est stable.
      - name: Empreinte du contexte de base
        id: key
        run: |
          KEY=$(git ls-files -- Dockerfile packaging configs scripts services \
                                helpers opt-gsm tools patches \
                | sort | xargs sha256sum | sha256sum | cut -c1-16)
          REPO='${{ steps.img.outputs.repo }}'
```

```
    echo "key=$KEY"                                              >> "$GITHUB_OUTPUT"
    echo "image=${REGISTRY}/${REPO}/osmocom-nitb:base-${KEY}"    >> "$GITHUB_OUTPUT"
    echo "cache=${REGISTRY}/${REPO}/osmocom-nitb:cache"          >> "$GITHUB_OUTPUT"
    echo "Empreinte du contexte : $KEY"

- name: Log in to GHCR
  if: github.event_name != 'pull_request'
  uses: docker/login-action@v4
  with:
    registry: ${{ env.REGISTRY }}
    username: ${{ github.actor }}
    password: ${{ secrets.GITHUB_TOKEN }}

# Le manifeste suffit : pas besoin de tirer 11 Go pour savoir si l'image
# est la. Un `rebuild_base` coche force la reconstruction.
- name: L'image de base existe-t-elle deja ?
  id: probe
  run: |
    HIT=false
    if [ "${{ github.event.inputs.rebuild_base }}" != "true" ] \
       && [ "${{ github.event_name }}" != "pull_request" ] \
       && docker manifest inspect '${{ steps.key.outputs.image }}' >/dev/null 2>&1; then
      HIT=true
    fi
    echo "hit=$HIT" >> "$GITHUB_OUTPUT"
    if [ "$HIT" = true ]; then
      echo "::notice::Base inchangee, deja publiee : ${{ steps.key.outputs.image }} - rien a construire."
    else
      echo "::notice::Base absente ou forcee : construction."
    fi

# ── A partir d'ici, uniquement quand il faut vraiment construire ────────

# Menage du runner + /var/lib/docker sur /mnt, avec verification et
# echec immediat s'il n'y a la place nulle part. Trois jobs en ont
# besoin, d'ou l'action locale plutot qu'un bloc recopie trois fois.
- name: Disque pour docker
  if: steps.probe.outputs.hit != 'true'
  uses: ./.github/actions/docker-disk

# buildx/docker-container : le driver qui sait ecrire AILLEURS que dans
# le store d'images local - voir le step de build. Apres le deplacement
# du data-root, jamais avant : le redemarrage du demon laisserait le
# conteneur du builder orphelin.
- name: Buildx
  if: steps.probe.outputs.hit != 'true'
  uses: docker/setup-buildx-action@v4

# ── Le cache .deb entre deux runs ──────────────────────────────────────
# Ce que le Dockerfile compile sort en paquet .deb dans
# /var/cache/osmo-debs (packaging/osmo-deb.sh), et il le relit par
# `COPY .deb-cache/ /var/cache/osmo-debs/`. Sur l'hote, build.sh fait la
# navette entre les deux ; ici on n'appelle pas build.sh, donc on la fait
# a la main dans le step de build. Les .deb sont indexes par nom et
# version : contrairement au cache de couches, ce cache-la survit a une
# modification du Dockerfile.
# OSMO_DEB_REFRESH n'est pas passe : son defaut (Dockerfile:166) est 0,
# donc le cache est utilise. Ne pas y toucher ici.
- name: Preparer le cache .deb
  if: steps.probe.outputs.hit != 'true'
  run: |
    sudo mkdir -p /var/cache/osmo-debs
    sudo chown -R "$(id -u):$(id -g)" /var/cache/osmo-debs

- name: Cache .deb (osmo-deb)
  if: steps.probe.outputs.hit != 'true'
```

```
      uses: actions/cache@v5
      with:
        path: /var/cache/osmo-debs
        # CLE = L'EMPREINTE DU CONTEXTE, pas le numero du run.
        # [2026-09-13] C'etait `${{ github.run_id }}`, donc une entree NEUVE a
        # chaque passage. A 2-4 Go l'entree et 10 Go de quota par depot,
        # l'eviction LRU detruisait le cache au bout de trois runs - et tous
        # les autres caches du depot avec. Avec l'empreinte : contexte
        # inchange, la cle est deja la, rien n'est reecrit ; contexte change,
        # une seule entree s'ajoute. Le prefixe reprend la plus recente.
        key: osmo-debs-noble-${{ steps.key.outputs.key }}
        restore-keys: |
          osmo-debs-noble-

    - name: Temoin disque
      if: steps.probe.outputs.hit != 'true'
      run: |
        nohup bash -c 'while true; do
          echo "$(date -u +%H:%M:%S)  / $(df -h --output=avail / | tail -1 | tr -d " ")  /mnt $(df -h
↪ --output=avail /mnt | tail -1 | tr -d " ")  mem $(free -m | awk "/^Mem:/{print \$7}")Mo"
          sleep 30
        done' > "$RUNNER_TEMP/watchdog.log" 2>&1 &
        echo "temoin -> $RUNNER_TEMP/watchdog.log"

    # —— LE BUILD. Vers GHCR directement, jamais dans le docker local ————
    # [2026-09-13] C'etait `docker build -t ... .` puis `docker push`. Le
    # `-t` impose a buildkit d'EXPORTER l'image dans le store local : a ce
    # moment il detient ses snapshots (11 Go) et ecrit par-dessus les
    # archives de couches (11 Go de plus). C'est exactement la, en
    # « exporting layers », que le runner saturait et se faisait tuer -
    # exit 143, apres 1h20 de compilation reussie.
    #
    # `--push` (soit --output type=image,push=true) supprime cette moitie-la
    # du probleme : les couches sont compressees et poussees au fil de l'eau,
    # rien n'atterrit dans le store local. Il ne reste que le cache du
    # builder. Corollaire : plus aucune image `osmocom-nitb` sur le runner
    # apres le build - d'ou l'etape `debs` du Dockerfile pour la recolte.
    - name: Build et push osmocom-nitb (base)
      if: steps.probe.outputs.hit != 'true'
      env:
        IMAGE: ${{ steps.key.outputs.image }}
        CACHE: ${{ steps.key.outputs.cache }}
        PR: ${{ github.event_name == 'pull_request' }}
      run: |
        # Semer le contexte avec les .deb du cache : chaque etape qui les
        # trouve pose le paquet au lieu de cloner et compiler.
        mkdir -p .deb-cache
        cp -f /var/cache/osmo-debs/osmo-build-*.deb .deb-cache/ 2>/dev/null || true
        echo "cache .deb en entree : $(find .deb-cache -name 'osmo-build-*.deb' | wc -l) paquets"

        # base-<empreinte> est ce que les jobs derives tirent ; :cache et
        # :latest sont les deux commodites - l'un pour le --cache-from du
        # prochain build, l'autre pour qui veut « la derniere ». Les trois
        # noms sortent d'UN push : memes digests, les couches ne montent
        # qu'une fois.
        # Sur une pull_request rien n'est publie (le GITHUB_TOKEN d'un fork
        # n'a pas packages:write) : on batit pour valider le Dockerfile et on
        # garde le resultat dans le cache du builder, sans export ni nom.
        if [ "$PR" = true ]; then
          TAGS=(); OUT=(--output type=cacheonly)
        else
          BASE_REPO="${CACHE%:cache}"
          TAGS=(-t "$IMAGE" -t "$CACHE")
          [ "$GITHUB_REF_NAME" = "main" ] && TAGS+=(-t "$BASE_REPO:latest")
          OUT=(--push)
        fi
```

```
        # --target : Dockerfile finit par l'etape `debs`, pas par osmocom-nitb.
        # --cache-to inline : les metadonnees de cache voyagent DANS le
        # manifeste pousse, ce qui rend le --cache-from du prochain run
        # possible sans second depot de cache a gerer.
        docker buildx build \
          --target osmocom-nitb \
          --cache-from "type=registry,ref=$CACHE" \
          --cache-to   type=inline \
          "${TAGS[@]+"${TAGS[@]}"}" "${OUT[@]}" \
          -f Dockerfile .

        # Recolte des .deb : seconde passe sur l'etape `debs`. Le CONTEXTE
        # N'A PAS BOUGE (on ne vide .deb-cache/ qu'apres) donc les 75 etapes
        # sont toutes en cache dans le builder ; seul le COPY final tourne,
        # et n'ecrit sur le disque que les paquets. Equivalent sans image
        # locale de debs_from_image() (build.sh:306).
        rm -rf /tmp/debs-out && mkdir -p /tmp/debs-out
        docker buildx build \
          --target debs \
          --cache-from "type=registry,ref=$CACHE" \
          --output "type=local,dest=/tmp/debs-out" \
          -f Dockerfile .
        cp -f /tmp/debs-out/osmo-build-*.deb /var/cache/osmo-debs/ 2>/dev/null || true
        rm -rf /tmp/debs-out .deb-cache/*.deb
        echo "cache .deb en sortie : $(find /var/cache/osmo-debs -name 'osmo-build-*.deb' | wc -l) paquets, $(du
↪  -sh /var/cache/osmo-debs | cut -f1)"
        df -h / /mnt

    - name: Temoin - dernieres mesures
      if: always()
      run: tail -40 "$RUNNER_TEMP/watchdog.log" 2>/dev/null || echo "(pas de temoin)"

# ── Le hub : rapide, independant, donc en parallele du reste ──────────────
interstp:
  runs-on: ubuntu-24.04
  timeout-minutes: 120
  permissions:
    contents: read
    packages: write
    id-token: write
  steps:
    - name: Checkout
      uses: actions/checkout@v5

    - name: Image name
      id: img
      run: echo "repo=$(echo '${{ github.repository }}' | tr '[:upper:]' '[:lower:]')" >> "$GITHUB_OUTPUT"

    - name: Log in to GHCR
      if: github.event_name != 'pull_request'
      uses: docker/login-action@v4
      with:
        registry: ${{ env.REGISTRY }}
        username: ${{ github.actor }}
        password: ${{ secrets.GITHUB_TOKEN }}

    - name: Build interstp
      run: |
        docker build -f Dockerfile.stp -t interstp:ci .
        docker images interstp:ci --format '  {{.Repository}}:{{.Tag}} {{.Size}}'

    - name: Push interstp
      if: github.event_name != 'pull_request'
      env:
        IMG: ${{ env.REGISTRY }}/${{ steps.img.outputs.repo }}/interstp
```

```
      run: |
        TAGS="sha-${GITHUB_SHA::7}"
        [ "${GITHUB_REF_TYPE}" = "tag" ]    && TAGS="$TAGS ${GITHUB_REF_NAME}"
        [ "${GITHUB_REF_NAME}" = "main" ]   && TAGS="$TAGS latest"
        for t in $TAGS; do
          docker tag interstp:ci "$IMG:$t"
          docker push "$IMG:$t"
        done

# ── Les deux derivees : en parallele, chacune sur un runner neuf ──────────
# Quelques minutes chacune - elles ne compilent rien, elles rafraichissent des
# COPY par-dessus une base deja construite.
operator:
  needs: base
  if: github.event_name != 'pull_request'
  runs-on: ubuntu-24.04
  timeout-minutes: 90
  permissions:
    contents: read
    packages: write
    id-token: write
  steps:
    - name: Checkout
      uses: actions/checkout@v5

    - name: Disque pour docker
      uses: ./.github/actions/docker-disk

    - name: Buildx
      uses: docker/setup-buildx-action@v4

    - name: Log in to GHCR
      uses: docker/login-action@v4
      with:
        registry: ${{ env.REGISTRY }}
        username: ${{ github.actor }}
        password: ${{ secrets.GITHUB_TOKEN }}

    # Meme principe que la base : buildx pousse les couches vers GHCR sans
    # jamais materialiser l'image dans le store local. Le `docker pull` du
    # BASE disparait avec, buildkit ne tire que ce dont le build a besoin ;
    # et les couches heritees de la base sont MONTEES d'un depot a l'autre
    # (cross-repo blob mount, meme registre), donc la poussee ne renvoie que
    # les couches propres a Dockerfile.run.
    - name: Build et push osmo-operator
      env:
        BASE_IMAGE: ${{ needs.base.outputs.image }}
        IMG: ${{ env.REGISTRY }}/${{ needs.base.outputs.repo }}/osmo-operator
      run: |
        TAGS=(-t "$IMG:sha-${GITHUB_SHA::7}")
        [ "${GITHUB_REF_TYPE}" = "tag" ]  && TAGS+=(-t "$IMG:${GITHUB_REF_NAME}")
        [ "${GITHUB_REF_NAME}" = "main" ] && TAGS+=(-t "$IMG:latest")
        docker buildx build -f Dockerfile.run \
          --build-arg BASE="$BASE_IMAGE" \
          "${TAGS[@]}" --push .
        docker buildx imagetools inspect "$IMG:sha-${GITHUB_SHA::7}" \
          --format '{{.Manifest.Config.Digest}}  {{len .Manifest.Layers}} couches' || true
        df -h / /mnt

# L'image lite se deduit de la BASE, pas de osmo-operator : Dockerfile.lite
# elague /opt/GSM puis aplatit le resultat (FROM scratch + COPY / - effacer
# dans une couche de plus ne rend rien). C'est ce que fait build.sh:339.
lite:
  needs: base
  if: github.event_name != 'pull_request'
  runs-on: ubuntu-24.04
```

```yaml
    timeout-minutes: 90
    permissions:
      contents: read
      packages: write
      id-token: write
    steps:
      - name: Checkout
        uses: actions/checkout@v5

      - name: Disque pour docker
        uses: ./.github/actions/docker-disk

      - name: Buildx
        uses: docker/setup-buildx-action@v4

      - name: Log in to GHCR
        uses: docker/login-action@v4
        with:
          registry: ${{ env.REGISTRY }}
          username: ${{ github.actor }}
          password: ${{ secrets.GITHUB_TOKEN }}

      # L'aplatissement (FROM scratch + COPY /) recopie la racine entiere :
      # c'est le job qui a le plus a ecrire, pour un job qui ne compile rien.
      # Raison de plus pour que la sortie parte au registre et non dans le
      # store local.
      - name: Build et push osmo-operator-lite
        env:
          BASE_IMAGE: ${{ needs.base.outputs.image }}
          IMG: ${{ env.REGISTRY }}/${{ needs.base.outputs.repo }}/osmo-operator-lite
        run: |
          TAGS=(-t "$IMG:sha-${GITHUB_SHA::7}")
          [ "${GITHUB_REF_TYPE}" = "tag" ]  && TAGS+=(-t "$IMG:${GITHUB_REF_NAME}")
          [ "${GITHUB_REF_NAME}" = "main" ] && TAGS+=(-t "$IMG:latest")
          docker buildx build -f Dockerfile.lite \
            --build-arg BASE="$BASE_IMAGE" \
            "${TAGS[@]}" --push .
          echo "--- taille publiee ---"
          docker buildx imagetools inspect "$IMG:sha-${GITHUB_SHA::7}" || true
          df -h / /mnt
```

3.33 /opt/GSM/osmo-operator/armbian/build-armbian.sh

17017 octets, 272 lignes → 272 lignes

```
#!/bin/bash
# armbian/build-armbian.sh - l image SD arm64 par le framework OFFICIEL github.com/armbian/build
#
# [2026-10-20] Jusqu ici l image Pi sortait de build-iso.sh --arm : debootstrap, noyau et BSP rpi4b
# pris sur apt.armbian.com, partition firmware et .img assembles a la main (iso_modules/82-arm-natif,
# 92-rpi-image). Ici c est Armbian lui-meme qui fabrique le noyau, le BSP, le u-boot/firmware, le
# rootfs et l image ; nous ne faisons qu y POSER LA PILE, par le mecanisme prevu pour cela :
#     userpatches/customize-image.sh   execute dans le chroot de l image (sous qemu-user sur un hote x86)
#     userpatches/overlay/             monte en /tmp/overlay dans ce chroot
#
# La pile vient de l IMAGE DOCKER arm64 du projet (Dockerfile, build.sh --arch=arm64 : osmocom-nitb:arm64,
# ou la base publiee ghcr.io/<depot>/osmocom-nitb:base-arm64-<empreinte>) : exactement ce que
# iso_modules/50-injection-image.sh prend pour l ISO - les .deb du build (osmo-build-*), /usr/local,
# /opt/GSM, /etc/osmocom, les unites systemd. Elle est exportee dans l overlay par docker cp.
# Ce qu un hote x86 ne peut pas produire (toast, lanceur qosmo, c54x_exe/grgsm_exe, .deb du banc) est
# compile DANS le chroot par customize-image.sh, comme le faisait 82-arm-natif.sh.
#
#   sudo ./build-armbian.sh                      rpi4b, branche current, noble : clone, overlay, compile
#   sudo ./build-armbian.sh --prepare            tout sauf compile.sh (verifier userpatches/ et l overlay)
```

```
#   sudo ./build-armbian.sh --no-stack            une Armbian nue (pas d overlay, pas de customize)
#   sudo ./build-armbian.sh --board=orangepi5 --branch=edge --release=noble
#   sudo ./build-armbian.sh --image=osmocom-nitb:arm64      l image Docker qui fournit la pile
#
# [2026-10-24] LES CIBLES : --board= prend une carte, une liste (virgules) ou un groupe :
#   rpi4                  rpi4b (Raspberry Pi 4)
#   opi5                  TOUTES les Orange Pi 5 d Armbian : orangepi5, orangepi5b, orangepi5-plus,
#                         orangepi5pro, orangepi5-max, orangepi5-ultra (la Max et l Ultra ont chacune
#                         leur fichier de carte : ce ne sont pas des variantes de orangepi5.conf)
#   all                   rpi4 + opi5
# Une image par carte, construites l une apres l autre (meme overlay). La branche noyau est choisie
# PAR CARTE : --branch si la carte la propose (KERNEL_TARGET de config/boards/<carte>.*), sinon
# current, sinon la premiere qu elle propose (orangepi5b n a que edge et vendor). Les cartes .csc
# (communautaires : 5b, 5 Pro, 5 Max, 5 Ultra) se construisent avec EXPERT=yes.
#   --desktop             bureau GNOME (DESKTOP_ENVIRONMENT=gnome, DESKTOP_TIER=minimal, via
#                         armbian/configng) + le bureau du banc (fond, Conky, encart, icones, c123)
#   --banc / --multi      osmo-banc.service / osmo-multi.service actives au boot
#   --build-image         sans image Docker arm64 de la pile, la construire (build.sh --arch=arm64)
#                         au lieu d echouer
#   --armbian-dir=/chemin   le clone d armbian/build         [/opt/GSM/armbian-build]
#   --update                git pull du clone avant de construire
#   --install-deps          apt install qemu-user-static binfmt-support si l aarch64 n est pas emulable
#   --dry-run               affiche ce qui serait fait
#
# Prerequis hote : docker (compile.sh se met lui-meme dans un conteneur sur un hote non Ubuntu/x86),
# git, et l emulation aarch64 (qemu-user-static + binfmt) pour que le chroot de customize-image.sh
# tourne sur un hote x86. Sur un hote arm64 rien n est emule.
set -euo pipefail
HERE="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
REPO="$(cd "$HERE/.." && pwd)"

BOARD=rpi4b; BRANCH=current; RELEASE=noble
ARMBIAN_DIR="${ARMBIAN_DIR:-/opt/GSM/armbian-build}"
ARMBIAN_GIT="${ARMBIAN_GIT:-https://github.com/armbian/build}"
IMAGE=""; PREPARE=0; NO_STACK=0; UPDATE=0; INSTALL_DEPS=0; DRY=0
DESKTOP=0; BANC=0; MULTI=0; BUILD_IMAGE=0; BRANCH_GIVEN=0
DEB_CACHE="${OSMO_DEB_CACHE:-/var/cache/osmo-debs}"
for arg in "$@"; do
    case "$arg" in
        --board=*)       BOARD="${arg#*=}" ;;
        --branch=*)      BRANCH="${arg#*=}"; BRANCH_GIVEN=1 ;;
        --desktop)       DESKTOP=1 ;;
        --banc)          BANC=1 ;;
        --multi)         MULTI=1 ;;
        --build-image)   BUILD_IMAGE=1 ;;
        --release=*)     RELEASE="${arg#*=}" ;;
        --image=*)       IMAGE="${arg#*=}" ;;
        --armbian-dir=*) ARMBIAN_DIR="${arg#*=}" ;;
        --prepare)       PREPARE=1 ;;
        --no-stack)      NO_STACK=1 ;;
        --update)        UPDATE=1 ;;
        --install-deps)  INSTALL_DEPS=1 ;;
        --dry-run)       DRY=1 ;;
        -h|--help)       sed -n '2,46p' "$0"; exit 0 ;;
        *) echo "option inconnue : $arg" >&2; sed -n '2,46p' "$0" >&2; exit 2 ;;
    esac
done
say()  { printf '\033[1;32m[armbian]\033[0m %s\n' "$*"; }
warn() { printf '\033[1;33m[armbian] !\033[0m %s\n' "$*" >&2; }
die()  { printf '\033[1;31m[armbian] ECHEC\033[0m %s\n' "$*" >&2; exit 1; }
run()  { if [ "$DRY" = 1 ]; then echo "  + $*"; else "$@"; fi; }

# -- 1. l hote ----------------------------------------------------------------
[ "$(id -u)" = 0 ] || [ "$DRY" = 1 ] || die "a lancer en root (compile.sh d Armbian l exige) : sudo $0 $*"
command -v git >/dev/null    || die "git absent"
```

```
command -v docker >/dev/null || warn "docker absent : compile.sh ne pourra pas se conteneuriser (il faut alors un
 ↪  hote Ubuntu avec ses dependances)"
HOST_ARCH="$(dpkg --print-architecture 2>/dev/null || uname -m)"
if [ "$HOST_ARCH" != arm64 ] && [ "$NO_STACK" = 0 ]; then
    if [ ! -e /proc/sys/fs/binfmt_misc/qemu-aarch64 ]; then
        if [ "$INSTALL_DEPS" = 1 ]; then
            say "emulation aarch64 absente : apt install qemu-user-static binfmt-support"
            run apt-get install -y qemu-user-static binfmt-support
        elif [ "$DRY" = 1 ] || [ "$PREPARE" = 1 ]; then
            warn "pas d emulation aarch64 sur cet hote $HOST_ARCH : avant compile.sh, sudo $REPO/build.sh --arm
             ↪  (pose tous les paquets hote) ou --install-deps"
        else
            die "pas d emulation aarch64 sur cet hote $HOST_ARCH (/proc/sys/fs/binfmt_misc/qemu-aarch64 absent) :
             ↪  customize-image.sh ne pourrait pas tourner dans le chroot arm64. sudo $REPO/build.sh --arm pose
             ↪  tous les paquets hote (ou --install-deps)"
        fi
    fi
fi
OPI5_BOARDS="orangepi5 orangepi5b orangepi5-plus orangepi5pro orangepi5-max orangepi5-ultra"
BOARDS=""
for b in ${BOARD//,/ }; do
    case "$b" in
        rpi4|rpi4b) BOARDS="$BOARDS rpi4b" ;;
        opi5|orangepi5-all) BOARDS="$BOARDS $OPI5_BOARDS" ;;
        all)        BOARDS="$BOARDS rpi4b $OPI5_BOARDS" ;;
        *)          BOARDS="$BOARDS $b" ;;
    esac
done
BOARDS=$(echo $BOARDS | tr ' ' '\n' | awk '!vu[$0]++' | tr '\n' ' ')
say "hote $HOST_ARCH ($(nproc) coeurs) ; cartes : $BOARDS; RELEASE=$RELEASE ; bureau=$([ "$DESKTOP" = 1 ] && echo
 ↪  gnome || echo non) ; clone $ARMBIAN_DIR"

# ── 2. le framework ──────────────────────────────────────────────────────
if [ ! -f "$ARMBIAN_DIR/compile.sh" ]; then
    say "clone de $ARMBIAN_GIT (branche main, depth 1) -> $ARMBIAN_DIR"
    run git clone --depth 1 --branch main "$ARMBIAN_GIT" "$ARMBIAN_DIR"
elif [ "$UPDATE" = 1 ]; then
    say "mise a jour du clone"
    run git -C "$ARMBIAN_DIR" pull --ff-only
fi
# board_file CARTE : le fichier de carte d Armbian (.conf supportee, .csc communautaire, .wip)
board_file() {
    local e
    for e in conf csc wip tvb eos; do
        [ -f "$ARMBIAN_DIR/config/boards/$1.$e" ] && { echo "$ARMBIAN_DIR/config/boards/$1.$e"; return 0; }
    done
    return 1
}
# board_branch CARTE : --branch si la carte la propose, sinon current, sinon sa premiere branche
board_branch() {
    local f targets
    f=$(board_file "$1") || { echo "$BRANCH"; return; }
    targets=$(sed -n 's/^KERNEL_TARGET="\(.*\)".*/\1/p' "$f" | head -1)
    [ -n "$targets" ] || { echo "$BRANCH"; return; }
    case ",$targets," in
        *",$BRANCH,"*) echo "$BRANCH" ;;
        *",current,"*) [ "$BRANCH_GIVEN" = 1 ] && warn "$1 : pas de branche $BRANCH ($targets), current a la
         ↪  place" >&2; echo current ;;
        *) warn "$1 : pas de branche $BRANCH$([ "$BRANCH" = current ] || echo " ni current") ($targets),
         ↪  ${targets%%,*} a la place" >&2; echo "${targets%%,*}" ;;
    esac
}
if [ "$DRY" = 0 ]; then
    for b in $BOARDS; do
        board_file "$b" >/dev/null || die "carte inconnue d Armbian : $b (voir $ARMBIAN_DIR/config/boards/)"
    done
```

```
fi

# ── 3. userpatches : le script de personnalisation et l overlay ──────────────
UP="$ARMBIAN_DIR/userpatches"
OV="$UP/overlay/osmo"
run install -d "$UP" "$OV"
if [ "$NO_STACK" = 1 ]; then
    say "--no-stack : pas de customize-image.sh (Armbian nue)"
    run rm -f "$UP/customize-image.sh"
else
    run install -m755 "$HERE/userpatches/customize-image.sh" "$UP/customize-image.sh"
    say "userpatches/customize-image.sh pose"
fi

# ── 4. l overlay : la pile, exportee de l image Docker arm64 ─────────────────
if [ "$NO_STACK" = 0 ]; then
    if [ -z "$IMAGE" ]; then
        if docker image inspect osmocom-nitb:arm64 >/dev/null 2>&1; then
            IMAGE=osmocom-nitb:arm64
        else
            KEY=$(cd "$REPO" && git ls-files -- Dockerfile packaging configs scripts services helpers opt-gsm
             ↪  tools patches | sort | xargs sha256sum | sha256sum | cut -c1-16)
            REMOTE=$(cd "$REPO" && git remote get-url origin 2>/dev/null | sed -E 's#.*github.com[:/]##;
             ↪  s#\.git$##' | tr '[:upper:]' '[:lower:]')
            GHCR="ghcr.io/${REMOTE:-bbaranoff/osmo-operator}/osmocom-nitb:base-arm64-$KEY"
            if [ "$DRY" = 0 ] && docker pull --platform linux/arm64 "$GHCR" >/dev/null 2>&1; then
                IMAGE="$GHCR"
            elif [ "$DRY" = 1 ]; then
                warn "aucune image Docker arm64 de la pile en local ($GHCR non testee en dry-run) : il faudra d
                 ↪  abord sudo $REPO/build.sh --arch=arm64"
                IMAGE="osmocom-nitb:arm64"
            elif [ "$BUILD_IMAGE" = 1 ]; then
                say "aucune image Docker arm64 de la pile : build.sh --arch=arm64 (emule sur un hote x86 : ~10 h au
                 ↪  premier build)"
                bash "$REPO/build.sh" --arch=arm64 || die "build.sh --arch=arm64 a echoue"
                if docker image inspect osmocom-nitb:arm64 >/dev/null 2>&1; then IMAGE=osmocom-nitb:arm64; else
                 ↪  IMAGE=osmocom-nitb; fi
            else
                die "aucune image Docker arm64 de la pile : ni osmocom-nitb:arm64 en local, ni $GHCR. Sur cet hote
                 ↪  : sudo $REPO/build.sh --arch=arm64 (emule, ~10 h au premier build) ou --build-image, ou lancer
                 ↪  docker-arm.yml sur GitHub (runner arm64 natif) puis relancer ici."
            fi
        fi
    fi
    if [ "$DRY" = 0 ]; then
        ARCH=$(docker image inspect --format '{{.Architecture}}' "$IMAGE" 2>/dev/null || echo "?")
        [ "$ARCH" = arm64 ] || die "l image $IMAGE est $ARCH, pas arm64"
    fi
    say "pile : image Docker $IMAGE -> overlay $OV"
    if [ "$DRY" = 1 ]; then
        echo "  + docker create/cp -> $OV/debs/ et $OV/rootfs/ (usr/local, opt/GSM, opt/node, root/.env,
         ↪  etc/osmocom, etc/asterisk, lib/systemd/system)"
    else
        rm -rf "$OV"; install -d "$OV/debs" "$OV/rootfs/usr" "$OV/rootfs/opt" "$OV/rootfs/root" "$OV/rootfs/etc"
         ↪  "$OV/rootfs/lib/systemd/system"
        CID=$(docker create --platform linux/arm64 "$IMAGE" /bin/true)
        trap 'docker rm "$CID" >/dev/null 2>&1 || true' EXIT
        # les .deb du build : ceux de la suite cible, cache de l hote d abord, sinon ceux de l image
        cp -f "$DEB_CACHE"/osmo-build-*"~${RELEASE}_arm64.deb" "$OV/debs/" 2>/dev/null || true
        if ! ls "$OV/debs"/*.deb >/dev/null 2>&1; then
            docker cp "$CID:/var/cache/osmo-debs/." "$OV/debs/" 2>/dev/null || true
            find "$OV/debs" -name '*.deb' ! -name "*~${RELEASE}_arm64.deb" -delete 2>/dev/null || true
        fi
        NDEBS=$(find "$OV/debs" -name '*.deb' | wc -l)
        # L overlay est un ARBRE (rootfs/), recopie tel quel sur / par customize-image.sh : /usr/local
        # (binaires, libs), /opt/GSM sans les arbres que les .deb posent deja (meme liste que
```

```
        # 50-injection-image.sh), /opt/node, l environnement de root, /etc/osmocom, /etc/asterisk, les unites.
        docker cp "$CID:/usr/local" "$OV/rootfs/usr/"
        EXCL=()
        if [ "$NDEBS" -gt 0 ]; then
            for t in osmocom-bb qosmo c54x_exe grgsm_exe qemu-install firmware; do EXCL+=("--exclude=GSM/$t"
             ↪  "--exclude=GSM/$t/*"); done
        fi
        docker cp "$CID:/opt/GSM" - | tar -x -C "$OV/rootfs/opt" "${EXCL[@]+"${EXCL[@]}"}"
        docker cp "$CID:/opt/node" "$OV/rootfs/opt/" 2>/dev/null || true
        for d in .env .venv-qemu; do docker cp "$CID:/root/$d" "$OV/rootfs/root/" 2>/dev/null || true; done
        docker cp "$CID:/etc/osmocom" "$OV/rootfs/etc/" 2>/dev/null || true
        docker cp "$CID:/etc/asterisk" "$OV/rootfs/etc/" 2>/dev/null || true
        for svc in osmo-bts-trx osmo-bsc osmo-msc osmo-hlr osmo-mgw osmo-stp osmo-ggsn osmo-sgsn osmo-pcu
         ↪  osmo-sip-connector; do
           docker cp "$CID:/lib/systemd/system/$svc.service" "$OV/rootfs/lib/systemd/system/" 2>/dev/null || true
        done
        cp -f "$REPO"/services/*.service "$OV/rootfs/lib/systemd/system/" 2>/dev/null || true
        docker rm "$CID" >/dev/null; trap - EXIT
        # [2026-10-24] osmo-operator = CET arbre (fichiers suivis + non suivis, sans .git ni ignores),
        # comme l ISO qui fait foi sur le depot et non sur la copie de l image : le bureau (data/,
        # configs/, tools/) et les correctifs pas encore dans une image publiee.
        rm -rf "$OV/rootfs/opt/GSM/osmo-operator"; install -d "$OV/rootfs/opt/GSM/osmo-operator"
        (cd "$REPO" && git ls-files -co --exclude-standard -z | tar --null --ignore-failed-read -T - -cf -
         ↪  2>/dev/null) \
            | tar -xf - -C "$OV/rootfs/opt/GSM/osmo-operator"
        # le gardien de l encart et du Conky : le MEME que l ISO, extrait de 85-installeur-bureau.sh
        sed -n "/<<'PANEL'$/,/^PANEL$/p" "$REPO/iso_modules/85-installeur-bureau.sh" | sed '1d;$d' >
         ↪  "$OV/osmo-desktop-panel"
        [ -s "$OV/osmo-desktop-panel" ] || warn "osmo-desktop-panel non extrait de
         ↪  iso_modules/85-installeur-bureau.sh : pas de Conky au login"
        printf 'IMAGE=%s\nRELEASE=%s\nBOARDS=%s\nDESKTOP=%s\nDATE=%s\nNDEBS=%s\n' "$IMAGE" "$RELEASE" "$BOARDS"
         ↪  "$DESKTOP" "$(date -Is)" "$NDEBS" > "$OV/SOURCE"
        say "overlay pret : $NDEBS .deb, $(du -sh "$OV" | cut -f1)"
    fi
fi

# Les marqueurs que customize-image.sh lit dans l overlay (les services actives au boot)
if [ "$NO_STACK" = 0 ] && [ "$DRY" = 0 ]; then
    rm -f "$OV/OSMO_IMAGE_BANC" "$OV/OSMO_IMAGE_MULTI" "$OV/OSMO_ARM_GLAVA"
    [ "${OSMO_ARM_GLAVA:-0}" = 1 ] && : > "$OV/OSMO_ARM_GLAVA"
    [ "$BANC" = 1 ]  && : > "$OV/OSMO_IMAGE_BANC"
    [ "$MULTI" = 1 ] && : > "$OV/OSMO_IMAGE_MULTI"
fi

# ── 5. compile.sh, une fois par carte ──────────────────────────────────
# Le bureau : armbian/build ne prend plus DESKTOP_ENVIRONMENT_CONFIG_NAME ni
# DESKTOP_APPGROUPS_SELECTED (lib/functions/configuration/config-desktop.sh) : le
# bureau vient des YAML de armbian/configng, choisi par DESKTOP_ENVIRONMENT + DESKTOP_TIER.
# GNOME minimal y tire gdm3, pulseaudio et Xorg - ce que le Conky et c123 attendent.
if [ "$DESKTOP" = 1 ]; then
    DESK=(BUILD_DESKTOP=yes DESKTOP_ENVIRONMENT=gnome DESKTOP_TIER=minimal)
else
    DESK=(BUILD_DESKTOP=no)
fi
IMGS=()
for b in $BOARDS; do
    br=$(board_branch "$b")
    CMD=(./compile.sh build "BOARD=$b" "BRANCH=$br" "RELEASE=$RELEASE" BUILD_MINIMAL=no "${DESK[@]}"
     ↪  KERNEL_CONFIGURE=no COMPRESS_OUTPUTIMAGE=sha,img)
    case "$(board_file "$b" 2>/dev/null)" in *.conf|"") ;; *) CMD+=(EXPERT=yes) ;; esac
    if [ "$PREPARE" = 1 ] || [ "$DRY" = 1 ]; then
        echo "  + (cd $ARMBIAN_DIR && ${CMD[*]})"
        continue
    fi
    say "[$b] compile.sh (long : noyau + rootfs + customize sous qemu-user) : ${CMD[*]}"
    t0=$(date +%s)
```

```
    (cd "$ARMBIAN_DIR" && "${CMD[@]}") || die "[$b] compile.sh a echoue"
    IMG=$(find "$ARMBIAN_DIR/output/images" -maxdepth 1 -name '*.img' -newermt "@$t0" 2>/dev/null | head -1)
    [ -n "$IMG" ] || die "[$b] pas de nouvelle image dans $ARMBIAN_DIR/output/images/"
    IMGS+=("$IMG")
    say "[$b] image : $IMG  ($(du -h "$IMG" | cut -f1))"
done
if [ "$PREPARE" = 1 ]; then
    say "--prepare : fini. Pour construire : les commandes ci-dessus, en root"
    exit 0
fi
[ "$DRY" = 1 ] && exit 0
say "images :"; for i in "${IMGS[@]}"; do echo "    $i"; done
say "flash : sudo dd if=<image> of=/dev/sdX bs=4M status=progress conv=fsync   (comptes : root/osmo, osmocom/osmo)"
```

3.34 /opt/GSM/osmo-operator/armbian/userpatches/customize-image.sh

13772 octets, 195 lignes → 195 lignes

```
#!/bin/bash
# armbian/userpatches/customize-image.sh - la pile osmo-operator posee dans l image Armbian
#
# Execute par armbian/build DANS LE CHROOT de l image (arguments : $RELEASE $LINUXFAMILY $BOARD
# $BUILD_DESKTOP), avec userpatches/overlay monte en /tmp/overlay. Ce que fait l ISO en plusieurs
# modules (50-injection-image, 80-chroot, 82-arm-natif, 84-comptes), ramene a l essentiel :
#   1. les paquets Ubuntu dont la pile a besoin a l execution (liste de 80-chroot.sh, sans noyau ni live)
#   2. dpkg -i des .deb du build docker (osmo-build-* : libosmocore, osmo-*, gapk, QEMU, osmocom-bb...)
#   3. l arbre /tmp/overlay/osmo/rootfs recopie sur / (/usr/local, /opt/GSM, /etc/osmocom, unites...)
#   4. ce qu un hote x86 ne peut pas produire, compile ICI (aarch64) : toast, lanceur qosmo,
#      c54x_exe / grgsm_exe, les .deb du banc (packaging/build-debs.sh)
#   5. comptes (root/osmo, osmocom/osmo), plus d assistant Armbian au premier login, services
# Le noyau, le firmware du Pi, l agrandissement de la racine : Armbian s en charge.
set -euo pipefail
RELEASE="${1:-}"; LINUXFAMILY="${2:-}"; BOARD="${3:-}"; BUILD_DESKTOP="${4:-}"
OV=/tmp/overlay/osmo
export DEBIAN_FRONTEND=noninteractive LC_ALL=C.UTF-8 LANG=C.UTF-8
log() { echo "[osmo-customize] $*"; }
log "release=$RELEASE family=$LINUXFAMILY board=$BOARD desktop=$BUILD_DESKTOP overlay=$OV"
[ -d "$OV" ] || { log "pas d overlay $OV : image Armbian nue"; exit 0; }
[ -f "$OV/SOURCE" ] && log "pile : $(tr '\n' ' ' < "$OV/SOURCE")"
case "$RELEASE" in noble) ;; *) log "ATTENTION : les .deb du build visent noble, pas $RELEASE" ;; esac

# ── 1. paquets d execution ──────────────────────────────────────────
apt-get update
PKGS="ca-certificates openssl netcat-openbsd socat tcpdump git logrotate systemd-resolved build-essential wget
      libtalloc2 libtalloc-dev libpcsclite1 libsctp1 libsctp-dev libc-ares2
      libgnutls30t64 libgnutls28-dev libmnl-dev libmnl0
      libortp-dev libdbi1t64 libdbd-sqlite3 sqlite3
      libfftw3-single3 libusb-1.0-0 libgsm1 libasound2t64
      libsofia-sip-ua-glib3t64 liburing2 libslirp0
      iproute2 iptables net-tools lksctp-tools
      tmux telnet expect whiptail lsb-release openssh-server sudo
      console-setup keyboard-configuration locales psmisc cryptsetup rsync
      python3 python3-venv python3-scapy tshark wireshark-common
      libasound2-plugins pulseaudio pulseaudio-utils alsa-utils
      binutils-arm-none-eabi gdb-multiarch asterisk ffmpeg
      libzmq5 libfftw3-dev cmake libpcsclite-dev meson ninja-build flex bison
      libgcrypt20-dev libssl-dev libidn-dev libyaml-dev libnghttp2-dev libmicrohttpd-dev libcurl4-gnutls-dev
      libtins-dev libc-ares-dev zstd kpartx"
# shellcheck disable=SC2086
apt-get install -y --no-install-recommends $PKGS || {
    log "apt : certains paquets manquent sur $RELEASE/arm64, on pose ce qui existe"
    for p in $PKGS; do apt-get install -y --no-install-recommends "$p" >/dev/null 2>&1 || log "  absent : $p"; done
}
```

```
# ── 2. les .deb du build docker ──────────────────────────────────────
if ls "$OV"/debs/*.deb >/dev/null 2>&1; then
    install -d /var/cache/osmo-debs
    cp -f "$OV"/debs/*.deb /var/cache/osmo-debs/
    dpkg -i --force-overwrite /var/cache/osmo-debs/osmo-build-*.deb || apt-get install -y -f
    ldconfig
    log "$(ls /var/cache/osmo-debs/*.deb | wc -l) paquets du build poses"
else
    log "pas de .deb du build dans l overlay : tout vient de l arbre rootfs/"
fi

# ── 3. l arbre de l image docker ─────────────────────────────────────
if [ -d "$OV/rootfs" ]; then
    cp -a "$OV/rootfs/." /
    ldconfig
    log "arbre recopie : $(du -sh "$OV/rootfs" | cut -f1) (/usr/local, /opt/GSM, /etc/osmocom, unites)"
fi

# ── 4. compilation native (aarch64) ──────────────────────────────────
if [ ! -x /usr/local/bin/toast ]; then
    ( set -e; cd /opt/GSM
      wget -qO gsm-1.0.24.tar.gz https://www.quut.com/gsm/gsm-1.0.24.tar.gz
      tar xzf gsm-1.0.24.tar.gz && rm -f gsm-1.0.24.tar.gz && cd gsm-1.0-pl24
      make >/dev/null 2>&1 || true; [ -x bin/toast ] || make toast >/dev/null 2>&1
      for b in toast untoast tcat; do [ -e "bin/$b" ] && install -m755 "bin/$b" "/usr/local/bin/$b"; done
    ) && log "toast compile" || log "toast non compile (on continue)"
fi
for fork in qosmo; do
    if [ -d "/opt/GSM/$fork/tools/qosmo-launch" ]; then
        make -s -C "/opt/GSM/$fork/tools/qosmo-launch" ALIAS="$fork" "$fork" && install -Dm755
        ↪  "/opt/GSM/$fork/tools/qosmo-launch/$fork" "/usr/local/bin/$fork" \
           && log "lanceur /usr/local/bin/$fork compile" || log "lanceur $fork : echec (run.sh appellera
            ↪  qemu-system-arm directement)"
    fi
done
for e in c54x_exe grgsm_exe; do
    [ -d "/opt/GSM/$e" ] && { make -s -B -C "/opt/GSM/$e" && log "$e compile" || log "$e : compilation echouee"; }
done
/usr/local/bin/qemu-system-arm --version >/dev/null 2>&1 && log "qemu-system-arm (aarch64) s execute" || log
↪  "qemu-system-arm absent ou KO"
if [ -x /opt/GSM/osmo-operator/packaging/build-debs.sh ]; then
    install -d /var/cache/osmo-debs
    env OSMO_OPERATOR_SRC=/opt/GSM/osmo-operator QOSMO_SRC=/opt/GSM/qosmo bash
     ↪  /opt/GSM/osmo-operator/packaging/build-debs.sh --out /var/cache/osmo-debs \
        && log ".deb du banc construits dans /var/cache/osmo-debs" || log ".deb du banc non construits (l image
         ↪  sort sans eux)"
fi

# ── 4b. le bureau du banc (BUILD_DESKTOP=yes : GNOME minimal pose par Armbian) ─
# [2026-10-24] Ce que l ISO --desktop pose en plus de GNOME (iso_modules/80-chroot.sh,
# 85-installeur-bureau.sh), ramene a ce qui a un sens sur une carte : X11 en autologin
# root, le fond GSM LAB, sombre/Yaru bleu, les icones du depot et le dock, le Conky et
# l encart (osmo-desktop-panel, extrait par build-armbian.sh), c123 et ses paquets GTK.
# Pas de calamares, grub, secure boot, firefox Mozilla, extensions GNOME.
# GLava N EST PAS pose par defaut : le cercle NCS veut OpenGL >= 4.2 (imageLoad/Store),
# que ni le V3D du Pi 4 ni le Mali G610 des Orange Pi 5 n offrent sous Mesa (GL 3.1).
# OSMO_ARM_GLAVA=1 sudo ./build-armbian.sh ... pour essayer quand meme (marqueur OSMO_ARM_GLAVA de l overlay).
if [ "$BUILD_DESKTOP" = yes ]; then
    REPO=/opt/GSM/osmo-operator
    DPKGS="conky-all fonts-dejavu python3-pil python3-gi python3-gi-cairo python3-cairo
           gir1.2-gtk-3.0 gir1.2-gdkpixbuf-2.0 gir1.2-vte-2.91 wmctrl x11-utils x11-xserver-utils
           xdotool zenity librsvg2-common yaru-theme-gtk yaru-theme-icon yaru-theme-gnome-shell
           wireshark vlc linphone-desktop gnome-terminal"
    apt-get update
    # shellcheck disable=SC2086
    apt-get install -y --no-install-recommends $DPKGS || {
```

```
    for p in $DPKGS; do apt-get install -y --no-install-recommends "$p" >/dev/null 2>&1 || log "  bureau,
     ↪  absent : $p"; done
}

# gdm : autologin root sous X11 (le banc tourne en root ; xdotool, xprop, le Conky veulent X)
sed -i "/pam_succeed_if.so user != root quiet_success/s/^/#/" \
    /etc/pam.d/gdm-password /etc/pam.d/gdm-autologin 2>/dev/null || true
install -d /etc/gdm3
printf '[daemon]\nAutomaticLoginEnable=true\nAutomaticLogin=root\nWaylandEnable=false\n' >
 ↪  /etc/gdm3/custom.conf
rm -f /etc/xdg/autostart/gnome-initial-setup-first-login.desktop

# les defauts de schema : fond, theme, veille, clavier, dock (memes valeurs que l ISO)
OVR=/usr/share/glib-2.0/schemas/99-osmo-live.gschema.override
FOND=/usr/share/backgrounds/gsm-lab-ville.jpg
for f in gsm-lab-ville.jpg gsm-lab-antenne.jpg gsm-lab-fond.jpg; do
    [ -f "$REPO/configs/$f" ] && install -Dm644 "$REPO/configs/$f" "/usr/share/backgrounds/$f"
done
[ -f "$FOND" ] || FOND=/usr/share/backgrounds/gsm-lab-fond.jpg
printf "[org.gnome.desktop.session]\nidle-delay=uint32
 ↪  0\n\n[org.gnome.desktop.screensaver]\nlock-enabled=false\nidle-activation-
 ↪  enabled=false\n\n[org.gnome.settings-daemon.plugins.power]\nsleep-inactive-ac-type='nothing'\nsleep-
 ↪  inactive-battery-type='nothing'\n\n[org.gnome.desktop.input-sources]\nsources=[('xkb','%s')]\n"
 ↪  \
    "${OSMO_KB:-fr}" > "$OVR"
for sec in org.gnome.desktop.interface org.gnome.desktop.interface:GNOME-Greeter; do
    printf "\n[%s]\ncolor-scheme='default'\ngtk-theme='Yaru-blue'\nicon-theme='Yaru-blue'\ncursor-
     ↪  theme='Yaru'\n" "$sec" >>
     ↪  "$OVR"
done
if [ -f "$FOND" ]; then
    for sec in org.gnome.desktop.background org.gnome.desktop.background:GNOME-Greeter; do
        printf "\n[%s]\npicture-uri='file://%s'\npicture-uri-dark='file://%s'\npicture-
         ↪  options='zoom'\nprimary-color='#0d1b2a'\n" "$sec" "$FOND" "$FOND" >>
         ↪  "$OVR"
    done
fi
printf "\n[org.gnome.shell]\nfavorite-apps=['osmo-info.desktop', 'org.gnome.Nautilus.desktop',
 ↪  'osmo-launch.desktop', 'osmo-tutorial.desktop', 'osmo-addition.desktop', 'osmo-update.desktop',
 ↪  'linphone.desktop', 'org.wireshark.Wireshark.desktop', 'org.gnome.Terminal.desktop']\n" >> "$OVR"
printf "[org.gnome.settings-daemon.plugins.media-keys]\ncustom-keybindings=['/org/gnome/settings-
 ↪  daemon/plugins/media-keys/custom-keybindings/osmo-op-next/','/org/gnome/settings-daemon/plugins/media-
 ↪  keys/custom-keybindings/osmo-op-prev/']\n\n[org.gnome.settings-daemon.plugins.media-keys.custom-
 ↪  keybinding:/org/gnome/settings-daemon/plugins/media-keys/custom-keybindings/osmo-op-next/]\nname='osmo-
 ↪  operator : arret suivant (Conky)'\ncommand='/usr/local/bin/osmo-op
 ↪  --next'\nbinding='<Control><Mod5>Right'\n\n[org.gnome.settings-daemon.plugins.media-keys.custom-
 ↪  keybinding:/org/gnome/settings-daemon/plugins/media-keys/custom-keybindings/osmo-op-prev/]\nname='osmo-
 ↪  operator : arret precedent (Conky)'\ncommand='/usr/local/bin/osmo-op
 ↪  --prev'\nbinding='<Control><Mod5>Left'\n" \
    > /usr/share/glib-2.0/schemas/98-osmo-keys.gschema.override
glib-compile-schemas /usr/share/glib-2.0/schemas 2>/dev/null || log "glib-compile-schemas a echoue"

# les icones du depot (sauf multi : posee par addition.sh --multi, cf. 85-installeur-bureau.sh)
install -d /usr/share/applications /usr/share/icons/hicolor/scalable/apps
for d in "$REPO"/data/desktop/*.desktop; do
    [ -f "$d" ] || continue
    e=$(basename "$d" .desktop); [ "$e" = osmo-multi ] && continue
    install -m644 "$d" "/usr/share/applications/$e.desktop"
    [ -f "$REPO/data/$e.svg" ] && install -m644 "$REPO/data/$e.svg"
     ↪  "/usr/share/icons/hicolor/scalable/apps/$e.svg"
done
gtk-update-icon-cache -f -q /usr/share/icons/hicolor 2>/dev/null || true

# les commandes du bureau dans le PATH (memes liens que iso_modules/70-scripts.sh)
ln -sf "$REPO/start-native.sh"        /usr/local/bin/osmo-start-native
ln -sf "$REPO/tools/osmo-fond.py"     /usr/local/bin/osmo-fond
```

```
    ln -sf "$REPO/tools/osmo-op.sh"       /usr/local/bin/osmo-op
    ln -sf "$REPO/tools/osmo-drivers.sh"  /usr/local/bin/osmo-drivers
    ln -sf "$REPO/tools/overlay-draw.py"  /usr/local/bin/overlay-draw
    ln -sf "$REPO/tools/osmo-c123-gtk.py" /usr/local/bin/c123
    ln -sf "$REPO/tools/osmo-c123-gtk.py" /usr/local/bin/osmo-c123

    # le Conky et l encart, gardes en vie toute la session (le meme gardien que l ISO)
    if [ -s "$OV/osmo-desktop-panel" ]; then
        install -m755 "$OV/osmo-desktop-panel" /usr/local/bin/osmo-desktop-panel
        install -d /etc/xdg/autostart
        printf '[Desktop Entry]\nType=Application\nName=Conky osmo-operator\nComment=Tableau de bord du banc
         ↪ GSM\nExec=/usr/local/bin/osmo-desktop-panel\nIcon=utilities-system-
         ↪ monitor\nTerminal=false\nNoDisplay=true\nOnlyShowIn=GNOME;\nX-GNOME-Autostart-enabled=true\nX-GNOME-
         ↪ Autostart-Delay=6\n'
         ↪ \
            > /etc/xdg/autostart/osmo-conky.desktop
        log "bureau : Conky + encart a la connexion"
    fi

    if [ -f "$OV/OSMO_ARM_GLAVA" ]; then
        apt-get install -y --no-install-recommends git meson ninja-build gcc pkg-config patch \
            libpulse-dev libx11-dev libxext-dev libxrender-dev libxcomposite-dev libgl-dev >/dev/null \
            && bash "$REPO/tools/glava-build.sh" && log "GLava pose (OSMO_ARM_GLAVA)" \
            || log "GLava non pose"
    fi
    log "bureau du banc pose (fond $FOND)"
fi

# les alias du banc (banc-rssi, banc-trx, c123-1...) : profile.d ET vraies commandes, bureau ou non
if [ -f /opt/GSM/osmo-operator/scripts/osmo-banc-aliases.sh ]; then
    install -m644 /opt/GSM/osmo-operator/scripts/osmo-banc-aliases.sh /etc/profile.d/98-osmo-banc-aliases.sh
    bash /opt/GSM/osmo-operator/tools/osmo-banc-cmd.sh --install /usr/local/bin
fi

# ── 5. comptes, premier login, services ──────────────────────────────
id -u osmocom >/dev/null 2>&1 || useradd -m -s /bin/bash -c 'Compte osmocom (non privilegie)' osmocom
echo 'osmocom:osmo' | chpasswd
echo 'root:osmo'    | chpasswd
usermod -aG sudo,audio,dialout,plugdev osmocom 2>/dev/null || true
rm -f /root/.not_logged_in_yet            # pas d assistant Armbian au premier login : les comptes sont poses
for u in osmo-egprs-web osmo-pulse; do systemctl enable "$u" 2>/dev/null || true; done
for u in osmo-banc osmo-multi; do        # poses, pas actives : OSMO_IMAGE_BANC=1 / OSMO_IMAGE_MULTI=1 dans l
 ↪ overlay pour les activer
    v="OSMO_IMAGE_${u#osmo-}"; v="${v^^}"
    [ -f "$OV/$v" ] && systemctl enable "$u" 2>/dev/null && log "$u active au boot"
done
apt-get clean; rm -rf /var/lib/apt/lists/*
log "termine"
```

3.35 /opt/GSM/osmo-operator/checks/_mode.sh

40107 octets, 879 lignes → 879 lignes

```
# ============================================================================
#  checks/_mode.sh - une seule regle de detection docker/natif, pour tous
#                    les scripts de checks/
#  Bibliotheque : a SOURCER, jamais a executer dans une boucle de tests.
# ============================================================================
#
#  POURQUOI CE FICHIER EXISTE
#  --------------------------
#  Le lab vit dans deux mondes, et les checks doivent dire la verite dans les
#  deux :
#
```

```
#    docker - un conteneur osmo-operator-N par operateur sur le reseau
#             gsm-inter (172.20.0.0/24), le hub SS7 est le conteneur
#             osmo-inter-stp (172.20.0.10) ; on atteint tout par "docker exec".
#    natif  - ISO, VM ou machine nue : un seul operateur (sauf topologie netns),
#             tout tourne ici, les VTY sont sur 127.0.0.1, les configs dans
#             /etc/osmocom, les demons sont des units systemd ou des processus.
#
#  Les PORTS VTY sont les MEMES dans les deux mondes - seul l'hote change :
#      osmo-stp 4239 · osmo-msc 4254 · osmo-bsc 4242 · osmo-hlr 4258
#      osmo-sgsn 4245 · osmo-ggsn 4260 · osmo-mgw 4243 · mobile/BB 4247
#
#  Avant ce fichier, la meme detection etait recopiee dans cinq scripts, avec
#  cinq variantes : deux d'entre elles employaient "docker ps | grep -aq", qui
#  rend 141 sous "set -o pipefail" (grep sort au premier match, le producteur
#  prend un SIGPIPE) - l'inter-STP etait alors declare absent alors qu'il
#  tournait, de facon intermittente. Une seule regle, ecrite une fois, corrigee
#  une fois.
#
#  CONTRAT
#  -------
#    • le fichier ne definit QUE des fonctions et des variables ": =" ;
#      il n'affiche rien, ne pose aucune couleur, ne compte aucun PASS/FAIL :
#      chaque check garde ses ok/fail/warn/skip, donc son format de sortie et
#      son code de retour ;
#    • les sondes rendent 0 = vrai, non-0 = faux ;
#    • sourcable deux fois sans effet de bord (garde OSMO_MODE_LIB_LOADED) ;
#    • aucun "grep -aq" en fin de pipeline, aucun "awk ... exit" au milieu d'un
#      tube : sous "set -o pipefail" ces deux formes rendent 141 ;
#    • compatible "set -euo pipefail" chez l'appelant.
#
#  DEUX QUESTIONS DIFFERENTES - POURQUOI start-native.sh GARDE SA DETECTION
#  ------------------------------------------------------------------------
#  start-native.sh:159 a sa propre detect_runtime_env(), et elle N'EST PAS un
#  doublon de osmo_mode() : les deux fonctions repondent a deux questions qui
#  n'ont pas la meme reponse.
#
#    detect_runtime_env()  "ou est-ce que JE tourne ?"  → docker | vm | bare
#        Question de l'INTERIEUR du noeud. Elle decide du PLAN D'ADRESSAGE :
#        en docker l'inter-STP est 172.20.0.10 et le noeud a deja son IP ; en
#        VM/ISO l'inter-STP est une autre machine (192.168.1.49 sur le banc, ou une host-only selon le montage) et
 ↪  l'adresse
#        arrive par DHCP, donc peut manquer au lancement.
#
#    osmo_mode()           "comment j'ATTEINS les noeuds d'ici ?" → docker | native
#        Question de PILOTAGE. Elle decide entre "docker exec" et "executer
#        ici".
#
#  Elles se CONTREDISENT volontairement dans un cas, et c'est la preuve qu'il ne
#  faut pas les fusionner : DANS un conteneur du depot, detect_runtime_env rend
#  "docker" (le plan d'adressage est bien celui de docker) alors qu'osmo_mode
#  rend "native" - car "docker exec" vers soi-meme n'a aucun sens, meme
#  quand le socket est monte. Un script qui prendrait l'une pour l'autre
#  s'enverrait des commandes a lui-meme par le demon docker.
#
#  Ce qui EST partage, en revanche, c'est la convention aval : start-native.sh
#  transmet "--$NODE_MODE" (--docker / --native) a network/setup-wan-mesh.sh
#  (start-native.sh:756), et c'est exactement ce que consomme osmo_mode_force().
#  Les deux vocabulaires coincident ; seules les fonctions restent distinctes.
#
#  DESIGNER UN NOEUD - <op>
#  ------------------------
#  Toutes les fonctions d'execution prennent en premier argument un <op>, qui
#  accepte indifferemment :
#      3                   le numero d'operateur (forme normale)
#      osmo-operator-3     l'etiquette (ce que rendent osmo_ops/osmo_node)
#      hub | osmo-inter-stp le hub SS7
```

```
#  En docker c'est le nom du conteneur ; en natif c'est cette machine, avec le
#  prefixe "ip netns exec osmo-opN" si et seulement si l'espace de noms
#  existe (topologie multi-operateur native, run_modules/12-netns.sh).
#
#  L'ETIQUETTE osmo-operator-N EST CONSERVEE EN NATIF. Ce n'est pas une
#  coquetterie : checks/vty-debug-dump.sh ecrit "container=osmo-operator-N"
#  dans son dump, et checks/operator_summary.sh y accroche quinze extractions.
#  L'etiquette est une cle de correlation, pas un objet interrogeable.
#
#  SOURCING (le meme bloc partout ; marche depuis le depot et depuis /opt) :
#      _here="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
#      for _c in "$_here/_mode.sh" /opt/GSM/osmo-operator/checks/_mode.sh; do
#          [ -r "$_c" ] && { . "$_c"; break; }
#      done
#      command -v osmo_mode >/dev/null || { echo "checks/_mode.sh introuvable" >&2; exit 1; }
# ============================================================================

[ -n "${OSMO_MODE_LIB_LOADED:-}" ] && return 0
OSMO_MODE_LIB_LOADED=1

# Racine du depot : ce fichier est dans checks/, la racine est un cran au-dessus.
# Sert a retrouver globals.conf ; surchargeable pour les tests.
_osmo_lib_here="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
: "${OSMO_REPO:=$(dirname "$_osmo_lib_here")}"
unset _osmo_lib_here

# Plan d'adressage de la dorsale inter-operateurs. La MEME valeur sert aux deux
# mondes : docker la pose avec "--network gsm-inter --ip", le natif
# multi-operateur la recree en netns (BACKBONE_NET, run_modules/12-netns.sh:47).
: "${OSMO_BACKBONE_NET:=172.20.0}"
: "${OSMO_NETNS_PREFIX:=${NETNS_PREFIX:-osmo-op}}"

# Prefixe de racine, mode natif seulement : permet de viser un arbre de test
# (OSMO_NATIVE_ROOT=/tmp/essai) au lieu du /etc de la machine. Vide en
# production. Meme role que NATIVE_ROOT dans network/setup-wan-mesh.sh.
: "${OSMO_NATIVE_ROOT:=}"

# Memo "quel client VTY ce noeud possede-t-il ?", pour ne pas relancer un
# "command -v nc" a chaque commande VTY (global_check en passe des dizaines).
# "-g" : la bibliotheque doit rester utilisable meme sourcee DEPUIS une
# fonction, ou un "declare" nu creerait une variable locale qui disparaitrait
# au retour.
declare -gA _OSMO_VTY_TOOL 2>/dev/null || declare -A _OSMO_VTY_TOOL

# ============================================================================
#  1. MODE
# ============================================================================

# osmo_in_container - tourne-t-on DANS un conteneur DE CE DEPOT ?
# /.dockerenv seul serait vrai dans n'importe quel conteneur ; on exige aussi
# /etc/docker-entrypoint-cmd, depose par scripts/entrypoint.sh. C'est le couple
# qu'emploient start-docker.sh:20 et start-native.sh:159.
osmo_in_container() {
    [ -f /.dockerenv ] && [ -f /etc/docker-entrypoint-cmd ]
}

# Detection, du plus explicite au plus deduit.
_osmo_mode_detect() {
    # Dans un conteneur du depot, on EST le noeud : "docker exec" vers
    # soi-meme n'a pas de sens, meme si le socket docker est monte (c'est le cas
    # du conteneur du dashboard). C'est la regle de start-native.sh:237.
    if osmo_in_container; then printf 'native\n'; return 0; fi

    # OSMO_NATIVE=1 est deja pose par l'ISO (services/osmo-egprs-web.service).
    if [ "${OSMO_NATIVE:-}" = "1" ]; then printf 'native\n'; return 0; fi
```

```
    # LA regle du depot (checks/wan_ss7_check.sh:88, network/setup-wan-mesh.sh:122) :
    # on ne se fie pas a la presence du binaire docker - l'hote du lab l'a, une
    # machine de dev aussi - mais a celle d'un conteneur operateur EN COURS.
    # On capture la liste AVANT de la filtrer : "docker ps | grep -aq" rendrait
    # 141 sous pipefail.
    if command -v docker >/dev/null 2>&1; then
        local names
        names="$(docker ps --format '{{.Names}}' 2>/dev/null || true)"
        if grep -ax 'osmo-operator-1' >/dev/null 2>&1 <<<"$names"; then
            printf 'docker\n'; return 0
        fi
    fi
    printf 'native\n'
}

# osmo_mode - ecrit "docker" ou "native". Memoise dans OSMO_MODE, qui fait
# aussi office d'entree : "OSMO_MODE=native ./checks/global_check.sh" force.
osmo_mode() {
    case "${OSMO_MODE:-}" in
        docker|native) : ;;
        *) OSMO_MODE="$(_osmo_mode_detect)" ;;
    esac
    printf '%s\n' "$OSMO_MODE"
}

osmo_is_docker() { [ "$(osmo_mode)" = docker ]; }
osmo_is_native() { [ "$(osmo_mode)" = native ]; }

# osmo_mode_force <docker|native|--docker|--native|--mode=X>
# A glisser tel quel dans la boucle "case" d'analyse des options de chaque
# script : l'explicite gagne toujours sur la detection. Rend 2 sur un argument
# qui n'est pas un mode (l'appelant sait alors que l'option ne le concerne pas).
osmo_mode_force() {
    local m="${1:-}"
    case "$m" in
        --mode=*) m="${m#*=}" ;;
        --docker) m=docker ;;
        --native) m=native ;;
    esac
    case "$m" in
        docker|native) OSMO_MODE="$m"; return 0 ;;
        *) return 2 ;;
    esac
}

# ============================================================================
#  2. INVENTAIRE
# ============================================================================

# Lit une cle de globals.conf ("N_OPERATORS=3"). On lit le fichier avec awk
# directement : pas de tube, donc pas de piege pipefail.
_osmo_globals() {
    local k="${1:-}" f v
    for f in "${OSMO_GLOBALS:-}" "$OSMO_REPO/globals.conf" \
             /opt/GSM/osmo-operator/globals.conf ./globals.conf; do
        [ -n "$f" ] && [ -r "$f" ] || continue
        v="$(awk -F= -v k="$k" '$1 == k { gsub(/[" \r\t]/, "", $2); if ($2 != "") print $2 }' "$f" | tail -1)"
        [ -n "$v" ] && { printf '%s\n' "$v"; return 0; }
    done
    return 1
}

# osmo_ops - identifiants des operateurs presents, un par ligne, tries.
#   docker : les conteneurs osmo-operator-N en cours.
#   natif  : $OSMO_OP_IDS (pose par services/osmo-egprs-web.service), puis les
#            netns osmo-opN (multi-operateur natif), puis N_OPERATORS de
```

```
#            globals.conf, sinon 1. Aucun inventaire n'existe en natif : la vie
#            d'un noeud se deduit des demons, pas d'une liste de conteneurs.
# == LA TOPOLOGIE MULTI-OPERATEUR ============================================
# [2026-08-31] Ces checks supposaient UN SEUL mode pour tout le banc :
# osmo_mode() rend "docker" ou "native", et osmo_exec/osmo_ops/osmo_hub_ip s en
# servent globalement. C etait vrai tant qu une machine portait soit des
# conteneurs, soit un operateur natif.
#
# Le banc multi-operateur est MIXTE : op 1 natif sur l hote, op 2 et 3 en
# conteneurs, hub en conteneur. Le mode global tranchait donc pour tout le
# monde, et le check partait interroger un "osmo-operator-1" qui n existe plus,
# au point code 1.23.2 qui n est celui de personne - d ou des "MSC VTY 4254
# inaccessible" qui ne disaient rien de l etat reel du banc, juste qu on
# sondait la mauvaise adresse.
#
# /etc/osmocom/osmo-multi.conf est ecrit par addition.sh et fait autorite :
#   MULTI_OPS="1:native::1.1.2:150 2:docker:172.20.0.12:1.2.2:250 ..."
#   MULTI_HUB_IP=172.20.0.10  MULTI_HUB_NAME=osmo-inter-stp
# On le lit quand il existe ; sinon RIEN ne change, l ancienne detection reste.
OSMO_MULTI_CONF="${OSMO_MULTI_CONF:-/etc/osmocom/osmo-multi.conf}"
_OSMO_MULTI_LOADED=""

_osmo_multi_load() {
    [ -n "$_OSMO_MULTI_LOADED" ] && return "$_OSMO_MULTI_LOADED"
    _OSMO_MULTI_LOADED=1                      # 1 = pas de topologie
    [ -r "$OSMO_MULTI_CONF" ] || return 1
    # shellcheck disable=SC1090
    . "$OSMO_MULTI_CONF" 2>/dev/null || return 1
    [ -n "${MULTI_OPS:-}" ] || return 1
    _OSMO_MULTI_LOADED=0
    return 0
}

# osmo_multi_active - rc 0 si une topologie multi-operateur est declaree.
osmo_multi_active() { _osmo_multi_load; }

# osmo_op_mode <id> - "native" ou "docker" POUR CET OPERATEUR.
# Sans topologie, retombe sur le mode global : le comportement d avant.
osmo_op_mode() {
    local want="${1:-}" spec idx mode
    if _osmo_multi_load; then
        for spec in $MULTI_OPS; do
            IFS=: read -r idx mode _ <<< "$spec"
            [ "$idx" = "$want" ] && { printf '%s\n' "$mode"; return 0; }
        done
    fi
    osmo_mode
}

# osmo_hub_pc - point code du hub, depuis la topologie ; vide sinon.
osmo_hub_pc() {
    _osmo_multi_load || return 1
    [ -n "${MULTI_HUB_PC:-}" ] || return 1
    printf '%s\n' "$MULTI_HUB_PC"
}

# osmo_op_pc <id> - point code attendu, depuis la topologie ; vide sinon.
osmo_op_pc() {
    local want="${1:-}" spec idx mode ip pc
    _osmo_multi_load || return 1
    for spec in $MULTI_OPS; do
        IFS=: read -r idx mode ip pc _ <<< "$spec"
        [ "$idx" = "$want" ] && { printf '%s\n' "$pc"; return 0; }
    done
    return 1
}
```

```
osmo_ops() {
    local ids=""
    # La topologie declaree prime sur toute detection : elle enumere les
    # operateurs ATTENDUS, natif compris - un operateur arrete doit apparaitre
    # dans le bilan comme manquant, pas disparaitre de la liste.
    if _osmo_multi_load; then
        ids="$(tr ' ' '\n' <<<"$MULTI_OPS" | cut -d: -f1 | sed -n 's/^\([0-9][0-9]*\)$/\1/p')"
        [ -n "$ids" ] && { sort -n -u <<<"$ids" | sed '/^$/d'; return 0; }
    fi
    if [ "$(osmo_mode)" = docker ]; then
        local names
        names="$(docker ps --format '{{.Names}}' 2>/dev/null || true)"
        ids="$(sed -n 's/^osmo-operator-\([0-9][0-9]*\)$/\1/p' <<<"$names")"
    else
        ids="$(tr ', ' '\n\n' <<<"${OSMO_OP_IDS:-}" | sed -n 's/^\([0-9][0-9]*\)$/\1/p')"
        if [ -z "$ids" ]; then
            ids="$(ip netns list 2>/dev/null | awk '{print $1}' \
                   | sed -n "s/^${OSMO_NETNS_PREFIX}\([0-9][0-9]*\)\$/\1/p")"
        fi
        if [ -z "$ids" ]; then
            local n; n="$(_osmo_globals N_OPERATORS || true)"
            case "$n" in ''|*[!0-9]*) n=1 ;; esac
            [ "$n" -ge 1 ] 2>/dev/null || n=1
            ids="$(seq 1 "$n")"
        fi
    fi
    [ -n "$ids" ] || return 0
    sort -n -u <<<"$ids" | sed '/^$/d'
}

# osmo_op_exists <id> - l'operateur existe-t-il ?
# Remplace "docker ps | grep osmo-operator-N" de global_check.sh:131, dont le
# motif n'etait pas ancre : "--op=1" y acceptait aussi osmo-operator-10.
osmo_op_exists() {
    local want="${1:-}" id
    case "$want" in ''|*[!0-9]*) return 2 ;; esac
    while read -r id; do
        [ "$id" = "$want" ] && return 0
    done <<<"$(osmo_ops)"
    return 1
}

# osmo_node <op> - etiquette du noeud : "osmo-operator-N" ou "osmo-inter-stp".
# En docker c'est le nom du conteneur ; en natif c'est un NOM D'AFFICHAGE, et
# c'est ce qui preserve mot pour mot la banniere "OPERATEUR 1 (osmo-operator-1)"
# et le champ "container=" du dump.
osmo_node() {
    case "${1:-}" in
        hub|osmo-inter-stp) printf 'osmo-inter-stp\n' ;;
        osmo-operator-*)    printf '%s\n' "$1" ;;
        [0-9]*)             printf 'osmo-operator-%s\n' "$1" ;;
        *)                  printf '%s\n' "${1:-}" ;;
    esac
}

# Numero d'operateur d'un <op> ; vide pour le hub ou pour l'inconnu.
_osmo_id() {
    case "${1:-}" in
        hub|osmo-inter-stp) printf '' ;;
        osmo-operator-*)    printf '%s\n' "${1##*-}" ;;
        [0-9]*)             printf '%s\n' "$1" ;;
        *)                  printf '' ;;
    esac
}
```

```
# osmo_hub - un inter-STP est-il interrogeable ICI ? Ecrit son etiquette si oui.
#   docker : le conteneur osmo-inter-stp tourne.
#   natif  : cette machine EST le hub - OSMO_ROLE=interstp dans /etc/osmo-role
#            (pose par build-iso.sh), ou la sonde de start-interstp.sh:100.
# Sur un noeud operateur natif le hub est une AUTRE machine : rc 1, et c'est
# osmo_hub_hint qui dit quoi faire. Un "0 AS actif" fabrique se lirait comme
# une panne.
osmo_hub() {
    if [ "$(osmo_mode)" = docker ]; then
        local names
        names="$(docker ps --format '{{.Names}}' 2>/dev/null || true)"
        grep -ax 'osmo-inter-stp' >/dev/null 2>&1 <<<"$names" || return 1
        printf 'osmo-inter-stp\n'; return 0
    fi
    if [ -r /etc/osmo-role ] \
       && grep -aE '^OSMO_ROLE[[:space:]]*=[[:space:]]*interstp' /etc/osmo-role >/dev/null 2>&1; then
        printf 'osmo-inter-stp\n'; return 0
    fi
    if pgrep -f 'osmo-stp .*osmo-stp-interop\.cfg' >/dev/null 2>&1; then
        printf 'osmo-inter-stp\n'; return 0
    fi
    return 1
}

# OSMO_HUB_M3UA_PORT - port M3UA du hub inter-STP (helpers/create_interop.sh
# ecrit "listen m3ua 2908"). Les STP operateur, eux, ecoutent sur 2905.
: "${OSMO_HUB_M3UA_PORT:=2908}"

# _osmo_hub_ip_probe - demande aux noeuds operateur ou est leur hub. Deux
# sources, dans cet ordre :
#   1. OSMO_HUB_IP, injecte dans chaque conteneur par start-docker.sh (-e OSMO_HUB_IP)
#      des que --hub-ip / --wan designe un inter-STP hors dorsale docker ;
#   2. le "remote-ip" de l'asp asp-to-inter dans /etc/osmocom/osmo-stp.cfg, qui
#      reste vrai meme sans environnement (conteneur relance a la main, ISO).
# Rend 1 sans rien ecrire si aucun noeud ne sait repondre.
_osmo_hub_ip_probe() {
    local op v=""
    for op in $(osmo_ops 2>/dev/null); do
        v="$(osmo_exec "$op" printenv OSMO_HUB_IP 2>/dev/null | tr -d "\r" | tail -1)"
        [ -n "$v" ] && { printf '%s\n' "$v"; return 0; }
        v="$(osmo_exec "$op" cat /etc/osmocom/osmo-stp.cfg 2>/dev/null \
             | awk '/asp asp-to-inter/{f=1} f&&/remote-ip/{print $2; exit}' | tr -d "\r")"
        [ -n "$v" ] && { printf '%s\n' "$v"; return 0; }
    done
    return 1
}

# osmo_hub_ip - adresse du hub. Rien a dire → rc 1.
#   docker + hub local  : la dorsale, 172.20.0.10 ;
#   docker sans hub local : l'adresse WAN vue par les operateurs (--hub-ip) ;
#   natif : OSMO_HUB_IP de /etc/osmo-role (idiome de wan_ss7_check.sh:193),
#           puis, a defaut, la meme sonde par les noeuds.
# Le resultat est mis en cache : la sonde coute un "docker exec" par noeud, et
# les checks appellent cette fonction plusieurs fois par passage.
_OSMO_HUB_IP_CACHE=""
osmo_hub_ip() {
    if [ -n "$_OSMO_HUB_IP_CACHE" ]; then
        [ "$_OSMO_HUB_IP_CACHE" = "-" ] && return 1
        printf '%s\n' "$_OSMO_HUB_IP_CACHE"; return 0
    fi
    local v=""
    # MULTI_HUB_IP fait autorite : sur un banc mixte, deduire l adresse du hub
    # du mode global menait a la mauvaise (la boucle locale pour un natif,
    # alors que le hub vit dans un conteneur).
    if _osmo_multi_load && [ -n "${MULTI_HUB_IP:-}" ]; then
        _OSMO_HUB_IP_CACHE="$MULTI_HUB_IP"
```

```
        printf '%s\n' "$MULTI_HUB_IP"; return 0
    fi
    if [ "$(osmo_mode)" = docker ]; then
        if osmo_hub >/dev/null 2>&1; then
            printf '%s.10\n' "$OSMO_BACKBONE_NET"; return 0
        fi
    else
        [ -r /etc/osmo-role ] && \
            v="$(awk -F= '/^OSMO_HUB_IP=/ { gsub(/[ \r\t]/, "", $2); print $2 }' /etc/osmo-role 2>/dev/null | tail
             ↪  -1)"
    fi
    [ -n "$v" ] || v="$(_osmo_hub_ip_probe 2>/dev/null || true)"
    if [ -z "$v" ]; then _OSMO_HUB_IP_CACHE="-"; return 1; fi
    _OSMO_HUB_IP_CACHE="$v"
    printf '%s\n' "$v"
}

# osmo_hub_is_remote - rc 0 (et ecrit l'IP) quand un hub est CONFIGURE mais
# vit AILLEURS : c'est le cas WAN (--hub-ip / --wan), ou le noeud operateur
# natif. Une adresse de dorsale docker ou de boucle locale sans conteneur hub
# ne compte pas : la, le hub n'est pas distant, il est simplement absent.
osmo_hub_is_remote() {
    local ip
    osmo_hub >/dev/null 2>&1 && return 1
    ip="$(osmo_hub_ip 2>/dev/null)" || return 1
    [ -n "$ip" ] || return 1
    case "$ip" in
        "${OSMO_BACKBONE_NET}."*|127.*|localhost) return 1 ;;
    esac
    printf '%s\n' "$ip"
}

# osmo_hub_assoc [ip] [port] - combien de noeuds locaux ont une association
# SCTP ETABLIE vers le hub ? Le lien inter-STP est du M3UA sur SCTP : une
# sonde TCP ("nc -z" sur 2908) rend "Connection refused" meme quand le lien
# est parfaitement monte. C'est "ss --sctp" qui dit la verite, et il la dit
# depuis le NOEUD, pas depuis l'hote : le conteneur a sa propre pile reseau.
# Ecrit "etablies/total" ; rc 0 si au moins une, 1 si aucune, 2 si la sonde
# n'est pas praticable (ni ss, ni support --sctp, ni noeud interrogeable).
osmo_hub_assoc() {
    local ip="${1:-}" port="${2:-$OSMO_HUB_M3UA_PORT}" op out probed=0 n=0 tot=0 pat
    [ -n "$ip" ] || ip="$(osmo_hub_ip 2>/dev/null || true)"
    [ -n "$ip" ] || return 2
    pat="${ip//./\\.}:${port}"
    for op in $(osmo_ops 2>/dev/null); do
        tot=$((tot+1))
        out="$(osmo_exec "$op" ss -an --sctp 2>/dev/null || true)"
        [ -n "$out" ] || continue
        probed=1
        grep -aqE "ESTAB.*[[:space:]]${pat}([[:space:]]|$)" <<<"$out" && n=$((n+1))
    done
    [ "$probed" -eq 1 ] || return 2
    printf '%s/%s\n' "$n" "$tot"
    [ "$n" -gt 0 ]
}

# osmo_hub_hint - phrase prete pour skip(), quand le hub existe mais est
# ailleurs. Le VTY Osmocom n'ecoute que sur 127.0.0.1 (aucune conf du depot ne
# pose de "bind" sous "line vty") : depuis un noeud operateur, l'etat des AS
# et des ASP du hub n'est PAS observable. La commande citee existe deja.
osmo_hub_hint() {
    local ip
    ip="$(osmo_hub_ip 2>/dev/null || true)"
    if [ -n "$ip" ]; then
        printf 'hub distant %s : VTY 4239 non joignable depuis ce noeud - lancez ./launch/start-interstp.sh
         ↪  --status sur le hub\n' "$ip"
```

```
    else
        printf 'aucun inter-STP local, et aucune adresse de hub connue ici (OSMO_HUB_IP, --hub-ip) - hub non
        ↪  interrogeable\n'
    fi
}

# ============================================================================
#  3. EXECUTION SUR UN NOEUD
# ============================================================================

# Prefixe netns du mode natif. Vide en mono-operateur : la dorsale et les
# espaces de noms ne sont crees que si N_OPERATORS > 1.
_osmo_ns_cmd() {
    local id ns
    id="$(_osmo_id "${1:-}")"
    [ -n "$id" ] || return 0
    ns="${OSMO_NETNS_PREFIX}${id}"
    if [ -e "/run/netns/$ns" ] || [ -e "/var/run/netns/$ns" ]; then
        printf 'ip netns exec %s' "$ns"
    fi
}

# osmo_exec <op> <argv...> - substitution 1:1 de "docker exec "$container" ...".
osmo_exec() {
    local op="${1:-}"; shift || return 2
    [ $# -ge 1 ] || return 2
    # Le mode se decide operateur par operateur : sur un banc mixte, le natif
    # s interroge en local pendant que ses voisins passent par docker exec.
    if [ "$(osmo_op_mode "$(_osmo_id "$op")")" = docker ]; then
        local c; c="$(osmo_node "$op")"
        [ -n "$c" ] || return 2
        docker exec "$c" "$@"
        return $?
    fi
    local ns; ns="$(_osmo_ns_cmd "$op")"
    # shellcheck disable=SC2086  # ns doit etre decoupe en mots (ip netns exec X)
    if [ -n "$ns" ]; then $ns "$@"; else "$@"; fi
}

# Idem, mais l'entree standard est transmise au noeud ("docker exec -i").
# Interne : reserve au dialogue VTY, ou le tube est l'entree de nc/telnet.
_osmo_exec_in() {
    local op="${1:-}"; shift || return 2
    if [ "$(osmo_op_mode "$(_osmo_id "$op")")" = docker ]; then
        docker exec -i "$(osmo_node "$op")" "$@"
        return $?
    fi
    local ns; ns="$(_osmo_ns_cmd "$op")"
    # shellcheck disable=SC2086
    if [ -n "$ns" ]; then $ns "$@"; else "$@"; fi
}

# osmo_cat <op> <chemin> - contenu d'un fichier du noeud (conf, marqueur...).
osmo_cat() {
    local op="${1:-}" path="${2:-}"
    [ -n "$path" ] || return 2
    if [ "$(osmo_mode)" = docker ]; then
        docker exec "$(osmo_node "$op")" cat "$path" 2>/dev/null
    else
        cat "${OSMO_NATIVE_ROOT}${path}" 2>/dev/null
    fi
}

# osmo_ast <op> <commande CLI> - asterisk -rx, identique dans les deux mondes.
osmo_ast() {
    local op="${1:-}" cli="${2:-}"
```

```
    [ -n "$cli" ] || return 2
    osmo_exec "$op" asterisk -rx "$cli" 2>/dev/null
}

# osmo_running <op> <demon> - ce demon tourne-t-il sur ce noeud ?
#
# En natif il faut les DEUX formes : sur l'ISO les demons sont des units
# (services/osmo-bts-trx.service...) mais core_svc_start retombe sur un lancement
# detache quand systemd est absent - auquel cas seul pgrep repond.
#
# Le repli "pgrep -f" n'est pas decoratif : /proc/PID/comm est TRONQUE a 15
# caracteres, donc "pgrep -ax osmo-sip-connector" (18) ne trouve jamais rien,
# quand bien meme le demon tourne. On retente alors sur la ligne de commande.
#
# Limite assumee du natif multi-operateur : "ip netns exec" n'isole pas les
# PID, la reponse porte donc sur la machine, pas sur l'operateur.
osmo_running() {
    local op="${1:-}" name="${2:-}"
    [ -n "$name" ] || return 2
    if [ "$(osmo_mode)" = docker ]; then
        osmo_exec "$op" pgrep -ax "$name" >/dev/null 2>&1 && return 0
        osmo_exec "$op" pgrep -f "(^|/)${name}([[:space:]]|\$)" >/dev/null 2>&1
        return $?
    fi
    if [ -d /run/systemd/system ] && command -v systemctl >/dev/null 2>&1; then
        systemctl is-active --quiet "$name" 2>/dev/null && return 0
    fi
    pgrep -ax "$name" >/dev/null 2>&1 && return 0
    pgrep -f "(^|/)${name}([[:space:]]|\$)" >/dev/null 2>&1
}

# osmo_sock <op> <chemin> - socket UNIX en ecoute ?
# L'ORDRE EST OBLIGATOIRE : on interroge ss AVANT le test de fichier. Sous
# PrivateTmp, /tmp/msc_mncc EXISTE sans etre visible par "test -S" (constate,
# run_modules/13-msc.sh:14) ; un "test -S" seul donnerait un "voix
# impossible" mensonger.
# awk lit TOUTE son entree (pas d'"exit" au milieu du tube) : sous pipefail,
# un awk qui sort tot ferait mourir ss en SIGPIPE et rendrait 141.
osmo_sock() {
    local op="${1:-}" p="${2:-}" out
    [ -n "$p" ] || return 2
    out="$(osmo_exec "$op" ss -xlH 2>/dev/null || true)"
    awk -v w="$p" '{ for (i = 1; i <= NF; i++) if ($i == w) f = 1 } END { exit !f }' <<<"$out" && return 0
    osmo_exec "$op" test -S "$p" 2>/dev/null
}

# osmo_port <op> <port> [tcp|udp|sctp] - ce port est-il en ecoute sur le noeud ?
# Remplace le "ss -tlnp | grep -ac ':7890'" de global_check.sh:493 : pas de
# tube fragile, et ":7890" ne peut plus se confondre avec 17890 ou 78901.
osmo_port() {
    local op="${1:-}" port="${2:-}" proto="${3:-tcp}" out
    local -a opts
    case "$port" in ''|*[!0-9]*) return 2 ;; esac
    case "$proto" in
        tcp)  opts=(-H -ltn) ;;
        udp)  opts=(-H -lun) ;;
        sctp) opts=(-H -ln --sctp) ;;
        *)    return 2 ;;
    esac
    out="$(osmo_exec "$op" ss "${opts[@]}" 2>/dev/null || true)"
    # Le 4e champ de ss est "adresse:port" local, pour tcp, udp et sctp.
    awk -v want="$port" '
        { a = $4
          if (a == "" || match(a, /:[0-9]+$/) == 0) next
          if (substr(a, RSTART + 1) == want) f = 1 }
        END { exit !f }' <<<"$out"
```

```
}

# osmo_host <op> - l'hote a joindre pour ce noeud DEPUIS ICI (relais SMS, SIP,
# M3UA...). Ce n'est PAS l'hote du VTY : celui-ci est toujours 127.0.0.1 vu de
# l'interieur du noeud, et osmo_vty s'en charge.
#   docker           : l'IP du conteneur sur la dorsale (172.20.0.1N, .10 = hub)
#   natif + netns    : la meme dorsale, recreee en netns
#   natif mono-op    : 127.0.0.1 - il n'y a ni pont gsm-inter ni 172.20.0.0/24
#   hub natif        : OSMO_HUB_IP de /etc/osmo-role, sinon 127.0.0.1
osmo_host() {
    local op="${1:-}" id
    case "$op" in
        hub|osmo-inter-stp)
            if [ "$(osmo_mode)" = docker ]; then
                printf '%s.10\n' "$OSMO_BACKBONE_NET"
            else
                local h; h="$(osmo_hub_ip 2>/dev/null || true)"
                printf '%s\n' "${h:-127.0.0.1}"
            fi
            return 0 ;;
    esac
    id="$(_osmo_id "$op")"
    if [ -n "$id" ] && { [ "$(osmo_mode)" = docker ] || [ -n "$(_osmo_ns_cmd "$op")" ]; }; then
        printf '%s.%s\n' "$OSMO_BACKBONE_NET" "$((10 + id))"
    else
        printf '127.0.0.1\n'
    fi
}

# ============================================================================
#  4. VTY
# ============================================================================

# Quel client VTY ce noeud possede-t-il ? nc de preference, telnet en repli.
# L'ISO installe netcat-openbsd ET telnet (build-iso.sh:693) mais PAS socat :
# ne pas s'appuyer sur socat comme le fait core_vty_ask de run_modules/.
# Le diagnostic est ecrit UNE FOIS par noeud, sur stderr, pour ne pas polluer la
# sortie du check ni la repeter a chaque commande.
_osmo_vty_tool() {
    local op="${1:-}" key tool
    key="$(osmo_node "$op")"
    tool="${_OSMO_VTY_TOOL[$key]:-}"
    if [ -z "$tool" ]; then
        if osmo_exec "$op" sh -c 'command -v nc' >/dev/null 2>&1; then
            tool=nc
        elif osmo_exec "$op" sh -c 'command -v telnet' >/dev/null 2>&1; then
            tool=telnet
        else
            tool=none
            printf 'checks/_mode.sh : aucun client VTY sur %s - installez netcat-openbsd ou telnet (apt-get
             ↪  install -y netcat-openbsd telnet)\n' \
                  "$key" >&2
        fi
        _OSMO_VTY_TOOL[$key]="$tool"
    fi
    [ "$tool" = none ] && return 1
    printf '%s\n' "$tool"
}

# osmo_vty_up <op> <port> - le VTY repond-il ?
# Remplace les cinq copies de "docker exec ... echo >/dev/tcp/127.0.0.1/PORT".
# En natif on essaie d'abord la sonde PASSIVE (ss) : elle n'ouvre ni ne referme
# de session VTY. Si elle ne voit rien (ss absent, /proc restreint), on retombe
# sur la sonde active - pas de faux negatif.
osmo_vty_up() {
    local op="${1:-}" port="${2:-}"
```

```
    case "$port" in ''|*[!0-9]*) return 2 ;; esac
    if [ "$(osmo_mode)" != docker ] && command -v ss >/dev/null 2>&1; then
        osmo_port "$op" "$port" tcp && return 0
    fi
    osmo_exec "$op" bash -c "echo >/dev/tcp/127.0.0.1/${port}" 2>/dev/null
}

# osmo_vty <op> <port> <commande>... - envoie les commandes au VTY, rend la
# sortie BRUTE sur stdout (banniere et invite comprises : c'est osmo_vty_clean
# qui filtre, et les scripts n'ont pas tous le meme filtre).
#
# Rend 1 SANS RIEN ECRIRE si le VTY ne repond pas, ou si le noeud n'a ni nc ni
# telnet : c'est le contrat des vty_cmd() actuelles, que les checks utilisent
# pour distinguer "service muet" de "service sans reponse a cette commande".
# La sonde prealable est celle d'osmo_vty_up (une connexion TCP ouverte puis
# refermee, comme aujourd'hui).
#
# L'appelant fournit ses propres "enable" / "end" : la bibliotheque
# n'invente aucune commande.
#
# LES TEMPORISATIONS SONT DES PARAMETRES, par variables d'environnement, parce
# que chaque script a les siennes et que les changer changerait la QUANTITE de
# sortie capturee - donc les "grep -ac" qui la comptent :
#     OSMO_VTY_OPEN  attente avant d'ecrire (0.5 ; ss7_check : 1)
#     OSMO_VTY_READ  attente de lecture     (2   ; ss7_check : 1.5)
#     OSMO_VTY_Q     nc -q<N>               (1   ; vty-debug-dump : 2)
# Le VTY Osmocom ignore ce qui arrive avant la fin de sa negociation telnet :
# ces sleep cadencent le DIALOGUE, ce ne sont pas des barrieres d'attente.
osmo_vty() {
    local op="${1:-}" port="${2:-}"
    [ $# -ge 3 ] || return 2
    shift 2
    case "$port" in ''|*[!0-9]*) return 2 ;; esac

    osmo_vty_up "$op" "$port" || return 1

    local body="" c
    for c in "$@"; do body="${body}${c}"$'\n'; done

    local tool
    tool="$(_osmo_vty_tool "$op")" || return 1

    local open="${OSMO_VTY_OPEN:-0.5}" read_s="${OSMO_VTY_READ:-2}" q="${OSMO_VTY_Q:-1}"
    # [2026-09-05] LE CLIENT VTY EST BORNE DE FORCE, ET C'EST telnet QUI L'EXIGE.
    # L'image docker des operateurs n'a que telnet (pas netcat-openbsd, que
    # l'ISO installe pour le natif) : _osmo_vty_tool y retombe donc sur telnet.
    # Or telnet, lui, N'A PAS d'equivalent du "-q" de nc : EOF sur son entree ne
    # le fait pas raccrocher, et le VTY Osmocom ne ferme pas non plus apres
    # "end". Resultat mesure : ss7_check.sh restait bloque pour toujours sur le
    # premier operateur en conteneur, donc jamais de matrice de connectivite
    # (l'encart affichait "premiere mesure en cours..." indefiniment) et le
    # bilan de start-multi.sh ne rendait jamais la main sur op2/op3.
    # Le "timeout" part DANS le noeud (docker exec ... timeout N telnet ...) :
    # tuer le docker exec d'ici laisserait le telnet vivant dans le conteneur.
    local kill_after="${OSMO_VTY_KILL:-8}"

    # nc/telnet ferment souvent avant d'avoir tout lu : le sous-shell producteur
    # prend alors un SIGPIPE et le tube rend 141. "|| true" garde la fonction
    # utilisable sous "set -euo pipefail".
    if [ "$tool" = nc ]; then
        { sleep "$open"; printf '%s' "$body"; sleep "$read_s"; } \
            | _osmo_exec_in "$op" timeout "$kill_after" nc "-q${q}" 127.0.0.1 "$port" 2>/dev/null || true
    else
        { sleep "$open"; printf '%s' "$body"; sleep "$read_s"; } \
            | _osmo_exec_in "$op" timeout "$kill_after" telnet 127.0.0.1 "$port" 2>/dev/null || true
    fi
```

```
    return 0
}

# osmo_vty_clean [motif supplementaire] - filtre stdin→stdout : retire banniere,
# invite et lignes vides. Le motif de base et L'ORDRE des filtres sont ceux des
# scripts, au caractere pres : y toucher changerait leur sortie.
# global_check.sh et ss7_check.sh filtrent en plus "Free Software lives", que
# vty-debug-dump.sh garde - d'ou l'argument optionnel :
#     osmo_vty_clean 'Free Software lives'
osmo_vty_clean() {
    local extra="${1:-}"
    local re="^(Trying|Connected|Escape|Welcome|OsmoSTP|OsmoHLR|OsmoMSC|\
OsmoBSC|OsmoBTS|OsmoMGW|OsmoPCU|OsmoSGSN|OsmoGGSN|OsmoSIP|\
VTY server|Use.*help|Press.*tab|[A-Za-z0-9_-]+[#>] |\
Enter password|% Unknown|% Command incomplete|% Error|\
Connection closed)"
    [ -n "$extra" ] && re="${re%)}|${extra})"
    grep -avE "$re" | sed 's/\r//' | grep -av '^[[:space:]]*$' || true
}

# osmo_vty_interactive <op> <port> [hote] - ouvre une session VTY INTERACTIVE
# (un humain devant le clavier), la ou osmo_vty envoie des commandes et rend la
# sortie. Substitution 1:1 de "docker exec -ti <conteneur> telnet <ip> <port>".
# L'hote reste un PARAMETRE parce que les groupes Baseband ecoutent sur
# 127.0.0.1, 127.0.0.2... (tools/vty-menu.sh:79) : c'est la seule cible VTY du
# depot qui ne soit pas 127.0.0.1.
osmo_vty_interactive() {
    local op="${1:-}" port="${2:-}" host="${3:-127.0.0.1}"
    case "$port" in ''|*[!0-9]*) return 2 ;; esac
    if [ "$(osmo_mode)" = docker ]; then
        docker exec -ti "$(osmo_node "$op")" telnet "$host" "$port"
        return $?
    fi
    local ns; ns="$(_osmo_ns_cmd "$op")"
    # shellcheck disable=SC2086  # ns doit etre decoupe en mots (ip netns exec X)
    if [ -n "$ns" ]; then $ns telnet "$host" "$port"; else telnet "$host" "$port"; fi
}

# osmo_vty_target <op> - "cible" a remettre a un client VTY qui N'EST PAS du
# bash et ne peut donc pas sourcer cette bibliotheque.
#
# Le cas existe et il est unique : tools/vty-connect.exp est ecrit en expect. La
# bibliotheque ne l'atteindra jamais ; le mode doit donc etre decide cote bash
# (tools/vty-menu.sh) et lui etre passe en ARGUMENT. La convention est :
#     docker → le nom du conteneur, qui devient "docker exec -ti <nom> telnet"
#     natif  → "-", qui devient un "telnet" direct
# "-" plutot que la chaine vide : une chaine vide se perd dans un [lindex]
# expect comme dans un "$@" bash, et l'on ne distingue plus "natif" de
# "argument oublie".
osmo_vty_target() {
    if [ "$(osmo_mode)" = docker ]; then osmo_node "${1:-}"; else printf -- '-\n'; fi
}

# ============================================================================
#  5. CATEGORIE A - REFUSER PROPREMENT QUAND DOCKER MANQUE
# ============================================================================
#
#  Certains scripts ne se convertissent pas : ils PILOTENT docker par nature -
#  ils construisent une image, creent un reseau 172.20.N.0/24, lancent un
#  conteneur. En natif il n'y a ni image, ni reseau a creer, ni conteneur : le
#  noeud, c'est la machine. Les convertir reviendrait a reecrire start-native.sh
#  et start-interstp.sh, qui existent deja - le depot a deja paye ce prix
#  ailleurs (cf. l'en-tete de start-nitb.sh : 4 copies divergentes de
#  si_bridge.py, 3 de gapk-start.sh).
#
#  Ce qui leur manque, c'est un REFUS. build-iso.sh embarque start-docker.sh,
```

```
#  build.sh et launch/osmo-launch.sh sur une ISO ou build-iso.sh:707 garantit
#  qu'il n'y a PAS de docker : aujourd'hui ces scripts y meurent en 127 sur
#  "docker: command not found", apres avoir deja touche l'hote, sans nommer
#  l'alternative qui, elle, est installee juste a cote.

# osmo_docker_ok - docker est-il REELLEMENT utilisable depuis ici ?
# Le binaire ne suffit pas : une machine de dev l'a souvent sans demon actif, et
# "docker build" y echoue alors sur un message de socket. On interroge le
# demon. Aucun tube : pas de piege pipefail.
osmo_docker_ok() {
    command -v docker >/dev/null 2>&1 || return 1
    docker info >/dev/null 2>&1
}

# osmo_require_docker [ligne d'alternative...] - refus propre, puis exit 3.
# Rend 0 (et ne dit rien) si docker est utilisable : s'appelle donc en tete de
# script sans condition.
#
# Le message suit le gabarit de launch/start-oqc.sh:41-51, le meilleur du
# depot : dire CE QUI MANQUE, POURQUOI ce script ne peut pas s'en passer, et
# SURTOUT quoi lancer a la place, par son nom exact. Sans les alternatives
# fournies en argument, on cite les deux du depot.
#
# Sortie sur stderr, code 3 : distinct de 1 (echec d'execution) et de 2 (mauvais
# argument), pour qu'un appelant puisse reconnaitre "mauvais monde" sans lire
# le texte.
osmo_require_docker() {
    osmo_docker_ok && return 0
    {
        printf '\n%s\n\n' "Docker n'est pas utilisable sur cette machine."
        if command -v docker >/dev/null 2>&1; then
            printf '%s\n' "  Le binaire docker est la, mais le demon ne repond pas."
            printf '%s\n' "  → sudo systemctl start docker, puis relancer ce script."
        else
            printf '%s\n' "  La commande docker est absente - c'est le cas normal sur l'ISO,"
            printf '%s\n' "  en VM et sur une machine nue : la pile y est installee en dur."
        fi
        printf '\n%s\n' "Ce script pilote docker (images, reseaux, conteneurs). Il n'a pas"
        printf '%s\n' "d'equivalent natif a ecrire : l'equivalent EXISTE DEJA, et il a un nom."
        printf '%s\n\n' "Lancez plutot :"
        if [ $# -gt 0 ]; then
            local _l; for _l in "$@"; do printf '  %s\n' "$_l"; done
        else
            printf '  %s\n' "./start-native.sh     - la pile complete de CE noeud (un operateur)"
            printf '  %s\n' "./launch/start-interstp.sh   - le hub SS7 inter-operateurs"
        fi
        printf '\n%s\n' "Sur l'ISO ces deux-la sont deja dans le PATH : osmo-start-direct."
        printf '%s\n\n' "Pour verifier l'etat sans rien demarrer : ./checks/global_check.sh"
    } >&2
    exit 3
}

# ============================================================================
#  6. DIVERS
# ============================================================================

# osmo_qgrep <args...> - "grep -aq" sans le piege.
# "cmd | grep -aq motif" : grep sort a la PREMIERE correspondance, le
# producteur recoit un SIGPIPE et meurt en 141 - et sous "set -o pipefail"
# c'est 141 que rend le tube. "trouve" se lit alors "pas trouve", de facon
# intermittente, selon que la sortie tenait dans le tampon du tube. Ici grep lit
# toute son entree et on jette sa sortie. Copie conforme de qgrep()
# (network/setup-wan-mesh.sh:69).
osmo_qgrep() { grep "$@" >/dev/null; }

# ============================================================================
```

```
#  Execute au lieu d'etre source : on ne fait rien de destructeur, on affiche
#  ce que la bibliotheque VOIT. Sert de test rapide, et evite qu'un futur
#  "for f in checks/*.sh; do $f; done" ne tombe dans le vide.
# ============================================================================
if [ "${BASH_SOURCE[0]}" = "$0" ]; then
    printf 'mode      : %s\n' "$(osmo_mode)"
    printf 'operateurs: %s\n' "$(osmo_ops | tr '\n' ' ')"
    if hub="$(osmo_hub)"; then
        printf 'inter-STP : %s (interrogeable ici)\n' "$hub"
    else
        printf 'inter-STP : %s\n' "$(osmo_hub_hint)"
    fi
    printf 'hote op.1 : %s\n' "$(osmo_host 1)"
    exit 0
fi
```

3.36 /opt/GSM/osmo-operator/checks/annuaire.sh

10571 octets, 202 lignes → 202 lignes

```
#!/bin/bash
# ============================================================================
# checks/annuaire.sh -- l'annuaire du banc, confronte a ce que le reseau en sait
#
# Le banc ne rend que des nombres : 001020002000001, 20001, 1.21.2, rctx 2150.
# Chacun est juste, aucun ne se retient, et il faut trois fenetres pour savoir
# si l'abonne 20001 existe, s'il a une cle, s'il est attache, et sur quelle
# machine. Ce script rassemble les quatre sources en une page :
#
#   data/*.txt  -> annuaire.csv    l'etat civil invente au lancement
#   hlr.db                          l'abonne existe-t-il, a-t-il une cle
#   VTY osmo-msc                    est-il ATTACHE, maintenant
#   annuaire-ss7.csv + VTY stp      quelle machine le sert, et sous quel PC
#
# CE QU'IL MONTRE
#   1. les abonnes : IMSI, MSISDN, nom, operateur, pays, cle, attachement
#   2. les incoherences : une fiche sans abonne, un abonne sans fiche, un
#      abonne sans cle - ce dernier etant le plus couteux, parce qu'il n'a
#      l'air de rien : osmo-msc le trouve, reclame ses vecteurs, n'obtient
#      rien, et rejette le mobile en accusant la carte SIM
#   3. l'annuaire SS7 : quel point code, quel routing context, quel hub, et
#      ce que le STP en dit en ce moment
#
# Ne modifie rien. Ne redemarre rien. Lecture seule.
#
# Usage : ./checks/annuaire.sh [--op N] [--ss7] [--no-color]
#   --op N      un seul operateur (defaut : tous ceux qui tournent)
#   --ss7       la section SS7 seule
#   --abo       la section abonnes seule
# ============================================================================
set -uo pipefail

ONLY_OP=""; WANT_ABO=1; WANT_SS7=1; USE_COLOR=1

# Les fiches ne sont pas au meme endroit selon d'ou l'on regarde : start-docker.sh les
# ecrit sur l'HOTE dans /var/tmp, et en pose une copie dans /etc/osmocom de
# chaque conteneur (elle voyage avec la pile). Lance depuis un conteneur, ce
# script ne trouvait donc rien et annoncait "pas de fiches" sur un banc qui en
# avait. On prend la premiere qui existe.
_premier(){ for f in "$@"; do [ -r "$f" ] && { printf '%s' "$f"; return 0; }; done; printf '%s' "$1"; }
ANNU="${OSMO_ANNUAIRE:-$(_premier /var/tmp/osmo-annuaire.csv /etc/osmocom/annuaire.csv)}"
ANNU_SS7="${OSMO_ANNUAIRE_SS7:-$(_premier /var/tmp/osmo-annuaire-ss7.csv /etc/osmocom/annuaire-ss7.csv)}"

while [ $# -gt 0 ]; do
```

```
  case "$1" in
    --op)       ONLY_OP="${2:-}"; shift ;;
    --op=*)     ONLY_OP="${1#*=}" ;;
    --ss7)      WANT_ABO=0 ;;
    --abo)      WANT_SS7=0 ;;
    --no-color) USE_COLOR=0 ;;
    -h|--help)  sed -n '2,29p' "$0" | sed 's/^# \{0,1\}//'; exit 0 ;;
  esac
  shift
done

if [ "$USE_COLOR" = 1 ] && [ -t 1 ]; then
  G=$'\033[0;32m'; R=$'\033[0;31m'; Y=$'\033[1;33m'; C=$'\033[0;36m'; B=$'\033[1m'; D=$'\033[2m'; Z=$'\033[0m'
else G=; R=; Y=; C=; B=; D=; Z=; fi
OK="${G}[ OK ]${Z}"; KO="${R}[FAIL]${Z}"; WARN="${Y}[WARN]${Z}"
head(){ printf '\n%s== %s ==%s\n' "$B" "$1" "$Z"; }

# ── Ou tourne-t-on ? ──────────────────────────────────────────────
# Sur l'hote on interroge les conteneurs ; dans un conteneur, soi-meme. Le
# meme script sert donc des deux cotes, et c'est voulu : un diagnostic qu'on ne
# peut lancer que d'un seul endroit ne sert qu'a la moitie des pannes.
if [ -f /.dockerenv ]; then
  MODE=local
  IN(){ shift; bash -c "$*" 2>/dev/null; }          # $1 ignore
  OPS="${OPERATOR_ID:-1}"
else
  MODE=docker
  IN(){ local c="$1"; shift; docker exec "$c" bash -c "$*" 2>/dev/null; }
  OPS="$(docker ps --format '{{.Names}}' 2>/dev/null | sed -n 's/^osmo-operator-\([0-9]\+\)$/\1/p' | sort -n | tr
   ↪  '\n' ' ')"
fi
[ -n "$ONLY_OP" ] && OPS="$ONLY_OP"
[ -n "${OPS// /}" ] || { printf '%s aucun operateur en service\n' "$KO"; exit 1; }
ct(){ [ "$MODE" = docker ] && echo "osmo-operator-$1" || echo local; }

# ── Les fiches, lues une fois ─────────────────────────────────────
# Absent n'est pas une erreur : l'annuaire nait au lancement de start-docker.sh, et un
# banc monte a la main n'en a pas. On le dit et on continue - la partie HLR et
# la partie SS7 restent lisibles sans lui.
if [ -r "$ANNU" ]; then
  printf '%s fiches abonnes : %s (%s lignes)\n' "$OK" "$ANNU" "$(( $(wc -l < "$ANNU") - 1 ))"
else
  printf '%s pas de fiches abonnes (%s) - les noms manqueront\n' "$WARN" "$ANNU"
fi
if [ -r "$ANNU_SS7" ]; then
  printf '%s annuaire SS7    : %s (%s lignes)\n' "$OK" "$ANNU_SS7" "$(( $(wc -l < "$ANNU_SS7") - 1 ))"
else
  printf '%s pas d annuaire SS7 (%s)\n' "$WARN" "$ANNU_SS7"
fi

fiche(){   # $1 = msisdn, $2 = champ (7=nom 3=operateur 4=pays 8=adresse 1=imsi)
  [ -r "$ANNU" ] || return 1
  awk -F';' -v n="$1" -v f="$2" 'NR>1 && $2==n {print $f; exit}' "$ANNU"
}

# ═══════════════════════════ 1. LES ABONNES ═══════════════════════════
if [ "$WANT_ABO" = 1 ]; then
for op in $OPS; do
  C_NAME="$(ct "$op")"
  head "ABONNES - operateur ${op} (${C_NAME})"

  # Le HLR fait foi sur l'existence et sur la cle. On lit la base plutot que le
  # VTY : "subscriber show" ne parle que d'un abonne a la fois, et il en faut
  # la liste. La colonne cle vient de auc_2g - un abonne sans ligne la-dedans
  # ne peut pas s'authentifier, quoi qu'en dise sa fiche.
  rows="$(IN "$C_NAME" 'command -v sqlite3 >/dev/null 2>&1 || exit 0
```

```
      sqlite3 -separator "|" /var/lib/osmocom/hlr.db "
        select s.imsi, coalesce(s.msisdn,\"\"),
               case when a.subscriber_id is null then \"non\" else \"oui\" end
        from subscriber s left join auc_2g a on a.subscriber_id = s.id
        order by s.imsi;"')"
    if [ -z "$rows" ]; then
      printf '  %s HLR illisible (sqlite3 absent, ou base vide)\n' "$KO"
      continue
    fi

    # Qui est ATTACHE, maintenant. C'est le cache d'abonnes du VLR : il ne
    # contient que ce qui s'est presente au reseau. Un abonne provisionne mais
    # jamais vu n'y figure pas - et c'est exactement l'information qui manque
    # quand "tout a l'air bon" dans le HLR.
    # "show subscriber all" n'existe PAS sur osmo-msc (il repond "% Unknown
    # command." et la sortie, non vide, passait pour une liste : tout le monde
    # ressortait non attache). Les formes valides sont : cache, imsi, msisdn, tmsi.
    att="$(IN "$C_NAME" 'exec 9<>/dev/tcp/127.0.0.1/4254 2>/dev/null || exit 0
        { printf "enable\n"; printf "show subscriber cache\n"; sleep 0.5; } >&9
        timeout 3 cat <&9' | tr -d '\r' | sed 's/\x1b\[[0-9;]*m//g')"
    case "$att" in *"Unknown command"*) att="" ;; esac

    # Les codes ANSI comptent dans la largeur d'un %-Ns : on remplit d'abord le
    # texte nu, on colore ensuite. Sans cela chaque cellule coloree decale la
    # ligne d'une dizaine de caracteres et la table devient illisible.
    col(){ local t; printf -v t '%-*s' "$2" "$1"; printf '%s%s%s' "${3:-}" "$t" "${Z}"; }
    printf '  %-16s %-7s %-4s %-8s %-20s %-15s %s\n' \
           IMSI MSISDN CLE ETAT NOM OPERATEUR PAYS
    n_sans_cle=0; n_sans_fiche=0
    while IFS='|' read -r imsi msisdn cle; do
      [ -n "$imsi" ] || continue
      nom="$(fiche "$msisdn" 7)"; ope="$(fiche "$msisdn" 3)"; pays="$(fiche "$msisdn" 4)"
      if [ -z "$nom" ]; then nom="(pas de fiche)"; n_sans_fiche=$((n_sans_fiche+1)); nom_c="$D"; else nom_c=""; fi
      if printf '%s' "$att" | grep -q "$imsi"; then etat="attache"; etat_c="$G"; else etat="-"; etat_c="$D"; fi
      if [ "$cle" = non ]; then cle="NON"; cle_c="$R"; n_sans_cle=$((n_sans_cle+1)); else cle="oui"; cle_c="$G"; fi
      printf '  %-16s %-7s %s %s %s %-15s %s\n' \
             "$imsi" "${msisdn:-(aucun)}" \
             "$(col "$cle" 4 "$cle_c")" "$(col "$etat" 8 "$etat_c")" \
             "$(col "$nom" 20 "$nom_c")" "${ope:--}" "${pays:--}"
    done <<< "$rows"

    if [ "$n_sans_cle" -gt 0 ]; then
      printf '  %s %s abonne(s) SANS CLE : osmo-msc les trouve, reclame leurs vecteurs\n' "$KO" "$n_sans_cle"
      printf '         d authentification, n obtient rien, et rejette le mobile en\n'
      printf '         accusant la carte SIM. Reprovisionner : %ssubscriber imsi X update aud2g comp128v1 ki
       ↪  ...%s\n' "$C" "$Z"
    fi
    [ "$n_sans_fiche" -gt 0 ] && \
      printf '  %s %s abonne(s) sans fiche : le HLR les connait, l annuaire non (banc relance sans start-docker.sh
       ↪  ?)\n' "$WARN" "$n_sans_fiche"

    # Le mobile presente-t-il un IMSI que le HLR connait ? C'est la question que
    # personne ne pose, et la reponse est non plus souvent qu'on ne croit : les
    # deux valeurs sont ecrites par deux scripts differents.
    for f in /root/.osmocom/bb/mobile.cfg /root/.osmocom/bb/mobile_faketrx_bts1.cfg; do
      msi="$(IN "$C_NAME" "awk '\$1==\"imsi\"{print \$2; exit}' $f")"
      [ -n "$msi" ] || continue
      if printf '%s\n' "$rows" | cut -d'|' -f1 | grep -qx "$msi"; then
        printf '  %s %-28s presente %s - connu du HLR\n' "$OK" "$(basename "$f")" "$msi"
      else
        printf '  %s %-28s presente %s - INCONNU du HLR : pas d attachement possible\n' "$KO" "$(basename "$f")"
         ↪  "$msi"
      fi
    done
  done
fi
```

```
# ============================== 2. L ANNUAIRE SS7 ==============================
if [ "$WANT_SS7" = 1 ] && [ -r "$ANNU_SS7" ]; then
  head "ANNUAIRE SS7"
  printf '  %-5s %-16s %-12s %-9s %-9s %-9s %-6s %s\n' \
         NOEUD OPERATEUR PAYS PC_MSC PC_STP PC_BSC RCTX HUB
  awk -F';' 'NR>1 {printf "  %-5s %-16s %-12s %-9s %-9s %-9s %-6s %s\n", $1,$2,$3,$6,$7,$8,$12,$13}' "$ANNU_SS7"

  # Et ce que le STP en dit MAINTENANT. Un point code ecrit dans un fichier ne
  # prouve rien : c'est l'AS actif qui prouve que la machine parle.
  for op in $OPS; do
    C_NAME="$(ct "$op")"
    as_out="$(IN "$C_NAME" 'exec 9<>/dev/tcp/127.0.0.1/4239 2>/dev/null || exit 0
        { printf "enable\n"; printf "show cs7 instance 0 as all\n"; sleep 0.5; } >&9
        timeout 3 cat <&9' | tr -d '\r' | sed 's/\x1b\[[0-9;]*m//g')"
    if [ -z "$as_out" ]; then
      printf '  %s operateur %s : VTY STP (4239) muet\n' "$KO" "$op"
      continue
    fi
    act="$(printf '%s\n' "$as_out" | grep -c 'AS_ACTIVE' || true)"
    tot="$(printf '%s\n' "$as_out" | grep -cE '^as-' || true)"
    if [ "${act:-0}" -gt 0 ]; then
      printf '  %s operateur %s : %s/%s AS actifs\n' "$OK" "$op" "$act" "$tot"
    else
      printf '  %s operateur %s : aucun AS actif (%s declares)\n' "$KO" "$op" "$tot"
    fi
  done
fi

exit 0
```

3.37 /opt/GSM/osmo-operator/checks/check_all.sh

24033 octets, 484 lignes → 484 lignes

```
#!/bin/bash
# ============================================================================
# checks/check_all.sh -- lance TOUS les checks du banc, dans le bon ordre,
#                        et rend UN verdict.
#
# POURQUOI CE FICHIER EXISTE
# --------------------------
# checks/ contient huit scripts qui ne se recouvrent pas : global_check voit
# les demons, ss7_check voit la signalisation, diag-* voit l'interco d'un cote
# ou de l'autre, wan_ss7_check voit ce qui passe entre machines, annuaire voit
# les abonnes. Pris un par un ils sont justes ; pris a la main ils sont
# oublies, lances dans le desordre, ou lances sur le mauvais noeud - et le
# diagnostic finit par dependre de qui tape la commande.
#
# Ce script tape la commande. Il choisit CE QUI A UN SENS ICI (le mode et le
# role decident), execute dans l'ordre du plus general au plus fin, garde tout
# dans un journal, et termine par un tableau : une ligne par check, un verdict,
# un code de retour.
#
# CE QU'IL NE FAIT PAS
#   - il ne modifie rien, ne redemarre rien : tous les checks appeles sont en
#     lecture seule (seul --dump touche au niveau de log des demons, et il est
#     donc optionnel, jamais par defaut) ;
#   - il ne reimplemente aucun test : chaque verdict vient du script d'origine,
#     jamais d'une seconde lecture des VTY faite ici.
#
# CHOIX DES CHECKS - CE QUI EST LANCE, ET QUAND
#   mode           tous les checks respectent --docker/--native ; l'option est
#                  transmise telle quelle, donc un seul monde est regarde.
```

```
#   diag-interstp  seulement si un inter-STP est INTERROGEABLE ICI (conteneur
#                  osmo-inter-stp, ou OSMO_ROLE=interstp) : sur un noeud
#                  operateur le hub est une autre machine, son VTY n'ecoute que
#                  sur sa boucle locale, et le lancer produirait un "hub muet"
#                  qui se lit comme une panne.
#   diag-stp-oper. l'exact symetrique : seulement quand ce noeud n'est PAS le
#                  hub et qu'il a un /etc/osmo-role (donc un montage WAN).
#   wan_ss7_check  seulement si /etc/osmo-wan.conf existe : sans plan WAN il
#                  n'y a pas de "entre noeuds" a verifier.
#   annuaire       seulement si les fiches existent (var/tmp ou /etc/osmocom).
#   dump + resume  jamais par defaut : vty-debug-dump.sh passe les demons en
#                  DEBUG et ecrit plusieurs Mo. --dump les active tous les deux,
#                  le resume relisant le dump que le premier vient d'ecrire.
#
# COMMENT EST RENDU LE VERDICT D'UN CHECK
# Les scripts de checks/ n'ont pas tous la meme convention, et on ne la leur
# impose pas apres coup :
#   rc    global_check.sh / ss7_check.sh rendent le NOMBRE d'echecs ;
#   rcb   wan_ss7_check.sh rend 0 ou 1 ;
#   motif diag-*.sh et annuaire.sh rendent toujours 0 - leur verdict est dans
#         leur derniere section ("INTERCO SS7 : DOWN"), on l'y lit, sur la
#         sortie DEBARRASSEE des couleurs.
#
# Usage :
#   sudo ./checks/check_all.sh [options]
#     --quick|-q        transmis a global_check / ss7_check (passage rapide)
#     --verbose|-v      transmis aux checks qui le comprennent
#     --op=N            limiter a un operateur (global_check, annuaire)
#     --docker|--native forcer le monde ; sans option, detection automatique
#     --dump            ajouter vty-debug-dump.sh puis operator_summary.sh
#     --only=a,b        ne lancer que ces checks (voir --list)
#     --skip=a,b        lancer tous les checks sauf ceux-la
#     --list            afficher les checks, leur cle et leur applicabilite
#     --out FICHIER     journal complet (defaut /tmp/osmo-check-all-<date>.txt)
#     --no-log          ne rien ecrire sur disque
#     --timeout N       delai maximum par check, en secondes (defaut 300)
#     --no-color        sortie sans couleur
#
# Code de retour : nombre de checks en echec (0 = tout va bien), 125 au plus.
# ============================================================================

set -u

# -- Bibliotheque commune (detection docker/natif, inventaire, VTY) ------------
# Chemin RELATIF a CE fichier : le depot vit en /home/.../osmo-operator pendant le
# developpement et en /opt/GSM/osmo-operator sur l'ISO ; un chemin absolu casserait
# l'un des deux.
_here="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
for _c in "$_here/_mode.sh" /opt/GSM/osmo-operator/checks/_mode.sh; do
    [ -r "$_c" ] && { . "$_c"; break; }
done
command -v osmo_mode >/dev/null || { echo "checks/_mode.sh introuvable" >&2; exit 1; }

# -- Options -------------------------------------------------------------------
QUICK=0; VERBOSE=0; SPECIFIC_OP=""; WANT_DUMP=0
ONLY=""; SKIP_LIST=""; LIST_ONLY=0
LOGFILE=""; NO_LOG=0; TIMEOUT=300; USE_COLOR=1
MODE_FORCED=0; MODE_OPT=""

# OSMO_MODE pose dans l'environnement est deja une forme de forcage
# ("OSMO_MODE=native ./checks/check_all.sh") : on le signale comme tel, et on
# le transmet aux fils sous forme d'option pour qu'aucun d'eux ne redetecte.
case "${OSMO_MODE:-}" in docker|native) MODE_FORCED=1 ;; esac

while [ $# -gt 0 ]; do
    case "$1" in
```

```
        --quick|-q)   QUICK=1 ;;
        --verbose|-v) VERBOSE=1 ;;
        --op)         SPECIFIC_OP="${2:-}"; shift ;;
        --op=*)       SPECIFIC_OP="${1#*=}" ;;
        --dump)       WANT_DUMP=1 ;;
        --only)       ONLY="${2:-}"; shift ;;
        --only=*)     ONLY="${1#*=}" ;;
        --skip)       SKIP_LIST="${2:-}"; shift ;;
        --skip=*)     SKIP_LIST="${1#*=}" ;;
        --list)       LIST_ONLY=1 ;;
        --out)        LOGFILE="${2:-}"; shift ;;
        --out=*)      LOGFILE="${1#*=}" ;;
        --no-log)     NO_LOG=1 ;;
        --timeout)    TIMEOUT="${2:-}"; shift ;;
        --timeout=*)  TIMEOUT="${1#*=}" ;;
        --no-color)   USE_COLOR=0 ;;
        --docker|--native|--mode=*)
            osmo_mode_force "$1" || { echo "mode inconnu : $1" >&2; exit 1; }
            MODE_FORCED=1 ;;
        -h|--help)    sed -n '2,69p' "$0" | sed 's/^# \{0,1\}//'; exit 0 ;;
        *)            echo "option inconnue : $1 (voir --help)" >&2; exit 1 ;;
    esac
    shift
done

case "$TIMEOUT" in ''|*[!0-9]*) echo "--timeout attend un entier" >&2; exit 1 ;; esac

# ── Couleurs ──────────────────────────────────────────────────────────
if [ "$USE_COLOR" = 1 ] && [ -t 1 ]; then
    GREEN=$'\033[0;32m'; CYAN=$'\033[0;36m'; YELLOW=$'\033[1;33m'
    RED=$'\033[0;31m';   BOLD=$'\033[1m';    DIM=$'\033[2m'; NC=$'\033[0m'
else
    GREEN=; CYAN=; YELLOW=; RED=; BOLD=; DIM=; NC=
fi

banner() {
    echo ""
    echo "${BOLD}$*${NC}"
    printf '%s\n' "$(printf '─%.0s' $(seq 1 74))"
}

# ── Mode, role, inventaire ────────────────────────────────────────────
MODE="$(osmo_mode)"
[ "$MODE_FORCED" -eq 1 ] && MODE_SRC="force" || MODE_SRC="detecte"

# L'option de mode transmise aux fils. On la transmet TOUJOURS, meme quand elle
# a ete detectee ici : sans elle chaque fils refait la detection, et deux
# detections successives sur un hote qui a docker installe mais fait tourner le
# lab en natif peuvent ne pas tomber d'accord. Un rapport dont les moities
# regardent deux mondes differents n'est pas un rapport.
MODE_OPT="--$MODE"

HUB_LOCAL=0; HUB_LABEL=""
if HUB_LABEL="$(osmo_hub 2>/dev/null)"; then HUB_LOCAL=1; else HUB_LABEL=""; fi

ROLE=""
[ -r /etc/osmo-role ] && \
    ROLE="$(awk -F= '/^OSMO_ROLE=/ { gsub(/[ \r\t]/, "", $2); print $2 }' /etc/osmo-role 2>/dev/null | tail -1)"

OPS="$(osmo_ops 2>/dev/null | tr '\n' ' ' | sed 's/ *$//')"
N_OPS="$(printf '%s\n' "$OPS" | wc -w | tr -d ' ')"

# Les fiches d'abonnes ne sont pas au meme endroit selon d'ou l'on regarde :
# start-docker.sh les ecrit sur l'HOTE dans /var/tmp et en pose une copie dans
# /etc/osmocom de chaque conteneur. Meme regle que checks/annuaire.sh.
ANNU_FOUND=0
```

```
for f in "${OSMO_ANNUAIRE:-}" /var/tmp/osmo-annuaire.csv /etc/osmocom/annuaire.csv; do
    [ -n "$f" ] && [ -r "$f" ] && { ANNU_FOUND=1; break; }
done

# ── Journal ──────────────────────────────────────────────────────────
# Le journal recoit la sortie SANS les couleurs : un rapport qu'on envoie par
# mail ou qu'on relit six mois plus tard ne doit pas etre truffe d'echappements.
# Le terminal, lui, garde les siennes.
STRIP_ANSI='s/\x1b\[[0-9;]*[a-zA-Z]//g'
if [ "$NO_LOG" -eq 0 ]; then
    [ -n "$LOGFILE" ] || LOGFILE="/tmp/osmo-check-all-$(date '+%Y%m%d-%H%M%S').txt"
    if : >"$LOGFILE" 2>/dev/null; then :; else
        echo "${YELLOW}journal impossible a ecrire (${LOGFILE}) - on continue sans${NC}" >&2
        NO_LOG=1; LOGFILE=""
    fi
else
    LOGFILE=""
fi
journal() { [ -n "$LOGFILE" ] && printf '%s\n' "$*" | sed "$STRIP_ANSI" >>"$LOGFILE"; return 0; }

# ── Le catalogue ─────────────────────────────────────────────────────
# Un enregistrement par check :  cle | script | verdict | description
#   verdict = rc     le script rend le NOMBRE d'echecs (global_check, ss7_check)
#             rcb    le script rend 0 ou non-0 (wan_ss7_check)
#             motif  le script rend toujours 0 : le verdict se lit dans sa
#                    sortie, motif donne apres ':' (diag-*, annuaire)
#             info   sortie informative, jamais en echec (dump, resume)
# L'ordre est celui du diagnostic : d'abord ce qui tourne, puis la
# signalisation locale, puis l'interco, puis le WAN, puis les abonnes.
CATALOGUE=(
  "global|global_check.sh|rc|etat de tous les demons (STP, HLR, MSC, BSC, BTS, PCU, SGSN, GGSN, MGW)"
  "ss7|ss7_check.sh|rc|signalisation SS7 : AS, ASP, routes, catch-all vers l inter-STP"
  "interstp|diag-interstp.sh|motif:INTERCO SS7 : (DOWN|PARTIELLE)|indeterminable|interco vue du HUB : qui est
   ↪  attache, qui manque"
  "operator|diag-stp-operator.sh|motif:INTERCO SS7 : DOWN|interco vue du NOEUD : conf disque vs demon, SCTP vers
   ↪  le hub"
  "wan|wan_ss7_check.sh|rcb|entre noeuds : SIP/RTP, relais SMS, point codes, iptables"
  "annuaire|annuaire.sh|motif:\\[FAIL\\]|abonnes : IMSI, MSISDN, cles Ki, attachement, annuaire SS7"
  "dump|vty-debug-dump.sh|info|dump DEBUG de tous les VTY (--dump seulement)"
  "resume|operator_summary.sh|info|resume par operateur, lu sur le dump (--dump seulement)"
)

champ() { printf '%s' "$1" | cut -d'|' -f"$2"; }
# Le motif peut contenir des '|' (alternative regex) : il occupe donc tout ce
# qui est entre le 3e champ et le DERNIER, la description etant le dernier.
verdict_de()     { printf '%s' "${1#*|*|}" | sed 's/|[^|]*$//'; }
description_de() { printf '%s' "$1" | awk -F'|' '{print $NF}'; }

# applicable <cle> - ce check a-t-il un sens ici ? Ecrit la raison si non.
applicable() {
    case "$1" in
        global|ss7) return 0 ;;
        interstp)
            [ "$HUB_LOCAL" -eq 1 ] && return 0
            printf 'aucun inter-STP interrogeable ici (%s)' "$(osmo_hub_hint 2>/dev/null || echo 'hub distant ou
             ↪  absent')"
            return 1 ;;
        operator)
            [ "$HUB_LOCAL" -eq 1 ] && { printf 'ce noeud EST le hub - voir le check interstp'; return 1; }
            [ -r /etc/osmo-role ] && return 0
            printf '/etc/osmo-role absent : pas de montage WAN sur ce noeud'
            return 1 ;;
        wan)
            [ -r /etc/osmo-wan.conf ] && return 0
            printf '/etc/osmo-wan.conf absent : aucun plan WAN a verifier'
            return 1 ;;
```

```
        annuaire)
            [ "$ANNU_FOUND" -eq 1 ] && return 0
            printf 'aucune fiche (osmo-annuaire.csv) : banc lance sans annuaire'
            return 1 ;;
        dump|resume)
            [ "$WANT_DUMP" -eq 1 ] && return 0
            printf 'non demande (--dump)'
            return 1 ;;
    esac
    return 0
}

# args_de <cle> - les options a passer, chaque script n'ayant pas les memes.
args_de() {
    local a=("$MODE_OPT")
    case "$1" in
        global)
            [ "$QUICK" -eq 1 ]        && a+=(--quick)
            [ "$VERBOSE" -eq 1 ]      && a+=(--verbose)
            [ -n "$SPECIFIC_OP" ]     && a+=("--op=$SPECIFIC_OP") ;;
        ss7)
            [ "$QUICK" -eq 1 ]        && a+=(--quick)
            [ "$VERBOSE" -eq 1 ]      && a+=(--verbose) ;;
        interstp)
            # diag-interstp.sh et diag-stp-operator.sh sont anterieurs a
            # _mode.sh : ils lisent /etc/osmo-role et les VTY locaux, et ne
            # connaissent ni --docker ni --native. Leur passer MODE_OPT les
            # ferait sortir en "option inconnue".
            a=()
            [ "$USE_COLOR" -eq 0 ]    && a+=(--no-color) ;;
        operator)
            a=()
            [ "$USE_COLOR" -eq 0 ]    && a+=(--no-color) ;;
        wan)
            a=()
            [ "$VERBOSE" -eq 1 ]      && a+=(--verbose) ;;
        annuaire)
            a=()
            [ -n "$SPECIFIC_OP" ]     && a+=(--op "$SPECIFIC_OP")
            [ "$USE_COLOR" -eq 0 ]    && a+=(--no-color) ;;
        dump)
            a+=(--out "$DUMP_OUT") ;;
        resume)
            a+=("--dump=$DUMP_OUT") ;;
    esac
    printf '%s\n' "${a[@]:-}"
}

# ── --list : dire ce qui serait fait, sans rien faire ──────────────────
if [ "$LIST_ONLY" -eq 1 ]; then
    echo "${BOLD}checks disponibles${NC}   (mode ${MODE}, ${MODE_SRC})"
    echo ""
    printf '  %-10s %-24s %s\n' CLE SCRIPT ETAT
    printf '  %s\n' "$(printf '—%.0s' $(seq 1 72))"
    for e in "${CATALOGUE[@]}"; do
        k="$(champ "$e" 1)"; s="$(champ "$e" 2)"
        if raison="$(applicable "$k")"; then
            printf '  %-10s %-24s %slance%s\n' "$k" "$s" "$GREEN" "$NC"
        else
            printf '  %-10s %-24s %signore%s - %s\n' "$k" "$s" "$DIM" "$NC" "$raison"
        fi
        printf '  %-10s %-24s %s%s%s\n' "" "" "$DIM" "$(description_de "$e")" "$NC"
    done
    echo ""
    exit 0
fi
```

```
# ── En-tete ──────────────────────────────────────────────────────────────
DUMP_OUT="/tmp/osmo-check-all-dump-$(date '+%Y%m%d-%H%M%S').txt"
DATE_DEB="$(date '+%Y-%m-%d %H:%M:%S')"

en_tete() {
    echo "${CYAN}${BOLD}"
    echo "╔═══════════════════════════════════════════════════════════════════════════╗"
    echo "║   osmo-operator - CHECK COMPLET du banc                                   ║"
    echo "╚═══════════════════════════════════════════════════════════════════════════╝"
    echo "${NC}"
    printf '  date       : %s\n' "$DATE_DEB"
    printf '  machine    : %s\n' "$(hostname 2>/dev/null || echo '?')"
    printf '  mode       : %s%s%s (%s)\n' "$CYAN" "$MODE" "$NC" "$MODE_SRC"
    printf '  operateurs : %s   (%s)\n' "${N_OPS:-0}" "${OPS:-aucun}"
    if [ "$HUB_LOCAL" -eq 1 ]; then
        printf '  inter-STP  : %s%s%s (interrogeable ici)\n' "$GREEN" "$HUB_LABEL" "$NC"
    else
        printf '  inter-STP  : %s\n' "$(osmo_hub_hint 2>/dev/null || echo 'non interrogeable ici')"
    fi
    [ -n "$ROLE" ] && printf '  role       : %s\n' "$ROLE"
    [ -n "$LOGFILE" ] && printf '  journal    : %s\n' "$LOGFILE"
    # Les VTY Osmocom ecoutent sur la boucle locale et docker exige le socket :
    # sans droits, la moitie des checks rendront "muet" pour une raison qui n'a
    # rien a voir avec le reseau. On le dit AVANT, pas apres.
    [ "$(id -u)" -ne 0 ] && printf '  %s! lance sans root : certains checks seront incomplets (sudo
     ↪  recommande)%s\n' "$YELLOW" "$NC"
}
en_tete
en_tete_txt="$(en_tete 2>&1)"
journal "$en_tete_txt"

# ── Execution ────────────────────────────────────────────────────────────
NB_OK=0; NB_KO=0; NB_IGN=0
RES_CLE=(); RES_ETAT=(); RES_RC=(); RES_DUREE=(); RES_NOTE=()

dans_liste() {   # dans_liste <aiguille> <a,b,c>
    local x; local IFS=','
    for x in $2; do [ "$x" = "$1" ] && return 0; done
    return 1
}

enregistre() {   # enregistre <cle> <etat> <rc> <duree> <note>
    RES_CLE+=("$1"); RES_ETAT+=("$2"); RES_RC+=("$3"); RES_DUREE+=("$4"); RES_NOTE+=("${5:-}")
}

for entree in "${CATALOGUE[@]}"; do
    cle="$(champ "$entree" 1)"
    script="$(champ "$entree" 2)"
    verdict="$(verdict_de "$entree")"
    desc="$(description_de "$entree")"
    chemin="$_here/$script"

    # --only / --skip d'abord : ce que l'utilisateur a exclu n'est meme pas
    # evalue, et n'apparait pas comme "ignore par le banc" - la distinction
    # compte, l'un est un choix, l'autre un constat.
    [ -n "$ONLY" ]      && { dans_liste "$cle" "$ONLY"      || continue; }
    [ -n "$SKIP_LIST" ] && { dans_liste "$cle" "$SKIP_LIST" && continue; }

    if ! raison="$(applicable "$cle")"; then
        banner "[$cle] $script"
        echo "  ${DIM}ignore${NC} - $raison"
        journal ""; journal "== [$cle] $script =="; journal "  ignore - $raison"
        NB_IGN=$((NB_IGN+1)); enregistre "$cle" IGNORE "-" "-" "$raison"
        continue
    fi
```

```
if [ ! -x "$chemin" ]; then
    banner "[$cle] $script"
    echo "  ${RED}absent ou non executable${NC} : $chemin"
    journal ""; journal "== [$cle] $script =="; journal "  absent : $chemin"
    NB_KO=$((NB_KO+1)); enregistre "$cle" ABSENT "-" "-" "$chemin introuvable"
    continue
fi

mapfile -t opts < <(args_de "$cle")
# args_de rend une ligne vide quand il n'y a aucune option : le tableau
# contiendrait alors un argument vide, que certains scripts refusent.
[ "${#opts[@]}" -eq 1 ] && [ -z "${opts[0]}" ] && opts=()

banner "[$cle] $script ${DIM}${opts[*]:-}${NC}"
echo "  ${DIM}${desc}${NC}"
echo ""
journal ""
journal "═══════════════════════════════════════════════════════════════════════════"
journal "== [$cle] $script ${opts[*]:-}"
journal "   $desc"
journal "═══════════════════════════════════════════════════════════════════════════"

tmp="$(mktemp "${TMPDIR:-/tmp}/osmo-check-$cle-XXXXXX")"
debut=$SECONDS
# timeout : un VTY qui ne repond jamais (demon fige, docker exec suspendu)
# bloquerait tout le rapport. On coupe, on le dit, on continue.
# PIPESTATUS et non "set -o pipefail" : sous pipefail un "grep -q" en fin
# de tube, dans l'un des fils, rendrait 141 (voir checks/_mode.sh).
timeout "$TIMEOUT" "$chemin" "${opts[@]}" 2>&1 | tee "$tmp"
rc="${PIPESTATUS[0]}"
duree=$((SECONDS - debut))

[ -n "$LOGFILE" ] && sed "$STRIP_ANSI" "$tmp" >>"$LOGFILE"
propre="$(sed "$STRIP_ANSI" "$tmp")"
rm -f "$tmp"

etat=OK; note=""
if [ "$rc" -eq 124 ]; then
    etat=TIMEOUT; note="coupe apres ${TIMEOUT}s"
else
    case "$verdict" in
        rc)
            # global_check.sh et ss7_check.sh rendent le nombre d'echecs.
            if [ "$rc" -ne 0 ]; then etat=ECHEC; note="$rc test(s) en echec"; fi ;;
        rcb)
            if [ "$rc" -ne 0 ]; then etat=ECHEC; note="code de retour $rc"; fi ;;
        motif:*)
            # Le script rend toujours 0 : son verdict est dans sa sortie.
            # On retient la LIGNE entiere, pas le motif : "[FAIL]" tout seul
            # dans le bilan n'apprend rien, "[FAIL] operateur 1 : VTY STP
            # (4239) muet" dispense d'aller rouvrir le journal.
            m="${verdict#motif:}"
            nb="$(printf '%s\n' "$propre" | grep -Ec -- "$m")"
            if [ "${nb:-0}" -gt 0 ]; then
                etat=ECHEC
                note="$(printf '%s\n' "$propre" | grep -Em1 -- "$m" | sed 's/^[[:space:]]*//;
                 ↪  s/[[:space:]]*$//' | cut -c1-60)"
                [ "$nb" -gt 1 ] && note="$note (+$((nb-1)))"
            fi ;;
        info)
            etat=INFO ;;
    esac
fi

case "$etat" in
```

```
        OK)      NB_OK=$((NB_OK+1));  echo ""; echo "  ${GREEN}→ $cle : OK${NC} (${duree}s)" ;;
        INFO)    NB_OK=$((NB_OK+1));  echo ""; echo "  ${CYAN}→ $cle : termine${NC} (${duree}s)" ;;
        TIMEOUT) NB_KO=$((NB_KO+1));  echo ""; echo "  ${RED}→ $cle : TIMEOUT${NC} - $note" ;;
        *)       NB_KO=$((NB_KO+1));  echo ""; echo "  ${RED}→ $cle : PROBLEMES${NC} - $note (${duree}s)" ;;
    esac
    journal "  → $cle : $etat ${note} (${duree}s, rc=$rc)"

    enregistre "$cle" "$etat" "$rc" "${duree}s" "$note"
done

# ── Bilan ──────────────────────────────────────────────────────────────
bilan() {
    echo ""
    printf '%s\n' "$(printf '=%.0s' $(seq 1 74))"
    echo "  ${BOLD}BILAN - $(date '+%Y-%m-%d %H:%M:%S')${NC}"
    printf '%s\n' "$(printf '=%.0s' $(seq 1 74))"
    printf '  %-10s %-9s %-6s %-7s %s\n' CHECK ETAT RC DUREE COMMENTAIRE
    printf '  %s\n' "$(printf '-%.0s' $(seq 1 72))"
    local i coul
    for i in "${!RES_CLE[@]}"; do
        case "${RES_ETAT[$i]}" in
            OK)                coul="$GREEN" ;;
            INFO)              coul="$CYAN" ;;
            IGNORE)            coul="$DIM" ;;
            ECHEC|TIMEOUT|ABSENT) coul="$RED" ;;
            *)                 coul="$YELLOW" ;;
        esac
        printf '  %-10s %s%-9s%s %-6s %-7s %s\n' \
            "${RES_CLE[$i]}" "$coul" "${RES_ETAT[$i]}" "$NC" \
            "${RES_RC[$i]}" "${RES_DUREE[$i]}" "${RES_NOTE[$i]}"
    done
    printf '  %s\n' "$(printf '-%.0s' $(seq 1 72))"
    printf '  %s%d ok%s   %s%d en echec%s   %s%d ignore%s   (mode %s)\n' \
        "$GREEN" "$NB_OK" "$NC" "$RED" "$NB_KO" "$NC" "$DIM" "$NB_IGN" "$NC" "$MODE"
    echo ""
    if [ "$NB_KO" -eq 0 ] && [ "$NB_OK" -eq 0 ]; then
        # Tout ignore ou tout exclu : dire "BANC OK" ici serait un mensonge par
        # omission - rien n'a ete regarde.
        echo "  ${YELLOW}${BOLD}AUCUN CHECK N A TOURNE${NC} - tout etait ignore ou exclu (--list pour savoir
        ↪  pourquoi)"
    elif [ "$NB_KO" -eq 0 ]; then
        echo "  ${GREEN}${BOLD}BANC OK${NC} - ${NB_OK} check(s) passes, aucun en echec"
    else
        echo "  ${RED}${BOLD}PROBLEMES DETECTES${NC} - ${NB_KO} check(s) en echec"
        echo ""
        echo "  ${DIM}rejouer un seul check en detail :${NC}"
        for i in "${!RES_CLE[@]}"; do
            case "${RES_ETAT[$i]}" in
                ECHEC|TIMEOUT) echo "    ./checks/check_all.sh --only=${RES_CLE[$i]} --verbose" ;;
            esac
        done
    fi
    [ -n "$LOGFILE" ] && { echo ""; echo "  ${CYAN}journal complet :${NC} $LOGFILE"; }
    [ "$WANT_DUMP" -eq 1 ] && [ -s "$DUMP_OUT" ] && echo "  ${CYAN}dump VTY        :${NC} $DUMP_OUT"
    printf '%s\n' "$(printf '=%.0s' $(seq 1 74))"
    echo ""
}
bilan
journal "$(bilan)"

# Meme convention que global_check.sh / ss7_check.sh : le code de retour est le
# nombre d'echecs. Plafonne a 125 - au-dela, un code de sortie shell change de
# sens (126, 127 et 128+n sont reserves).
[ "$NB_KO" -gt 125 ] && NB_KO=125
exit "$NB_KO"
```

3.38 /opt/GSM/osmo-operator/checks/diag-interstp.sh

12337 octets, 259 lignes → 259 lignes

```
#!/bin/bash
# ============================================================================
# checks/diag-interstp.sh -- diagnostic SS7 vu du HUB (inter-STP)
#
# Repond a une seule question : QUI est attache, et qui manque ?
# L'identite de ce qui est attache vient du hub lui-meme (colonne AS Name et
# adresse distante de "show cs7 instance 0 asp"). La table /etc/osmo-wan.conf
# ne sert plus qu'a l'appel des absents : un noeud absent de la table reste
# indiscernable d'un noeud eteint.
#
# CE QU'IL VERIFIE
#   1. role et adresse d'ecoute (le multi-homing 0.0.0.0 est signale)
#   2. le demon tourne, le port M3UA ecoute
#   3. chaque AS declare par le hub : AS_ACTIVE / AS_DOWN, et les ASP rattaches
#   4. l'appel de la table WAN : quelle adresse attendue n'est vue nulle part
#   5. les routes : PROHIB / UNAVAIL signales
#   6. le flapping : deux mesures espacees, les ASP dynamiques renumerotes
#      trahissent des associations qui se reetablissent en boucle
#
# Ne modifie rien. Ne redemarre rien. Lecture seule.
#
# Usage : sudo ./checks/diag-interstp.sh [--watch] [--no-color]
#   --watch     seconde mesure a +10 s pour detecter le flapping
# ============================================================================
set -uo pipefail

ROLE_FILE=/etc/osmo-role
WAN_FILE=/etc/osmo-wan.conf
CFG=/etc/osmocom/osmo-stp-interop.cfg
# Le hub peut avoir ete pousse hors de 4239 (co-localise avec le STP
# operateur, cf. start-interstp.sh) : relire son VTY dans sa config.
VTY_PORT=4239
_hubp="$(sed -n 's/^ *bind 127\.0\.0\.1 \([0-9]*\)$/\1/p' "$CFG" 2>/dev/null | head -1)"
[ -n "$_hubp" ] && VTY_PORT="$_hubp"
M3UA_PORT=2908
WATCH=0; USE_COLOR=1

while [ $# -gt 0 ]; do
  case "$1" in
    --watch)    WATCH=1 ;;
    --no-color) USE_COLOR=0 ;;
    -h|--help)  sed -n '2,23p' "$0" | sed 's/^# \{0,1\}//'; exit 0 ;;
  esac
  shift
done

if [ "$USE_COLOR" = 1 ] && [ -t 1 ]; then
  G=$'\033[0;32m'; R=$'\033[0;31m'; Y=$'\033[1;33m'; C=$'\033[0;36m'; B=$'\033[1m'; Z=$'\033[0m'
else G=; R=; Y=; C=; B=; Z=; fi
OK="${G}[ OK ]${Z}"; KO="${R}[FAIL]${Z}"; WARN="${Y}[WARN]${Z}"
head(){ printf '\n%s== %s ==%s\n' "$B" "$1" "$Z"; }

vty(){ { printf 'enable\n'; for c in "$@"; do printf '%s\n' "$c"; done; sleep 0.6; } \
       | timeout 8 nc -q1 127.0.0.1 "$VTY_PORT" 2>/dev/null | tr -d '\r' | sed 's/\x1b\[[0-9;]*m//g'; }

getv(){ awk -F= -v k="$1" '$1==k{gsub(/[ \r\t]/,"",$2);v=$2} END{print v}' "$ROLE_FILE" 2>/dev/null; }

# ---- identite ----------------------------------------------------------
head "IDENTITE"
ROLE="$(getv OSMO_ROLE)"; SELF="$(getv OSMO_HUB_IP)"
printf '  hostname : %s\n' "$(hostname)"
printf '  role     : %s%s%s   adresse annoncee : %s%s%s\n' "$C" "${ROLE:-?}" "$Z" "$C" "${SELF:-?}" "$Z"
```

```
[ "$ROLE" = "interstp" ] || printf '  %s role=%s : ce script vise le HUB (voir diag-stp-operator.sh)\n' "$WARN"
 ↪  "${ROLE:-?}"

# adresse d'ecoute effective
LISTEN_IP="$(awk '/listen m3ua/{f=1} f&&/local-ip/{print $2; exit}' "$CFG" 2>/dev/null)"
LISTEN_PORT="$(awk '/listen m3ua/{print $3; exit}' "$CFG" 2>/dev/null)"
printf '  ecoute   : %s:%s\n' "${LISTEN_IP:-?}" "${LISTEN_PORT:-?}"
if [ "$LISTEN_IP" = "0.0.0.0" ]; then
  printf '  %s local-ip 0.0.0.0 : SCTP annonce TOUTES les adresses de la machine\n' "$WARN"
  printf '         dans son INIT (NAT, alias 172.20.x, host-only). Les noeuds tentent\n'
  printf '         des chemins morts et les ASP montent puis retombent, en boucle.\n'
  printf '         Corriger : set_stp_ip.sh --inter --ip <adresse du hub>\n'
fi

# ---- demon ------------------------------------------------------------------
head "DEMON"
ST="$(systemctl is-active osmo-interstp 2>/dev/null)"
[ -z "$ST" ] && ST="$(systemctl is-active osmo-stp 2>/dev/null)"
if [ "$ST" = active ]; then printf '  %s service actif (%s)\n' "$OK" "$ST"
else printf '  %s service inactif (%s)\n' "$KO" "${ST:-inconnu}"; fi
if ss -Sanl 2>/dev/null | grep -q ":${LISTEN_PORT:-$M3UA_PORT} "; then
  printf '  %s port M3UA %s en ecoute (SCTP)\n' "$OK" "${LISTEN_PORT:-$M3UA_PORT}"
else
  printf '  %s port M3UA %s PAS en ecoute\n' "$KO" "${LISTEN_PORT:-$M3UA_PORT}"
fi

# ---- table WAN (sert seulement a l appel des absents) -----------------------
declare -A EXP_IP=()
NODES=""
if [ -r "$WAN_FILE" ]; then
  # Ancre en debut de ligne : la ligne de commentaire du fichier rappelle le
  # format WAN_NODES="id:ip:indicatif ..." et etait lue comme une entree, d ou
  # deux noeuds fantomes ("id", "...") toujours portes manquants a l appel.
  spec="$(awk -F\" '/^WAN_NODES=/{print $2}' "$WAN_FILE" 2>/dev/null)"
  for tok in $spec; do
    id="${tok%%:*}"; rest="${tok#*:}"; ip="${rest%%:*}"
    EXP_IP[$id]="$ip"; NODES="$NODES $id"
  done
fi

# ---- AS / ASP ---------------------------------------------------------------
head "AS DECLARES PAR LE HUB"
AS_OUT="$(vty 'show cs7 instance 0 as all')"
ASP_OUT="$(vty 'show cs7 instance 0 asp')"

# Le hub construit ses AS a partir du seul NOMBRE de noeuds et apparie les ASP
# par routing context. Apparier ici as-n<n>op1 avec l'adresse attendue du noeud
# n, lue dans une table locale rangee autrement que celle en service, donnait
# l'etat d'une machine a sa voisine : un noeud sain sortait en panne et l'autre
# en marche. On lit donc le rattachement la ou le hub le declare : colonne
# AS Name et adresse distante de sa propre sortie.
# Ces colonnes ont bouge selon les versions de libosmo-sigtran : l'etat et
# l'adresse sont reperes a leur forme, pas a leur rang.
ASP_ROWS="$(printf '%s\n' "$ASP_OUT" | awk '
  /ASP_/ {
    st = ""; rem = ""
    for (i = 2; i <= NF; i++) {
      if ($i ~ /^ASP_/) st = $i
      if ($i ~ /^[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+(:[0-9]+)?$/) rem = $i
    }
    sub(/:[0-9]+$/, "", rem)
    printf "%s|%s|%s|%s\n", $1, $2, (st == "" ? "?" : st), (rem == "" ? "?" : rem)
  }')"
AS_ROWS="$(printf '%s\n' "$AS_OUT" | awk '$1 ~ /^as-/ {print $1 "|" $2}')"

if [ -z "$AS_OUT" ]; then
```

```
  printf '  %s VTY injoignable sur 127.0.0.1:%s -- nc absent, ou demon arrete\n' "$KO" "$VTY_PORT"
elif [ -z "$AS_ROWS" ]; then
  printf '  %s aucun AS declare : conf du hub vide, ou autre instance cs7\n' "$KO"
else
  while IFS='|' read -r as_name as_state; do
    [ -n "$as_name" ] || continue
    if [ "$as_state" = AS_ACTIVE ]; then m="$OK"; else m="$KO"; fi
    printf '  %s %-12s %s\n' "$m" "$as_name" "${as_state:-?}"
    attache=0
    while IFS='|' read -r asp_name asp_as asp_state asp_rem; do
      [ "$asp_as" = "$as_name" ] || continue
      attache=1
      printf '         %-12s %-12s depuis %s\n' "$asp_name" "$asp_state" "$asp_rem"
    done <<< "$ASP_ROWS"
    [ "$attache" = 0 ] && printf '         aucun ASP rattache : noeud eteint, asp en shutdown, ou SCTP bloque\n'
  done <<< "$AS_ROWS"
  # NF : sans aucun ASP, printf emet quand meme une ligne vide, dont le champ
  # 2 (vide) ne matche pas /^as-/. Le bloc annoncait alors un orphelin "()" et
  # un desaccord de routing context - dans l'etat ou il n'y a tout simplement
  # rien d'attache, celui-la meme pour lequel on lance ce diagnostic.
  orph="$(printf '%s\n' "$ASP_ROWS" | awk -F'|' 'NF && $2 !~ /^as-/ {printf "%s(%s) ", $1, $4}')"
  if [ -n "$orph" ]; then
    printf '  %s ASP connecte mais rattache a aucun AS : %s\n' "$WARN" "${orph% }"
    printf '         son routing context ne correspond a aucun AS du hub\n'
  fi
fi

head "APPEL DE LA TABLE WAN (${WAN_FILE})"
if [ -z "$ASP_OUT" ]; then
  printf '  %s pas de reponse VTY : rien a confronter a la table\n' "$KO"
elif [ -z "$NODES" ]; then
  printf '  %s table absente ou vide : impossible de dire qui manque a l appel,\n' "$WARN"
  printf '         la liste ci-dessus reste complete pour ce qui est attache\n'
else
  # Une adresse attachee absente de la table = noeud masque (NAT) : le hub voit
  # alors l'adresse du routeur et pas celle du noeud. Sans ce rappel on declare
  # absent un noeud qui parle, au seul motif qu'il est masque.
  tab=" "
  for n in $NODES; do tab="$tab${EXP_IP[$n]} "; done
  hors=""
  for ip in $(printf '%s\n' "$ASP_ROWS" | awk -F'|' '$4 ~ /^[0-9]/ {print $4}' | sort -u); do
    case "$tab" in *" $ip "*) ;; *) hors="$hors $ip" ;; esac
  done
  # Une adresse hors table ne peut masquer QU'UN seul noeud. Sans ce compte,
  # un unique ASP masque - l'etat permanent d'un conteneur derriere le NAT -
  # degradait en WARN tous les noeuds reellement absents, et le FAIL "aucune
  # association" ne pouvait plus jamais sortir.
  n_hors=0
  for _ in $hors; do n_hors=$(( n_hors + 1 )); done
  for n in $NODES; do
    if printf '%s\n' "$ASP_ROWS" | awk -F'|' -v ip="${EXP_IP[$n]}" '$4 == ip {f = 1} END {exit !f}'; then
      printf '  %s %-15s %-20s (noeud %s de la table)\n' "$OK" "${EXP_IP[$n]}" "attachee" "$n"
    elif [ "$n_hors" -gt 0 ]; then
      n_hors=$(( n_hors - 1 ))
      printf '  %s %-15s %-20s (noeud %s de la table)\n' "$WARN" "${EXP_IP[$n]}" "pas vue telle quelle" "$n"
    else
      printf '  %s %-15s %-20s (noeud %s de la table)\n' "$KO" "${EXP_IP[$n]}" "aucune association" "$n"
    fi
  done
  [ -n "$hors" ] && printf '  %s attachees hors table :%s -- masquage NAT probable\n' "$WARN" "$hors"
fi

head "ASP ATTACHES"
if [ -n "$ASP_OUT" ]; then
  printf '%s\n' "$ASP_OUT" | grep -E 'asp-|ASP Name' | sed 's/^/  /' | cut -c1-160
else
```

```
  printf '  %s pas de reponse VTY\n' "$KO"
fi

# ---- routes ----------------------------------------------------------------
head "ROUTES"
RT="$(vty 'show cs7 instance 0 route')"
if [ -n "$RT" ]; then
  bad="$(printf '%s\n' "$RT" | grep -E 'PROHIB|UNAVAIL|INACC' || true)"
  n_ok="$(printf '%s\n' "$RT" | grep -cE '^[0-9]+\.[0-9]+\.[0-9]+/' || true)"
  printf '  %s routes declarees\n' "${n_ok:-0}"
  if [ -n "$bad" ]; then
    printf '  %s destinations interdites ou injoignables :\n' "$WARN"
    printf '%s\n' "$bad" | sed 's/^/    /'
    printf '    (TFP herite d un flapping : relancer osmo-stp sur le noeud concerne)\n'
  else
    printf '  %s aucune route PROHIB/UNAVAIL\n' "$OK"
  fi
else
  printf '  %s pas de reponse VTY\n' "$KO"
fi

# ---- flapping --------------------------------------------------------------
if [ "$WATCH" = 1 ]; then
  head "FLAPPING (2e mesure a +10 s)"
  first="$(printf '%s\n' "$ASP_OUT" | grep -oE 'asp-dyn-[0-9]+' | sort -u | tr '\n' ' ')"
  printf '  t0    : %s\n' "${first:-aucun}"
  sleep 10
  second="$(vty 'show cs7 instance 0 asp' | grep -oE 'asp-dyn-[0-9]+' | sort -u | tr '\n' ' ')"
  printf '  t+10s : %s\n' "${second:-aucun}"
  if [ "$first" = "$second" ]; then
    printf '  %s ASP stables - pas de reetablissement en boucle\n' "$OK"
  else
    printf '  %s les ASP dynamiques ont change de numero : les associations se\n' "$KO"
    printf '         reetablissent en boucle. Cause habituelle : local-ip 0.0.0.0\n'
    printf '         cote hub (multi-homing vers des adresses injoignables).\n'
  fi
fi

# ---- verdict ---------------------------------------------------------------
head "VERDICT"
if [ -n "${AS_OUT:-}" ]; then
  # Deux conventions de nom coexistent : as-n<noeud>op<op> en mode WAN, et
  # as-op<i> hors mode WAN (helpers/create_interop.sh). Le verdict ne comptait
  # que la premiere : la section ci-dessus listait trois AS_ACTIVE et le verdict
  # concluait juste en dessous "aucun AS actif". Une sortie qui se dement
  # elle-meme fait chercher une panne qui n'existe pas.
  tot="$(printf '%s\n' "$AS_OUT" | grep -cE 'as-(n[0-9]+)?op[0-9]+' || true)"
  act="$(printf '%s\n' "$AS_OUT" | grep -E 'as-(n[0-9]+)?op[0-9]+' | grep -c 'AS_ACTIVE' || true)"
  if [ "${act:-0}" -gt 0 ] && [ "${act:-0}" = "${tot:-0}" ]; then
    printf '  %sINTERCO SS7 : UP%s  - %s/%s AS actifs\n' "$G$B" "$Z" "$act" "$tot"
  elif [ "${act:-0}" -gt 0 ]; then
    printf '  %sINTERCO SS7 : PARTIELLE%s - %s/%s AS actifs\n' "$Y$B" "$Z" "$act" "$tot"
    echo   "  les noeuds manquants : demon arrete, ASP en shutdown, ou adresse fausse."
    echo   "  sur le noeud : checks/diag-stp-operator.sh"
  else
    printf '  %sINTERCO SS7 : DOWN%s - aucun AS actif\n' "$R$B" "$Z"
    echo   "  a verifier, dans l ordre :"
    echo   "    - local-ip du hub (0.0.0.0 = multi-homing, voir plus haut)"
    echo   "    - les noeuds pointent-ils la bonne adresse ? (remote-ip)"
    echo   "    - leur ASP est-il en 'no shutdown' ?"
  fi
else
  printf '  %sindeterminable%s : le VTY du hub ne repond pas\n' "$R$B" "$Z"
fi
echo "------------------------------------------------------------------------"
exit 0
```

3.39 /opt/GSM/osmo-operator/checks/diag-stp-operator.sh

7631 octets, 159 lignes → 159 lignes

```
#!/bin/bash
# ============================================================================
# diag-stp-operator.sh -- diagnostic SS7/interco vu d'un noeud operateur
#
# Lit le role et le plan WAN, compare la conf sur disque a ce que le demon
# osmo-stp fait tourner reellement (VTY), teste le transport SCTP vers
# l'inter-STP, et rend un verdict clair : interco UP / DOWN + la cause.
#
# Ne modifie rien. Ne redemarre rien. Lecture seule.
#
# Usage : sudo ./diag-stp-operator.sh [--node N] [--hub-ip IP] [--no-color]
# ============================================================================
set -uo pipefail

ROLE_FILE=/etc/osmo-role
WAN_FILE=/etc/osmo-wan.conf
CONF_DIR=/etc/osmocom
STP_CFG="$CONF_DIR/osmo-stp.cfg"
MSC_CFG="$CONF_DIR/osmo-msc.cfg"
BSC_CFG="$CONF_DIR/osmo-bsc.cfg"
VTY_PORT=4239
M3UA_PORT=2908

NODE=""; HUB_IP=""; USE_COLOR=1
while [ $# -gt 0 ]; do
  case "$1" in
    --node) NODE="${2:-}"; shift ;;
    --hub-ip) HUB_IP="${2:-}"; shift ;;
    --no-color) USE_COLOR=0 ;;
    -h|--help) sed -n '2,14p' "$0" | sed 's/^# \{0,1\}//'; exit 0 ;;
  esac
  shift
done

if [ "$USE_COLOR" = 1 ] && [ -t 1 ]; then
  G=$'\033[0;32m'; R=$'\033[0;31m'; Y=$'\033[1;33m'; C=$'\033[0;36m'; B=$'\033[1m'; Z=$'\033[0m'
else G=; R=; Y=; C=; B=; Z=; fi
OK="${G}[ OK ]${Z}"; KO="${R}[FAIL]${Z}"; WARN="${Y}[WARN]${Z}"; NA="[ -- ]"
PROB=0
hr(){ printf '%s\n' "----------------------------------------------------------------------------"; }
section(){ printf '\n%s== %s ==%s\n' "$B" "$1" "$Z"; }

# ---- identite ----------------------------------------------------------
getv(){ awk -F= -v k="$1" '$1==k{gsub(/[ \r\t]/,"",$2);v=$2} END{print v}' "$ROLE_FILE" 2>/dev/null; }
[ -z "$NODE" ]   && NODE="$(getv OSMO_WAN_NODE)"
[ -z "$HUB_IP" ] && HUB_IP="$(getv OSMO_HUB_IP)"
ROLE="$(getv OSMO_ROLE)"
[ -z "$NODE" ] && NODE="?"

# IP locale de ce noeud, depuis la table WAN si dispo
SELF_IP=""
if [ -r "$WAN_FILE" ]; then
  wl="$(awk -F\" '/WAN_NODES=/{print $2}' "$WAN_FILE" 2>/dev/null)"
  for tok in $wl; do
    id="${tok%%:*}"; rest="${tok#*:}"; ip="${rest%%:*}"
    [ "$id" = "$NODE" ] && SELF_IP="$ip"
  done
fi

section "IDENTITE"
printf '  hostname     : %s\n' "$(hostname)"
printf '  role         : %s%s%s   noeud : %s%s%s\n' "$C" "${ROLE:-?}" "$Z" "$C" "$NODE" "$Z"
```

```
printf '  inter-STP    : %s%s%s   ce noeud (WAN) : %s%s%s\n' "$C" "${HUB_IP:-?}" "$Z" "$C" "${SELF_IP:-?}" "$Z"
if [ "$ROLE" != "operator" ]; then
  printf '  %s ce n est pas un noeud operateur (role=%s) -- utilise diag-interstp.sh\n' "$WARN" "${ROLE:-?}"
fi

# plan attendu pour ce noeud
if [ "$NODE" != "?" ]; then
  EXP_PC="1.${NODE}1.2"; EXP_MSC="1.${NODE}1.1"; EXP_BSC="1.${NODE}1.3"
  EXP_RK=$(( NODE*1000 + 150 ))
else
  EXP_PC=""; EXP_MSC=""; EXP_BSC=""; EXP_RK=""
fi

# ---- conf sur disque ------------------------------------------------------
section "CONF SUR DISQUE ($STP_CFG)"
f_pc="$(awk '/^cs7 instance/{c=1} c&&/^ *point-code /{print $2; exit}' "$STP_CFG" 2>/dev/null)"
f_rip="$(awk '/asp asp-to-inter/{f=1} f&&/remote-ip/{print $2; exit}' "$STP_CFG" 2>/dev/null)"
f_lip="$(awk '/asp asp-to-inter/{f=1} f&&/local-ip/{print $2; exit}' "$STP_CFG" 2>/dev/null)"
f_rk="$(awk '/as as-inter/{f=1} f&&/routing-key/{print $2; exit}' "$STP_CFG" 2>/dev/null)"
f_shut="$(awk '/asp asp-to-inter/{f=1} f&&/^ *no shutdown/{print "no"; exit} f&&/^ *shutdown/{print "yes"; exit}
 ↪  f&&/^ *as /{exit}' "$STP_CFG" 2>/dev/null)"
m_pc="$(awk '/^cs7 instance/{c=1} c&&/^ *point-code /{print $2; exit}' "$MSC_CFG" 2>/dev/null)"
b_pc="$(awk '/^cs7 instance/{c=1} c&&/^ *point-code /{print $2; exit}' "$BSC_CFG" 2>/dev/null)"

chk(){ # label got exp
  local label="$1" got="$2" exp="$3" m="$OK"
  if [ -n "$exp" ] && [ "$got" != "$exp" ]; then m="$KO"; PROB=$((PROB+1)); fi
  if [ -z "$got" ]; then m="$WARN"; got="(vide)"; fi
  printf '  %-26s %-16s %s\n' "$label" "$got" "$m$( [ -n "$exp" ]&&[ "$got" != "$exp" ]&&printf ' attendu %s'
   ↪  "$exp")"
}
chk "point-code STP"        "$f_pc"  "$EXP_PC"
chk "asp remote-ip (hub)"   "$f_rip" "$HUB_IP"
chk "asp local-ip (self)"   "$f_lip" "$SELF_IP"
chk "routing-key inter"     "$f_rk"  "$EXP_RK"
if [ "$f_shut" = "no" ]; then printf '  %-26s %-16s %s\n' "asp administratif" "no shutdown" "$OK"
elif [ "$f_shut" = "yes" ]; then printf '  %-26s %-16s %s\n' "asp administratif" "shutdown" "$KO"; PROB=$((PROB+1))
else printf '  %-26s %-16s %s\n' "asp administratif" "(defaut, actif)" "$OK"; fi
chk "point-code MSC"        "$m_pc"  "$EXP_MSC"
chk "point-code BSC"        "$b_pc"  "$EXP_BSC"

# ---- services -------------------------------------------------------------
section "SERVICES (systemd)"
for s in osmo-stp osmo-msc osmo-bsc osmo-hlr; do
  st="$(systemctl is-active "$s" 2>/dev/null)"
  m="$OK"; [ "$st" = active ] || { m="$KO"; PROB=$((PROB+1)); }
  printf '  %-14s %-10s %s\n' "$s" "${st:-inconnu}" "$m"
done

# ---- VTY : ce que le demon fait tourner reellement ------------------------
vty(){ # commands...
  { printf 'enable\n'; for c in "$@"; do printf '%s\n' "$c"; done; sleep 0.5; } \
    | timeout 6 nc -q1 127.0.0.1 "$VTY_PORT" 2>/dev/null | tr -d '\r' | sed 's/\x1b\[[0-9;]*m//g'
}
section "VTY osmo-stp (etat RUNTIME, port $VTY_PORT)"
ASP="$(vty 'show cs7 instance 0 asp')"
if [ -z "$ASP" ]; then
  printf '  %s VTY injoignable -- osmo-stp est-il lance ?\n' "$KO"; PROB=$((PROB+1))
  r_state=""; r_rem=""
else
  line="$(printf '%s\n' "$ASP" | grep -E 'asp-to-inter' | head -1)"
  r_state="$(printf '%s' "$line" | awk '{print $3}')"
  r_rem="$(printf '%s' "$line" | awk '{print $8}')"
  printf '  asp-to-inter  etat=%s%s%s  remote=%s\n' "$C" "${r_state:-absent}" "$Z" "${r_rem:-?}"
  nact="$(printf '%s\n' "$ASP" | grep -c 'ASP_ACTIVE')"
  printf '  ASP locaux ACTIVE (MSC/HLR internes) : %s\n' "$nact"
```

```
  # derive interne : fichier corrige mais demon pas recharge
  if [ -n "$r_rem" ] && [ -n "$f_rip" ] && [ "$r_rem" != "${f_rip}:${M3UA_PORT}" ] && [ "${r_rem%:*}" != "$f_rip"
   ↪  ]; then
    printf '  %s DERIVE : disque remote=%s mais demon tourne remote=%s -> reload/restart requis\n' "$WARN"
     ↪  "$f_rip" "$r_rem"
    PROB=$((PROB+1))
  fi
fi

# ---- transport ------------------------------------------------------------
section "TRANSPORT vers inter-STP ${HUB_IP:-?}:$M3UA_PORT"
if [ -n "$HUB_IP" ]; then
  if ping -c1 -W2 "$HUB_IP" >/dev/null 2>&1; then printf '  %s ping %s\n' "$OK" "$HUB_IP"
  else printf '  %s ping %s injoignable\n' "$KO" "$HUB_IP"; PROB=$((PROB+1)); fi
fi
estab="$(ss -Sanp 2>/dev/null | grep -E "${HUB_IP//./\\.}:$M3UA_PORT" | grep -i estab)"
if [ -n "$estab" ]; then printf '  %s association SCTP ETABLIE\n' "$OK"; printf '    %s\n' "$estab"
else printf '  %s aucune association SCTP etablie vers le hub\n' "$KO"; fi

# ---- verdict --------------------------------------------------------------
section "VERDICT"
if [ "$r_state" = "ASP_ACTIVE" ] && [ -n "$estab" ]; then
  printf '  %sINTERCO SS7 : UP%s  (asp-to-inter ACTIVE, SCTP etabli)\n' "$G$B" "$Z"
else
  printf '  %sINTERCO SS7 : DOWN%s\n' "$R$B" "$Z"
  echo   "  causes probables (par ordre) :"
  [ "$f_shut" = yes ] && echo "    - asp-to-inter en 'shutdown' dans la conf"
  { [ -n "$EXP_PC" ] && [ "$f_pc" != "$EXP_PC" ]; } && echo "    - point-code STP disque ($f_pc) != attendu
   ↪  ($EXP_PC)"
  { [ -n "$HUB_IP" ] && [ "$f_rip" != "$HUB_IP" ]; } && echo "    - remote-ip disque ($f_rip) != hub ($HUB_IP)"
  { [ -n "$r_rem" ] && [ "${r_rem%:*}" != "$f_rip" ]; } && echo "    - demon tourne une conf obsolete -> systemctl
   ↪  restart osmo-stp osmo-msc osmo-bsc"
  [ -z "$estab" ] && echo "    - pas de SCTP etabli (hub down, pare-feu, ou asp pas monte)"
  echo "  (rappel : ce script ne redemarre rien)"
fi
hr
exit 0
```

3.40 /opt/GSM/osmo-operator/checks/global_check.sh

25674 octets, 602 lignes → 602 lignes

```
#!/bin/bash
# global_check.sh - Verification complete de tous les composants Osmocom
#
# Verifie pour chaque operateur :
#   - STP (4239) : ASP, AS, routes
#   - HLR (4258) : connexions GSUP, abonnes
#   - MSC (4254) : ASP, SCCP, subscribers
#   - BSC (4242) : ASP, SCCP, BTS, OML/RSL
#   - BTS (4241) : etat des transceivers, LCHAN
#   - PCU (4240) : BTS, TBF, MS (si GPRS actif)
#   - SGSN (4245) : MM context, PDP context (si GPRS)
#   - GGSN (4260) : PDP context, APN (si GPRS)
#   - MGW (4243) : endpoints MGCP
#   - SIP connector (4255) : etat (si present)
#   - Services : SMS relay, MNCC socket, Asterisk
#
# DOUBLE MODE - docker ou natif
#   docker : chaque operateur est un conteneur osmo-operator-N ; on l'interroge
#            par "docker exec" ; l'inter-STP est le conteneur osmo-inter-stp.
#   natif  : ISO, VM ou machine nue ; un seul operateur, les demons tournent
#            ici, les VTY sont sur 127.0.0.1 (memes ports), les configs dans
#            /etc/osmocom.
```

```
# Le mode est DETECTE (checks/_mode.sh) et affiche en tete de sortie : un
# diagnostic qui ne dit pas dans quel monde il a regarde est inexploitable.
# --docker / --native forcent, quand la detection ne peut pas trancher (hote
# qui a docker installe mais fait tourner le lab en natif, par exemple).
#
# Usage :
#   sudo ./global_check.sh [--verbose] [--quick] [--op=N] [--docker|--native]

set -u

# ── Couleurs ──────────────────────────────────────────────────────────────
GREEN='\033[0;32m'
CYAN='\033[0;36m'
YELLOW='\033[1;33m'
RED='\033[0;31m'
BOLD='\033[1m'
NC='\033[0m'

# ── Bibliotheque de mode (docker | natif) ─────────────────────────────────
# Chemin RELATIF au script : le depot vit en /home/.../osmo-operator pendant le
# developpement et en /opt/GSM/osmo-operator sur l'ISO ; un chemin absolu casserait
# l'un des deux. Le second candidat couvre le cas d'un appel par lien
# symbolique, ou dirname ne pointe pas sur checks/.
_here="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
for _c in "$_here/_mode.sh" /opt/GSM/osmo-operator/checks/_mode.sh; do
    [ -r "$_c" ] && { . "$_c"; break; }
done
command -v osmo_mode >/dev/null || { echo "checks/_mode.sh introuvable" >&2; exit 1; }

# ── Configuration ─────────────────────────────────────────────────────────
VERBOSE=0
QUICK=0
SPECIFIC_OP=""
MODE_FORCED=0

# OSMO_MODE pose dans l'environnement est deja une forme de forcage
# ("OSMO_MODE=native ./checks/global_check.sh") : on le signale comme tel.
case "${OSMO_MODE:-}" in docker|native) MODE_FORCED=1 ;; esac

for arg in "$@"; do
    case "$arg" in
        --verbose|-v) VERBOSE=1 ;;
        --quick|-q)   QUICK=1 ;;
        --op=*)       SPECIFIC_OP="${arg#*=}" ;;
        # osmo_mode_force rend 2 si l'argument n'est pas un mode : ici il en est
        # toujours un, le "|| true" ne sert qu'a survivre a un futur set -e.
        --docker|--native|--mode=*) osmo_mode_force "$arg" && MODE_FORCED=1 || true ;;
    esac
done

MODE="$(osmo_mode)"

PASS=0
FAIL=0
WARN=0
SKIP=0

# ── Helpers ───────────────────────────────────────────────────────────────
ok()   { echo -e "  ${GREEN}✓${NC} $*"; PASS=$((PASS+1)); }
fail() { echo -e "  ${RED}✗${NC} $*"; FAIL=$((FAIL+1)); }
warn() { echo -e "  ${YELLOW}⚠${NC} $*"; WARN=$((WARN+1)); }
skip() { echo -e "  ${YELLOW}-${NC} $*"; SKIP=$((SKIP+1)); }
info() { [[ $VERBOSE -eq 1 ]] && echo -e "    ${CYAN}|${NC} $*"; }

banner() {
    echo ""
```

```
    echo -e "${BOLD}$*${NC}"
    echo -e "$(printf '—%.0s' {1..70})"
}

section() {
    echo -e "\n  ${CYAN}${BOLD}$*${NC}"
}

# ── Fonction VTY ──────────────────────────────────────────────────────
# Meme signature qu'avant - premier argument : l'ETIQUETTE du noeud
# (osmo-operator-N), que osmo_vty traduit en "docker exec" ou en execution
# locale selon le mode. Les temporisations reproduisent a l'identique celles de
# l'ancienne implementation (0.5 / 2 / nc -q1) : les changer changerait la
# QUANTITE de sortie capturee, donc les "grep -c" qui la comptent plus bas.
vty_cmd() {
    local node="$1"
    local port="$2"
    local cmd="$3"
    local timeout="${4:-2}"

    local raw_output
    raw_output=$(OSMO_VTY_OPEN=0.5 OSMO_VTY_READ="$timeout" OSMO_VTY_Q=1 \
                 osmo_vty "$node" "$port" enable "$cmd" end) || return 1

    # Filtrage identique a l'ancien (motif et ordre des filtres) ; "Free
    # Software lives" en supplement, comme avant.
    printf '%s\n' "$raw_output" | osmo_vty_clean 'Free Software lives'
    return 0
}

# ── Verification rapide de disponibilite VTY ──────────────────────────
# En natif, osmo_vty_up sonde d'abord avec ss (passif, n'ouvre pas de session
# VTY) et ne retombe sur la connexion TCP que si ss ne voit rien.
vty_available() {
    osmo_vty_up "$1" "$2"
}

# ── Detection des operateurs ──────────────────────────────────────────
# docker : les conteneurs osmo-operator-N en cours.
# natif  : $OSMO_OP_IDS, sinon les netns osmo-opN, sinon N_OPERATORS de
#          globals.conf, sinon 1 - il n'existe pas d'inventaire en natif.
if [ -n "$SPECIFIC_OP" ]; then
    # osmo_op_exists est ANCRE, contrairement a l'ancien "docker ps | grep
    # osmo-operator-N" : "--op=1" acceptait aussi osmo-operator-10.
    if ! osmo_op_exists "$SPECIFIC_OP"; then
        echo -e "${RED}Operateur ${SPECIFIC_OP} non trouve (mode ${MODE})${NC}"
        exit 1
    fi
    OP_IDS=("$SPECIFIC_OP")
else
    mapfile -t OP_IDS < <(osmo_ops)
fi

N_OPS=${#OP_IDS[@]}

# Une ligne de l'encadre, remplie a la largeur exacte (58 colonnes). Le texte
# passe ici reste en ASCII : sous une locale C, ${#t} compterait les octets
# d'un caractere accentue et le cadre se decalerait.
box_line() {
    # Deux instructions, pas une : dans "local t="$1" pad=$(( ... ${#t} ))"
    # l'arithmetique est developpee AVANT l'affectation de t, ce qui donne
    # "variable sans liaison" sous set -u.
    local t="$1"
    local pad=$(( 58 - 5 - ${#t} ))
    [ "$pad" -lt 0 ] && pad=0
    printf "${CYAN}║     %s%*s║${NC}\n" "$t" "$pad" ""
```

```
}

echo ""
echo -e "${CYAN}╔══════════════════════════════════════════════════════════════════╗${NC}"
echo -e "${CYAN}║     Global Health Check - $(date '+%Y-%m-%d %H:%M:%S')            ║${NC}"
echo -e "${CYAN}║     ${N_OPS} operateur(s) a verifier                              ║${NC}"
# Le mode fait partie du diagnostic : sans lui, un "service DOWN" ne dit pas
# s'il a ete cherche dans un conteneur ou sur cette machine.
if [ "$MODE_FORCED" -eq 1 ]; then
    box_line "mode : ${MODE} (force)"
else
    box_line "mode : ${MODE} (detecte)"
fi
if [ "$MODE" = native ]; then
    box_line "natif : demons locaux, VTY sur 127.0.0.1"
else
    box_line "docker : conteneurs osmo-operator-N"
fi
echo -e "${CYAN}╚══════════════════════════════════════════════════════════════════╝${NC}"

if [ "$N_OPS" -eq 0 ]; then
    # En natif osmo_ops rend toujours au moins "1" : ce cas est donc celui de
    # docker sans conteneur en cours. On le dit, plutot que de laisser croire a
    # une panne des services.
    echo -e "${RED}Aucun operateur trouve (mode ${MODE})${NC}"
    [ "$MODE" = docker ] && echo -e "     ${CYAN}aucun conteneur osmo-operator-N en cours - sudo
     ↪  ./launch/start-docker.sh${NC}"
    exit 1
fi

# ═══════════════════════════════════════════════════════════════════
# Pour chaque operateur
# ═══════════════════════════════════════════════════════════════════
for op_id in "${OP_IDS[@]}"; do
    # osmo_node rend "osmo-operator-N" dans les DEUX modes : en docker c'est
    # le nom du conteneur, en natif un nom d'affichage. La banniere et les
    # champs "container=" que relit operator_summary.sh sont preserves.
    container="$(osmo_node "$op_id")"
    banner "OPERATEUR ${op_id} (${container})"

    # ---------------------------------------------------------------------
    # 1. STP (port 4239)
    # ---------------------------------------------------------------------
    section "STP (Signalisation)"

    if ! vty_available "$container" 4239; then
        fail "STP VTY (4239) inaccessible - service DOWN"
    else
        # ASP locaux
        asp_out=$(vty_cmd "$container" 4239 "show cs7 instance 0 asp")
        n_asp=$(echo "$asp_out" | grep -c 'ASP_ACTIVE' || true)

        if [ "$n_asp" -ge 2 ]; then
            ok "ASP actifs : ${n_asp} (MSC+BSC ok)"
        elif [ "$n_asp" -eq 1 ]; then
            warn "ASP actifs : 1 seul (manque MSC ou BSC)"
        else
            fail "ASP actifs : aucun"
        fi

        # AS locaux
        as_out=$(vty_cmd "$container" 4239 "show cs7 instance 0 as all")
        n_as=$(echo "$as_out" | grep -c 'AS_ACTIVE' || true)

        if [ "$n_as" -ge 2 ]; then
            ok "AS actifs : ${n_as}"
```

```
    else
        warn "AS actifs : ${n_as}"
    fi

    # Routes (pas de PROHIB)
    route_out=$(vty_cmd "$container" 4239 "show cs7 instance 0 route")
    n_prohib=$(echo "$route_out" | grep -ciE 'prohib' || true)

    if [ "$n_prohib" -eq 0 ]; then
        ok "Routes : aucun PROHIB"
    else
        fail "Routes : ${n_prohib} PROHIB detectes"
    fi

    # Route par defaut vers inter-STP.
    # L'inter-STP est un conteneur en docker (osmo-inter-stp, 172.20.0.10).
    # En natif il n'existe que si CETTE machine tient ce role, ou s'il est
    # sur une autre machine du WAN : dans ce cas l'absence de route
    # catch-all n'est pas une anomalie, c'est la topologie. On l'ignore
    # explicitement au lieu de fabriquer un avertissement.
    has_default=$(echo "$route_out" | grep -cE '0\.0\.0/0.*avail' || true)
    if [ "$has_default" -gt 0 ]; then
        ok "Route par defaut → inter-STP : presente"
    elif osmo_is_native && ! osmo_hub >/dev/null 2>&1; then
        skip "Route par defaut → inter-STP - ignore (natif) : $(osmo_hub_hint)"
    else
        warn "Route par defaut → inter-STP : absente"
    fi
fi

# ============================================================================
# 2. HLR (port 4258)
# ============================================================================
section "HLR (Abonnes)"

if ! vty_available "$container" 4258; then
    fail "HLR VTY (4258) inaccessible - service DOWN"
else
    # Connexions GSUP
    gsup_out=$(vty_cmd "$container" 4258 "show gsup-connections")
    vlr_conn=$(echo "$gsup_out" | grep -c "VLR" || true)
    smsc_conn=$(echo "$gsup_out" | grep -c "SMSC" || true)

    [ "$vlr_conn" -gt 0 ] && ok "GSUP : VLR connecte" || fail "GSUP : VLR absent"
    [ "$smsc_conn" -gt 0 ] && ok "GSUP : SMSC connecte" || warn "GSUP : SMSC absent"

    # Nombre d'abonnes
    sub_out=$(vty_cmd "$container" 4258 "show subscriber count")
    sub_count=$(echo "$sub_out" | grep -oE '[0-9]+' | head -1)
    info "Abonnes dans HLR : ${sub_count:-0}"
fi

# ============================================================================
# 3. MSC (port 4254)
# ============================================================================
section "MSC (Commutation)"

if ! vty_available "$container" 4254; then
    fail "MSC VTY (4254) inaccessible - service DOWN"
else
    # ASP vers STP
    msc_asp=$(vty_cmd "$container" 4254 "show cs7 instance 0 asp")
    msc_active=$(echo "$msc_asp" | grep -c 'ASP_ACTIVE' || true)
    [ "$msc_active" -gt 0 ] && ok "MSC→STP : ASP_ACTIVE" || fail "MSC→STP : pas de ASP actif"

    # SCCP
```

```
    msc_sccp=$(vty_cmd "$container" 4254 "show cs7 instance 0 sccp users")
    if echo "$msc_sccp" | grep -qE 'SSN 254'; then
        ok "SCCP SSN 254 (MSC-A) enregistre"
    else
        fail "SCCP SSN 254 absent"
    fi

    # Subscribers VLR
    vlr_out=$(vty_cmd "$container" 4254 "show subscriber count")
    vlr_count=$(echo "$vlr_out" | grep -oE '[0-9]+' | head -1 || echo "0")
    info "Subscribers dans VLR : ${vlr_count}"
fi

# ============================================================================
# 4. BSC (port 4242)
# ============================================================================
section "BSC (Controleur BTS)"

if ! vty_available "$container" 4242; then
    fail "BSC VTY (4242) inaccessible - service DOWN"
else
    # ASP vers STP
    bsc_asp=$(vty_cmd "$container" 4242 "show cs7 instance 0 asp")
    bsc_active=$(echo "$bsc_asp" | grep -c 'ASP_ACTIVE' || true)
    [ "$bsc_active" -gt 0 ] && ok "BSC→STP : ASP_ACTIVE" || fail "BSC→STP : pas de ASP actif"

    # SCCP
    bsc_sccp=$(vty_cmd "$container" 4242 "show cs7 instance 0 sccp users")
    echo "$bsc_sccp" | grep -qE 'SSN 254' && ok "SCCP SSN 254 (BSC) enregistre" || fail "SCCP SSN 254 absent"

    # BTS 0 etat
    bts_out=$(vty_cmd "$container" 4242 "show bts 0")

    if echo "$bts_out" | grep -qE "Oper 'Enabled'.*Avail 'OK'"; then
        ok "BTS 0 : Enabled / OK"
    elif echo "$bts_out" | grep -qE "Oper 'Enabled'"; then
        warn "BTS 0 : Enabled mais Avail != OK"
    else
        fail "BTS 0 : pas Enabled"
    fi

    # OML dans show bts 0
    oml_up=$(echo "$bts_out" | grep -c 'OML Link state: connected' || true)
    [ "$oml_up" -gt 0 ] && ok "OML : connecte" || fail "OML : deconnecte"

    # RSL : verifier dans show trx 0 0 (pas dans show bts 0)
    trx_out=$(vty_cmd "$container" 4242 "show trx 0 0")
    rsl_up=$(echo "$trx_out" | grep -c 'RSL State: connected' || true)

    if [ "$rsl_up" -gt 0 ]; then
        ok "RSL : connecte (TRX 0)"
    else
        # Fallback : verifier si c'est mentionne ailleurs
        rsl_alt=$(echo "$bts_out" | grep -c 'RSL' || true)
        if [ "$rsl_alt" -gt 0 ]; then
            warn "RSL : statut ambigu (verifier manuellement)"
        else
            fail "RSL : deconnecte"
        fi
    fi

    # Nombre de BTS
    n_bts=$(vty_cmd "$container" 4242 "show bts all" | grep -c '^bts' || true)
    info "BTS configurees : ${n_bts}"
fi
```

```
# ============================================================================
# 5. BTS (Station de base)
# ============================================================================
section "BTS (Station de base)"

if ! vty_available "$container" 4242; then
    skip "BSC VTY (4242) inaccessible - impossible de verifier les BTS"
else
    # Verifier BTS 0 directement
    bts0_out=$(vty_cmd "$container" 4242 "show bts 0")

    if [ -z "$bts0_out" ]; then
        fail "Impossible d'obtenir les informations de BTS 0"
    else
        # Verifier que la BTS existe (presence du nom)
        if echo "$bts0_out" | grep -q "BTS 0 is of osmo-bts type"; then
            ok "BTS 0 : configuree"

            # Etat operationnel
            if echo "$bts0_out" | grep -q "Oper 'Enabled'.*Avail 'OK'"; then
                ok "BTS 0 : operationnelle (Enabled/OK)"
            elif echo "$bts0_out" | grep -q "Oper 'Enabled'"; then
                warn "BTS 0 : Enabled mais Avail != OK"
            else
                fail "BTS 0 : pas Enabled"
            fi

            # OML
            if echo "$bts0_out" | grep -q "OML Link state: connected"; then
                ok "OML : connecte"
            else
                fail "OML : deconnecte"
            fi

            # RSL (present dans les statistiques en bas)
            if echo "$bts0_out" | grep -q "Number of RSL links connected.* 1"; then
                ok "RSL : connecte"
            elif echo "$bts0_out" | grep -q "Number of RSL links connected.* [1-9]"; then
                ok "RSL : connecte (${BASH_REMATCH[1]})"
            else
                # Verifier via TRX
                trx0_out=$(vty_cmd "$container" 4242 "show trx 0 0" 2>/dev/null)
                if echo "$trx0_out" | grep -q "RSL State: connected"; then
                    ok "RSL : connecte (TRX 0)"
                else
                    fail "RSL : deconnecte"
                fi
            fi

            # ARFCN
            arfcn=$(echo "$bts0_out" | grep -oE 'ARFCNs: [0-9]+' | head -1)
            [ -n "$arfcn" ] && info "$arfcn"

            # Uptime
            uptime=$(echo "$bts0_out" | grep "Seconds of uptime" | grep -oE '[0-9]+ s')
            [ -n "$uptime" ] && info "Uptime : ${uptime}"

            # Verifier s'il y a d'autres BTS
            n_bts=$(vty_cmd "$container" 4242 "show bts all" | grep -c '^bts' || true)
            if [ "$n_bts" -gt 1 ]; then
                ok "BTS multiples : ${n_bts} (BTS 0 OK)"
            fi
        else
            fail "BTS 0 non trouvee dans la configuration"
        fi
    fi
```

```
fi
# ============================================================================
# 6. PCU (port 4240) - GPRS
# ============================================================================
section "PCU (Controleur paquets)"

if ! vty_available "$container" 4240; then
    skip "PCU VTY (4240) - non disponible (GPRS peut-etre desactive)"
else
    # BTS
    pcu_bts=$(vty_cmd "$container" 4240 "show bts all")
    if [ -n "$pcu_bts" ]; then
        ok "PCU : connecte aux BTS"
    else
        warn "PCU : aucune BTS"
    fi

    # TBF
    tbf_out=$(vty_cmd "$container" 4240 "show tbf all")
    tbf_count=$(echo "$tbf_out" | grep -c 'TBF' || true)
    info "TBF actifs : ${tbf_count}"
fi

# ============================================================================
# 7. SGSN (port 4245) - Core paquets
# ============================================================================
section "SGSN (Serving GPRS)"

if ! vty_available "$container" 4245; then
    skip "SGSN VTY (4245) - non disponible (GPRS peut-etre desactive)"
else
    # MM context
    mm_out=$(vty_cmd "$container" 4245 "show mm-context all")
    mm_count=$(echo "$mm_out" | grep -c 'IMSI' || true)
    info "MM contexts : ${mm_count}"

    # PDP context
    pdp_out=$(vty_cmd "$container" 4245 "show pdp-context all")
    pdp_count=$(echo "$pdp_out" | grep -c 'PDP' || true)
    info "PDP contexts : ${pdp_count}"

    # NS entities
    ns_out=$(vty_cmd "$container" 4245 "show ns entities")
    ns_count=$(echo "$ns_out" | grep -c 'NSVC' || true)
    [ "$ns_count" -gt 0 ] && ok "NS entities : ${ns_count}" || warn "NS entities : 0"
fi

# ============================================================================
# 8. GGSN (port 4260) - Gateway GPRS
# ============================================================================
section "GGSN (Gateway GPRS)"

if ! vty_available "$container" 4260; then
    skip "GGSN VTY (4260) - non disponible (GPRS peut-etre desactive)"
else
    # PDP contexts
    ggsn_pdp=$(vty_cmd "$container" 4260 "show pdp-context all")
    ggsn_count=$(echo "$ggsn_pdp" | grep -c 'PDP' || true)
    info "PDP contexts GGSN : ${ggsn_count}"

    # APN
    apn_out=$(vty_cmd "$container" 4260 "show apn all")
    apn_count=$(echo "$apn_out" | grep -c 'APN' || true)
    [ "$apn_count" -gt 0 ] && ok "APN configures : ${apn_count}" || warn "APN configures : 0"
fi
```

```
# ============================================================================
# 9. MGW (port 4243) - Media Gateway
# ============================================================================
section "MGW (Media Gateway)"

if ! vty_available "$container" 4243; then
    fail "MGW VTY (4243) inaccessible"
else
    # Endpoints
    ep_out=$(vty_cmd "$container" 4243 "show mgcp endpoint all")
    ep_count=$(echo "$ep_out" | grep -c 'Endpoint' || true)
    info "Endpoints MGCP : ${ep_count}"

    # Connexions actives
    conn_out=$(vty_cmd "$container" 4243 "show mgcp stats")
    if echo "$conn_out" | grep -qE "active calls.*[1-9]"; then
        ok "MGW : appels actifs"
    else
        info "MGW : aucun appel actif"
    fi
fi

# ============================================================================
# 10. SIP connector (port 4255)
# ============================================================================
section "SIP connector (Voix)"

if ! vty_available "$container" 4255; then
    skip "SIP connector VTY (4255) - non disponible"
else
    ok "SIP connector : accessible"
    # Pas beaucoup de commandes show utiles dans osmo-sip-connector
fi

# ============================================================================
# 11. Services
# ============================================================================
section "Services"

# MNCC socket - osmo_sock interroge ss AVANT "test -S" : sous PrivateTmp
# (unit systemd de l'ISO) le socket existe sans etre visible par test -S, et
# un "voix impossible" mensonger en decoulerait.
if osmo_sock "$container" /tmp/msc_mncc; then
    ok "MNCC socket present (MSC ↔ SIP connector)"
else
    fail "MNCC socket absent - voix impossible"
fi

# SMS relay - awk sur le port exact, et non "grep -c :7890" qui aurait
# aussi accepte 17890.
if osmo_port "$container" 7890 tcp; then
    ok "SMS relay : ecoute sur port 7890"
else
    warn "SMS relay : non detecte"
fi

# Asterisk
if osmo_running "$container" asterisk; then
    ok "Asterisk : en cours d'execution"

    # Verification rapide Asterisk
    ast_out=$(osmo_ast "$container" "core show channels" || true)
    channels=$(echo "$ast_out" | grep -c "active channels" || true)
    info "Asterisk : ${channels} canaux actifs"
else
    warn "Asterisk : non detecte"
```

```
    fi

    # Verification des processus cles
    section "Processus"

    # En natif, osmo_running essaie systemctl PUIS pgrep -x PUIS pgrep -f : les
    # demons sont des units sur l'ISO mais des processus detaches quand systemd
    # est absent, et /proc/PID/comm est tronque a 15 caracteres ("pgrep -x
    # osmo-sip-connector" ne trouve jamais rien).
    if [ "$MODE" = native ] && [ "$N_OPS" -gt 1 ]; then
        info "natif multi-operateur : ip netns exec n'isole pas les PID, ces etats portent sur la machine"
    fi

    for proc in osmo-stp osmo-hlr osmo-msc osmo-bsc osmo-bts-trx osmo-mgw osmo-sip-connector; do
        if osmo_running "$container" "$proc"; then
            ok "$proc : en cours"
        else
            # Certains processus peuvent ne pas etre presents selon configuration
            skip "$proc : non trouve"
        fi
    done

    echo ""
done

# ═══════════════════════════════════════════════════════════════════════
# RESUME GLOBAL
# ═══════════════════════════════════════════════════════════════════════
echo ""
echo -e "$(printf '═%.0s' {1..70})"
echo -e "  ${BOLD}RESUME GLOBAL${NC}"
echo -e "$(printf '═%.0s' {1..70})"

TOTAL=$((PASS + FAIL + WARN + SKIP))

echo -e "  ${CYAN}mode   : ${MODE}${NC}  (${N_OPS} operateur(s))"
echo -e "  ${GREEN}✓ PASS : ${PASS}${NC}"
echo -e "  ${RED}✗ FAIL : ${FAIL}${NC}"
echo -e "  ${YELLOW}⚠ WARN : ${WARN}${NC}"
echo -e "  ${YELLOW}- SKIP : ${SKIP}${NC}"
echo -e "$(printf '─%.0s' {1..70})"

if [ "$FAIL" -eq 0 ] && [ "$WARN" -eq 0 ]; then
    echo -e "  ${GREEN}${BOLD}TOUT EST OK${NC}  -  Tous les composants fonctionnent"
elif [ "$FAIL" -eq 0 ]; then
    echo -e "  ${YELLOW}${BOLD}FONCTIONNEL AVEC AVERTISSEMENTS${NC}  -  Verifier les warnings"
else
    echo -e "  ${RED}${BOLD}PROBLEMES DETECTES${NC}  -  ${FAIL} test(s) en echec"
fi
echo -e "$(printf '═%.0s' {1..70})"
echo ""

exit "$FAIL"
```

3.41 /opt/GSM/osmo-operator/checks/operator_summary.sh

21443 octets, 485 lignes → 485 lignes

```
#!/bin/bash
# operator_summary.sh - Resume des caracteristiques principales de chaque operateur
#
# Analyse le dump produit par checks/vty-debug-dump.sh et en extrait, operateur
# par operateur, les quelques chiffres qui disent si le reseau vit : ASP actifs,
# routes SS7, abonnes HLR/VLR, etat des BTS, contextes PDP, appels en cours.
```

```
#
# DOUBLE MODE (docker / natif)
# ----------------------------
# Ce script ne parle a AUCUN demon : il lit un fichier. Le mode ne change donc
# pas ses quinze extractions - il change trois choses, et il faut les dire :
#
#   1. la GENERATION du dump (--live) : le generateur doit regarder le MEME
#      monde que nous, sinon on analyse le dump d'un autre lab. Le mode lui est
#      transmis par OSMO_MODE, y compris quand il a ete force ici.
#
#   2. l'INTER-STP : en docker c'est le conteneur osmo-inter-stp ; en natif
#      c'est soit cette machine (role interstp), soit une AUTRE machine - et
#      dans ce dernier cas son VTY n'est pas joignable d'ici, donc le dump n'en
#      contient rien. La section est alors IGNOREE, explicitement : un
#      "0 AS actif" fabrique se lirait comme une panne du coeur SS7.
#
#   3. l'etiquette "container=osmo-operator-N" du dump : en natif elle reste
#      la meme (checks/_mode.sh, osmo_node). C'est une CLE DE CORRELATION entre
#      les blocs du dump, pas un objet docker interrogeable - on ne la renomme
#      donc pas, sous peine de casser toutes les extractions ci-dessous.
#
# Le mode retenu est affiche en tete : un resume qui ne dit pas dans quel monde
# il a regarde n'est pas exploitable.
#
# Usage :
#   sudo ./operator_summary.sh [--dump=/chemin/dump.txt] [--live]
#                              [--docker|--native]

set -euo pipefail

# ── Bibliotheque commune de detection docker/natif ──────────────────────
# Chemin relatif a CE fichier (jamais absolu) : le depot vit dans /home/... en
# developpement et dans /opt/GSM/osmo-operator sur l'ISO.
_here="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
for _c in "$_here/_mode.sh" /opt/GSM/osmo-operator/checks/_mode.sh; do
    [ -r "$_c" ] && { . "$_c"; break; }
done
command -v osmo_mode >/dev/null || { echo "checks/_mode.sh introuvable" >&2; exit 1; }

# ── Couleurs ────────────────────────────────────────────────────────────
GREEN='\033[0;32m'
CYAN='\033[0;36m'
YELLOW='\033[1;33m'
RED='\033[0;31m'
BOLD='\033[1m'
NC='\033[0m'

# ── Configuration ───────────────────────────────────────────────────────
DUMP_FILE=""
LIVE_MODE=0
TEMP_DUMP=""

# OSMO_MODE herite de l'environnement vaut choix explicite, au meme titre que
# --docker/--native : on le signale comme "force" dans l'en-tete.
MODE_FORCED=0
[ -n "${OSMO_MODE:-}" ] && MODE_FORCED=1

for arg in "$@"; do
    case "$arg" in
        --dump=*) DUMP_FILE="${arg#*=}" ;;
        --live)   LIVE_MODE=1 ;;
        --docker|--native|--mode=*)
            # osmo_mode_force rend 2 si l'argument n'est pas un mode : seul
            # "--mode=n'importe-quoi" peut encore echouer ici.
            if osmo_mode_force "$arg"; then
                MODE_FORCED=1
```

```
            else
                echo "Mode inconnu : ${arg} (attendu : --docker, --native)" >&2
                exit 1
            fi
            ;;
        --help|-h)
            echo "Usage: $0 [--dump=/chemin/dump.txt] [--live] [--docker|--native]"
            echo "  --dump   : Utilise un fichier de dump existant"
            echo "  --live   : Genere un dump live avant l'analyse"
            echo "  --docker : Force le mode conteneurs (osmo-operator-N)"
            echo "  --native : Force le mode natif (ISO, VM, machine nue)"
            echo "             sans option, le mode est detecte automatiquement"
            exit 0
            ;;
    esac
done

MODE="$(osmo_mode)"
if [ "$MODE_FORCED" -eq 1 ]; then MODE_SRC="force"; else MODE_SRC="detecte"; fi

# ── Helper d'affichage ──────────────────────────────────────────────
print_header() {
    echo ""
    echo -e "${BOLD}$1${NC}"
    echo -e "$(printf '─%.0s' {1..70})"
}

print_row() {
    printf "  %-20s : %s\n" "$1" "$2"
}

print_section() {
    echo -e "\n  ${CYAN}${BOLD}$1${NC}"
}

# Ligne de la boite d'en-tete, calee sur les 58 colonnes du filet ╔═╗.
box_line() {
    local txt="$1" pad
    pad=$(( 58 - 5 - ${#txt} ))
    [ "$pad" -ge 0 ] || pad=0
    echo -e "${CYAN}║     ${txt}$(printf '%*s' "$pad" '')║${NC}"
}

# ── Extraction de donnees depuis le dump ────────────────────────────
#
# PIEGE PIPEFAIL, valable pour TOUTES les extractions de ce fichier : sous
# "set -o pipefail", un grep qui ne trouve rien (rc 1) ou un "head -1" qui
# sort tot (le producteur meurt en SIGPIPE, rc 141) fait echouer le tube entier,
# donc l'affectation "var=$(...)", donc - sous "set -e" - le script lui-meme.
# Sur un dump vide ou partiel (mode natif ou tout est absent, dump produit
# ailleurs) le resume s'interrompait sans un mot. D'ou les "|| true" : ils ne
# masquent aucune erreur, ils disent "rien trouve = zero", ce que wc et grep -c
# ont deja ecrit sur leur sortie.

extract_value() {
    local container="$1"
    local pattern="$2"
    local default="${3:-?}"

    grep -A20 "\[.*\] port=.* container=${container}" "$DUMP_FILE" \
        | grep -E "$pattern" \
        | head -1 \
        | sed -E 's/^[[:space:]]+//' \
        || echo "$default"
}
```

```
extract_count() {
    local container="$1"
    local section="$2"
    local pattern="$3"

    grep -A50 "\[.*\] port=.* container=${container}.*$section" "$DUMP_FILE" \
        | grep -E "$pattern" \
        | wc -l \
        | tr -d ' ' \
        || true
}

extract_subscriber_count() {
    local container="$1"
    local count

    count=$(grep -A20 "\[OsmoHLR\]" "$DUMP_FILE" \
        | grep -A5 "container=${container}" \
        | grep -E "subscriber count.*[0-9]+" \
        | grep -oE '[0-9]+' \
        | head -1 || true)

    echo "${count:-0}"
}

# Renseigne BTS_STATE, OML_OK et RSL_OK - des variables GLOBALES, a dessein.
# La version precedente ecrivait l'etat sur stdout ET posait OML_OK/RSL_OK dans
# la foulee ; appelee en "$( )" elle tournait dans un SOUS-SHELL, ou ces deux
# affectations mouraient avec lui : les lignes OML et RSL affichaient
# "✗ deconnecte" meme quand le dump montrait des liens connectes. On renseigne
# donc les trois au meme endroit, sans substitution de commande.
extract_bts_info() {
    local container="$1"
    local bts_id="$2"
    local bts_block oper avail oml rsl

    bts_block=$(grep -A50 "\[OsmoBSC\]" "$DUMP_FILE" \
        | grep -A30 "container=${container}" \
        | grep -A20 "bts ${bts_id}" \
        | sed -n '/bts [0-9]/,/^$/p' || true)

    oper=$(echo "$bts_block" | grep -c "Oper 'Enabled'" || true)
    avail=$(echo "$bts_block" | grep -c "Avail 'OK'" || true)
    oml=$(echo "$bts_block" | grep -c "OML Link state: connected" || true)
    rsl=$(echo "$bts_block" | grep -c "RSL State: connected" || true)

    if [ "$oper" -gt 0 ] && [ "$avail" -gt 0 ]; then
        BTS_STATE="OK"
    elif [ "$oper" -gt 0 ]; then
        BTS_STATE="Enabled"
    else
        BTS_STATE="Down"
    fi

    # Renseigne aussi OML/RSL pour info
    [ "$oml" -gt 0 ] && OML_OK=1 || OML_OK=0
    [ "$rsl" -gt 0 ] && RSL_OK=1 || RSL_OK=0
}

extract_asp_state() {
    local container="$1"
    local component="$2"  # STP, MSC, BSC

    local asp_block=""
    if [ "$component" = "STP" ]; then
        asp_block=$(grep -A30 "\[OsmoSTP\]" "$DUMP_FILE" \
```

```
            | grep -A20 "container=${container}" || true)
    elif [ "$component" = "MSC" ]; then
        asp_block=$(grep -A30 "\[OsmoMSC\]" "$DUMP_FILE" \
            | grep -A20 "container=${container}" || true)
    elif [ "$component" = "BSC" ]; then
        asp_block=$(grep -A30 "\[OsmoBSC\]" "$DUMP_FILE" \
            | grep -A20 "container=${container}" || true)
    fi

    echo "$asp_block" | grep -c "ASP_ACTIVE" || true
}

# ── Generation d'un dump live si demande ──────────────────────────────
if [ "$LIVE_MODE" -eq 1 ]; then
    echo -e "${CYAN}Generation d'un dump live (mode ${MODE})...${NC}"
    TEMP_DUMP="/tmp/vty-live-$$.txt"

    # Le generateur est cherche A COTE DE NOUS (BASH_SOURCE), plus dans le
    # repertoire courant : "./vty-debug-dump.sh" n'existait que si l'on avait
    # fait "cd checks/" au prealable, ce qui n'est pas le cas depuis l'ISO.
    DUMPER="$_here/vty-debug-dump.sh"
    [ -x "$DUMPER" ] || DUMPER="./vty-debug-dump.sh"

    # OSMO_MODE est transmis explicitement : le generateur doit interroger le
    # meme monde que celui qu'on analysera, meme si --native/--docker a force
    # un mode qu'il n'aurait pas detecte seul.
    if ! OSMO_MODE="$MODE" "$DUMPER" --out "$TEMP_DUMP" > /dev/null; then
        echo -e "${RED}Echec de la generation du dump (${DUMPER}, mode ${MODE}).${NC}" >&2
        if [ "$MODE" = native ]; then
            echo "  Mode natif : verifiez que les demons tournent (VTY 4239/4254/4242/4258...)." >&2
        else
            echo "  Mode docker : verifiez que les conteneurs osmo-operator-N sont en cours." >&2
        fi
        rm -f "$TEMP_DUMP"
        exit 1
    fi
    DUMP_FILE="$TEMP_DUMP"
    echo -e "${GREEN}✓ Dump genere : $DUMP_FILE${NC}"
fi

if [ -z "$DUMP_FILE" ] || [ ! -f "$DUMP_FILE" ]; then
    echo -e "${RED}Erreur : fichier de dump non specifie ou introuvable${NC}"
    echo "Utilisez --dump=/chemin/vers/dump.txt ou --live pour generer un dump"
    exit 1
fi

# ── Detection des noeuds presents DANS LE DUMP ────────────────────────
# "container=osmo-operator-N" est ecrit par vty-debug-dump.sh dans les deux
# mondes : en docker c'est le nom du conteneur, en natif l'etiquette rendue par
# osmo_node(). On lit donc le dump de la meme facon des deux cotes.
mapfile -t CONTAINERS < <(
    grep -E "container=osmo-operator-[0-9]+" "$DUMP_FILE" \
        | sed -E 's/.*container=([^ ]+).*/\1/' \
        | sort -u
)

INTER_STP=$(grep -E "container=osmo-inter-stp" "$DUMP_FILE" \
    | sed -E 's/.*container=([^ ]+).*/\1/' \
    | head -1 || true)

N_OPS=${#CONTAINERS[@]}

# ── En-tete ───────────────────────────────────────────────────────────
echo ""
echo -e "${CYAN}╔══════════════════════════════════════════════════════════════╗${NC}"
echo -e "${CYAN}║     Resume Operateurs - $(date '+%Y-%m-%d %H:%M')              ║${NC}"
```

```
echo -e "${CYAN}║     ${N_OPS} operateur(s) trouve(s)                                  ${NC}"
box_line "Mode : ${MODE} (${MODE_SRC})"
echo -e "${CYAN}╚═══════════════════════════════════════════════════════════════════${NC}"

# Dump muet : on le dit. Le comparer a l'inventaire vivant distingue "rien ne
# tourne" de "le dump vient d'ailleurs / est partiel" - deux pannes tres
# differentes, surtout en natif ou aucun "docker ps" ne peut trancher.
if [ "$N_OPS" -eq 0 ]; then
    live_ops="$(osmo_ops 2>/dev/null | tr '\n' ' ' || true)"
    live_ops="${live_ops% }"
    echo ""
    echo -e "  ${YELLOW}Aucun bloc "container=osmo-operator-N" dans ce dump.${NC}"
    print_row "Operateurs vus ici" "${live_ops:-aucun} (mode ${MODE})"
    print_row "Dump" "vide, partiel, ou produit sur un autre noeud"
fi

# ── Inter-STP ──────────────────────────────────────────────────────
if [ -n "$INTER_STP" ]; then
    print_header "INTER-STP HUB (${INTER_STP})"

    asp_count=$(extract_asp_state "$INTER_STP" "STP")
    print_row "ASP actifs" "$asp_count"

    # Routes
    routes=$(grep -A50 "\[OsmoSTP-hub\]" "$DUMP_FILE" \
        | grep -A30 "container=${INTER_STP}" \
        | grep -E '^[0-9]+\.[0-9]+\.[0-9]+/[0-9]+' \
        | wc -l || true)
    print_row "Routes configurees" "$routes"

    # AS
    as_count=$(grep -A50 "\[OsmoSTP-hub\]" "$DUMP_FILE" \
        | grep -A30 "container=${INTER_STP}" \
        | grep -c "AS_ACTIVE" || true)
    print_row "AS actifs" "$as_count"

elif osmo_is_native; then
    # Rien sur le hub dans le dump, et nous sommes en natif : soit cette machine
    # EST le hub et le dump est plus vieux qu'elle, soit le hub est une AUTRE
    # machine, dont le VTY 4239 n'ecoute que sur son propre 127.0.0.1 - il n'est
    # donc PAS observable d'ici. On l'ignore visiblement au lieu d'afficher des
    # compteurs a zero qui se liraient comme un coeur SS7 mort.
    print_header "INTER-STP HUB - ignore (natif)"
    if hub_here="$(osmo_hub)"; then
        print_row "Hub local" "${hub_here} present ici, mais absent du dump"
        print_row "Pour l'inclure" "relancez avec --live"
    else
        print_row "Hub" "$(osmo_hub_hint)"
    fi

elif hub_remote="$(osmo_hub_is_remote 2>/dev/null)"; then
    # WAN : le hub vit sur une AUTRE machine (--hub-ip / OSMO_HUB_IP), il ne
    # peut donc pas figurer dans un dump pris ici, et son VTY 4239 n'ecoute que
    # sur sa propre boucle locale. Ce qui est observable d'ici, c'est le LIEN
    # vers lui : l'association SCTP que chaque noeud tient ouverte.
    print_header "INTER-STP HUB - distant (${hub_remote})"
    hub_assoc="$(osmo_hub_assoc "$hub_remote" 2>/dev/null)" && hub_rc=0 || hub_rc=$?
    case "$hub_rc" in
        0) print_row "Lien M3UA/SCTP" "${hub_assoc} noeud(s) associe(s) a ${hub_remote}:${OSMO_HUB_M3UA_PORT}" ;;
        1) print_row "Lien M3UA/SCTP" "aucune association vers ${hub_remote}:${OSMO_HUB_M3UA_PORT}" ;;
        *) print_row "Lien M3UA/SCTP" "non sondable ici (ss --sctp absent des noeuds)" ;;
    esac
    print_row "Etat interne" "$(osmo_hub_hint)"

elif hub_here="$(osmo_hub)"; then
    # Mode docker : le conteneur tourne mais ne figure pas dans le dump - le
```

```
    # dump est donc anterieur au hub. Sans conteneur hub, on ne dit rien, comme
    # avant : il n'y a simplement pas d'inter-STP dans ce lab.
    print_header "INTER-STP HUB (${hub_here})"
    print_row "Etat" "conteneur en cours, mais absent du dump"
    print_row "Pour l'inclure" "relancez avec --live"
fi

# ── Boucle sur les operateurs ──────────────────────────────────────
# "${CONTAINERS[@]+...}" : sur un dump muet le tableau est vide, et sous
# "set -u" un ""${CONTAINERS[@]}"" nu casserait au lieu de ne rien faire.
for container in ${CONTAINERS[@]+"${CONTAINERS[@]}"}; do
    # Le numero est le suffixe de l'etiquette ; la mapfile n'a laisse passer que
    # "osmo-operator-<chiffres>", donc pas de sous-shell ni de grep ici.
    op_id="${container##*-}"

    print_header "OPERATEUR ${op_id} (${container})"

    # ── STP local ──────────────────────────────────────────────────
    print_section "SS7"

    # ASP actifs dans STP
    stp_asp=$(extract_asp_state "$container" "STP")
    print_row "ASP actifs (STP)" "$stp_asp"

    # Routes
    routes=$(grep -A50 "\[OsmoSTP\]" "$DUMP_FILE" \
        | grep -A30 "container=${container}" \
        | grep -E '^[0-9]+\.[0-9]+\.[0-9]+/[0-9]+' \
        | wc -l || true)
    print_row "Routes configurees" "$routes"

    # ── MSC ────────────────────────────────────────────────────────
    print_section "MSC"

    msc_asp=$(extract_asp_state "$container" "MSC")
    print_row "ASP actif" "$([ "$msc_asp" -gt 0 ] && echo "✓" || echo "✗")"

    # Subscribers VLR
    vlr_count=$(grep -A30 "\[OsmoMSC\]" "$DUMP_FILE" \
        | grep -A20 "container=${container}" \
        | grep -E "subscriber count.*[0-9]+" \
        | grep -oE '[0-9]+' \
        | head -1 || true)
    print_row "Subscribers VLR" "${vlr_count:-0}"

    # Transactions
    trans=$(grep -A30 "\[OsmoMSC\]" "$DUMP_FILE" \
        | grep -A10 "container=${container}" \
        | grep -c "transaction" || true)
    print_row "Transactions" "$trans"

    # ── BSC ────────────────────────────────────────────────────────
    print_section "BSC"

    bsc_asp=$(extract_asp_state "$container" "BSC")
    print_row "ASP actif" "$([ "$bsc_asp" -gt 0 ] && echo "✓" || echo "✗")"

    # BTS 0 etat
    OML_OK=0
    RSL_OK=0
    BTS_STATE="Down"
    extract_bts_info "$container" 0
    print_row "BTS 0 etat" "$BTS_STATE"
    print_row "  ├─ OML" "$([ "$OML_OK" -eq 1 ] && echo "✓ connecte" || echo "✗ deconnecte")"
    print_row "  └─ RSL" "$([ "$RSL_OK" -eq 1 ] && echo "✓ connecte" || echo "✗ deconnecte")"
```

```
# Nombre de BTS total
n_bts=$(grep -A50 "\[OsmoBSC\]" "$DUMP_FILE" \
    | grep -A30 "container=${container}" \
    | grep -c "^ bts [0-9]" || true)
print_row "BTS totales" "$n_bts"

# ── HLR ──────────────────────────────────────────────────────────
print_section "HLR"

subs_count=$(extract_subscriber_count "$container")
print_row "Abonnes" "$subs_count"

# Connexions GSUP
gsup=$(grep -A20 "\[OsmoHLR\]" "$DUMP_FILE" \
    | grep -A10 "container=${container}" \
    | grep -c "GSUP" || true)
print_row "Connexions GSUP" "$gsup"

# ── BTS (si disponible) ──────────────────────────────────────────
print_section "BTS (phy)"

# LCHAN actifs
lchan=$(grep -A30 "\[OsmoBTS\]" "$DUMP_FILE" \
    | grep -A10 "container=${container}" \
    | grep -c "lchan" || true)
print_row "LCHAN" "$lchan"

# ── SGSN/GGSN (si GPRS actif) ────────────────────────────────────
print_section "GPRS"

# NS entities
ns=$(grep -A30 "\[OsmoSGSN\]" "$DUMP_FILE" \
    | grep -A10 "container=${container}" \
    | grep -c "NS entity" || true)
print_row "NS entities" "$ns"

# PDP contexts
pdp=$(grep -A30 "\[OsmoGGSN\]" "$DUMP_FILE" \
    | grep -A10 "container=${container}" \
    | grep -c "PDP context" || true)
print_row "PDP contexts" "$pdp"

# ── Services auxiliaires ─────────────────────────────────────────
print_section "Services"

# SIP connector
sip_logs=$(grep -A10 "\[OsmoSIPconn\]" "$DUMP_FILE" \
    | grep -c "container=${container}" || true)
print_row "SIP connector" "$([ "$sip_logs" -gt 0 ] && echo "✓ actif" || echo "?")"

# SMS relay port
sms=$(grep -A10 "\[.*\]" "$DUMP_FILE" \
    | grep -A5 "container=${container}" \
    | grep -c "port 7890" || true)
print_row "SMS relay" "$([ "$sms" -gt 0 ] && echo "✓ actif" || echo "?")"

# ── Statistiques rapides ─────────────────────────────────────────
print_section "Statistiques"

# Appels actifs
calls=$(grep -A20 "\[OsmoMSC\]" "$DUMP_FILE" \
    | grep -A10 "container=${container}" \
    | grep -c "call-leg" || true)
print_row "Appels actifs" "$calls"

# Paging
```

```
    paging=$(grep -A20 "\[OsmoBSC\]" "$DUMP_FILE" \
        | grep -A10 "container=${container}" \
        | grep -c "paging" || true)
    print_row "Paging actifs" "$paging"

done

# ── Nettoyage ──────────────────────────────────────────────────────────
if [ "$LIVE_MODE" -eq 1 ] && [ -n "$TEMP_DUMP" ]; then
    rm -f "$TEMP_DUMP"
fi

echo ""
echo -e "${GREEN}╔═══════════════════════════════════════════════════════════════╗${NC}"
echo -e "${GREEN}║  Analyse terminee - Dump utilise : ${DUMP_FILE}${NC}"
echo -e "${GREEN}╚═══════════════════════════════════════════════════════════════╝${NC}"
echo ""
```

3.42 /opt/GSM/osmo-operator/checks/ss7_check.sh

24154 octets, 495 lignes → 495 lignes

```
#!/bin/bash
# ss7_check.sh - Verification SS7 pour architecture Inter-STP central
#
# Architecture cible :
#   - Inter-STP (0.23.0) central
#   - Chaque operateur a STP local (N.23.2) avec route par defaut vers inter-STP
#   - Routes dynamiques apprises via M3UA
#
# Ce qui est verifie :
#   1. Inter-STP : AS actifs pour chaque operateur
#   2. Inter-STP : ASP connectes
#   3. Inter-STP : pas de PROHIB
#   4. Chaque operateur : ASP actif vers inter-STP
#   5. Chaque operateur : routes locales (MSC/BSC) actives
#   6. Chaque operateur : route par defaut (0.0.0/0) vers inter-STP
#   7. Matrice de connectivite basique
#
# DOUBLE MODE - docker ET natif (ISO, VM, machine nue)
# ----------------------------------------------------
# Toute la mecanique de detection et d'acces aux noeuds vit dans checks/_mode.sh ;
# ici on ne fait qu'appeler ses fonctions. Les deux mondes :
#
#   docker : un conteneur osmo-operator-N par operateur, hub = conteneur
#            osmo-inter-stp ; on atteint tout par "docker exec".
#   natif  : un seul operateur (sauf topologie netns), les VTY sont sur
#            127.0.0.1 de CETTE machine. Le hub, lui, est SOIT cette machine
#            (OSMO_ROLE=interstp dans /etc/osmo-role), SOIT une AUTRE machine du
#            WAN - et dans ce second cas son VTY n'est PAS observable d'ici : le
#            VTY Osmocom n'ecoute que sur la boucle locale. On l'annonce alors
#            "ignore", on ne fabrique pas un "0 AS actif" qui se lirait comme
#            une panne alors que le hub tourne tres bien ailleurs.
#
# Les PORTS VTY sont les memes dans les deux mondes (4239 STP, 4254 MSC,
# 4242 BSC, 4258 HLR) : seul l'hote change, et c'est _mode.sh qui s'en occupe.
#
# Usage :
#   sudo ./ss7_check.sh [--verbose] [--quick] [--docker|--native]

set -u

# ── Bibliotheque commune (detection docker/natif, VTY, inventaire) ─────
# Chemin RELATIF au script : le depot tourne aussi bien depuis un clone que
```

```
# depuis /opt/GSM/osmo-operator sur l'ISO, et checks/ est copie en entier par
# build-iso.sh - la bibliotheque est donc toujours a cote de ce fichier.
_here="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
for _c in "$_here/_mode.sh" /opt/GSM/osmo-operator/checks/_mode.sh; do
    [ -r "$_c" ] && { . "$_c"; break; }
done
command -v osmo_mode >/dev/null || { echo "checks/_mode.sh introuvable" >&2; exit 1; }

# ── Couleurs ──────────────────────────────────────────────────────────
GREEN='\033[0;32m'
CYAN='\033[0;36m'
YELLOW='\033[1;33m'
RED='\033[0;31m'
BOLD='\033[1m'
NC='\033[0m'

# ── Configuration ─────────────────────────────────────────────────────
VERBOSE=0
QUICK=0

for arg in "$@"; do
    case "$arg" in
        --verbose|-v) VERBOSE=1 ;;
        --quick|-q)   QUICK=1 ;;
        # --docker / --native / --mode=X : forcer le monde au lieu de le
        # detecter. osmo_mode_force rend 2 si l'argument n'est pas un mode, ce
        # qui ne peut pas arriver ici puisque le motif du case l'a deja filtre.
        --docker|--native|--mode=*)
            osmo_mode_force "$arg" || { echo "mode inconnu : $arg" >&2; exit 1; } ;;
        -h|--help)
            echo "Usage : $0 [--verbose|-v] [--quick|-q] [--docker|--native]"
            echo "  --docker / --native : force le mode ; sans option, detection automatique."
            exit 0 ;;
    esac
done

INTER_STP="osmo-inter-stp"
PASS=0
FAIL=0
WARN=0
SKIP=0

# ── Helpers ───────────────────────────────────────────────────────────
ok()   { echo -e "  ${GREEN}✓${NC} $*"; PASS=$((PASS+1)); }
fail() { echo -e "  ${RED}✗${NC} $*"; FAIL=$((FAIL+1)); }
warn() { echo -e "  ${YELLOW}⚠${NC} $*"; WARN=$((WARN+1)); }
skip() { echo -e "  ${YELLOW}-${NC} $*"; SKIP=$((SKIP+1)); }
info() { [[ $VERBOSE -eq 1 ]] && echo -e "    ${CYAN}|${NC} $*"; }

banner() {
    echo ""
    echo -e "${BOLD}$*${NC}"
    echo -e "$(printf '─%.0s' {1..60})"
}

# ── Fonction VTY ──────────────────────────────────────────────────────
# Meme contrat qu'avant : rend 1 sans rien ecrire si le VTY ne repond pas,
# sinon la sortie nettoyee. Le dialogue lui-meme (nc/telnet, docker exec ou
# local, netns) est dans osmo_vty ; on garde ICI les temporisations de ce
# script - les changer changerait la QUANTITE de sortie capturee, donc les
# "grep -ac" qui la comptent juste en dessous.
: "${OSMO_VTY_OPEN:=1}"    # sleep avant d'ecrire   (valeur historique de ss7_check)
: "${OSMO_VTY_READ:=1.5}"  # sleep de lecture       (idem)
: "${OSMO_VTY_Q:=1}"       # nc -q1                 (idem)

vty_cmd() {
```

```
    local node="$1"
    local port="$2"
    local cmd="$3"

    local raw_output
    raw_output=$(osmo_vty "$node" "$port" enable "$cmd" end) || return 1

    # Filtrer les lignes de banniere/prompt (motif et ordre identiques a avant).
    printf '%s\n' "$raw_output" | osmo_vty_clean 'Free Software lives'
    return 0
}

# ── Verification rapide de disponibilite VTY ──────────────────────────
vty_available() {
    osmo_vty_up "$1" "$2"
}

# ── Detection du monde et inventaire ──────────────────────────────────
MODE="$(osmo_mode)"

# Operateurs : conteneurs osmo-operator-N en docker ; en natif, OSMO_OP_IDS,
# puis les netns osmo-opN, puis N_OPERATORS de globals.conf, sinon 1.
mapfile -t OPS < <(osmo_ops)
N_OPS=${#OPS[@]}

# Hub : conteneur osmo-inter-stp en docker ; en natif, cette machine SI elle
# porte le role interstp. Sur un noeud operateur natif, le hub est ailleurs :
# osmo_hub rend 1 et osmo_hub_hint dit quoi lancer, et ou.
HAS_INTER_STP=0
HUB_NODE="$INTER_STP"
if HUB_NODE="$(osmo_hub)"; then
    HAS_INTER_STP=1
else
    HUB_NODE="$INTER_STP"
fi

# HUB_REMOTE_IP - adresse du hub quand il ne tourne PAS ici : WAN docker
# (start-docker.sh --wan/--hub-ip injecte OSMO_HUB_IP dans les conteneurs), ou noeud
# operateur natif. Vide si le hub est local, ou si aucune adresse n'est connue.
HUB_REMOTE_IP="$(osmo_hub_is_remote 2>/dev/null || true)"

# HUB_EXPECTED - "un hub devrait exister" (≠ "je peux l'interroger").
# Un hub distant reste attendu des qu'une adresse est connue - OSMO_HUB_IP de
# /etc/osmo-role en natif, l'adresse --hub-ip vue par les conteneurs en docker
# WAN : la route par defaut du STP local vers ce hub, elle, est parfaitement
# verifiable d'ici, meme si l'etat interne du hub ne l'est pas. En docker SANS
# WAN les deux notions se confondent toujours : pas de conteneur, pas de hub.
HUB_EXPECTED=$HAS_INTER_STP
if [ "$HUB_EXPECTED" -eq 0 ] && [ -n "$HUB_REMOTE_IP" ]; then
    HUB_EXPECTED=1
fi

# Nombre d'AS/ASP attendus sur le hub. En docker c'est le nombre d'operateurs
# LOCAUX, qui sont exactement ceux qui s'attachent au hub. En natif le hub sert
# des operateurs DISTANTS : aucun inventaire local ne dit combien il devrait y
# en avoir - on juge alors sur la presence, pas sur un total invente.
EXPECT_AS=0
[ "$MODE" = docker ] && EXPECT_AS=$N_OPS

if osmo_is_docker; then
    MODE_LABEL="docker (conteneurs osmo-operator-N)"
elif osmo_in_container; then
    MODE_LABEL="natif (dans le conteneur ${HOSTNAME:-?})"
else
    MODE_LABEL="natif (cette machine)"
fi
```

```
# Ligne de cadre : la largeur est calculee en CARACTERES (${#s}) et non en
# octets - "printf %-58s" compte les octets et decale d'un cran par accent.
boxline() {
    local s="$1" n
    n=$(( 58 - ${#s} )); [ "$n" -lt 0 ] && n=0
    echo -e "${CYAN}║${s}$(printf '%*s' "$n" '')║${NC}"
}

echo ""
echo -e "${CYAN}╔══════════════════════════════════════════════════════════╗${NC}"
boxline "          SS7 Health Check - $(date '+%H:%M:%S')"
boxline "          mode : ${MODE_LABEL}"
if [ -n "$HUB_REMOTE_IP" ]; then
    boxline "          ${N_OPS} operateur(s)  inter-stp : distant ${HUB_REMOTE_IP}"
else
    boxline "          ${N_OPS} operateur(s)  inter-stp=${HAS_INTER_STP}"
fi
echo -e "${CYAN}╚══════════════════════════════════════════════════════════╝${NC}"

if [ "$N_OPS" -eq 0 ] && [ "$HAS_INTER_STP" -eq 0 ] && [ -z "$HUB_REMOTE_IP" ]; then
    if [ "$MODE" = docker ]; then
        echo -e "${RED}Aucun container trouve${NC}"
    else
        echo -e "${RED}Aucun operateur ni inter-STP sur cette machine${NC}"
    fi
    exit 1
fi

# ═══════════════════════════════════════════════════════════════
# CHECK 1 : Inter-STP (hub central)
# ═══════════════════════════════════════════════════════════════
if [ "$HAS_INTER_STP" -eq 1 ]; then
    # [2026-08-31] Le point code etait ecrit en dur "0.23.0", comme celui des
    # operateurs l etait en "${op_id}.23.2". Le hub est en 0.0.0 (c est ce que
    # start-docker.sh annonce au lancement, et ce que porte osmo-multi.conf) : le
    # bandeau affichait donc une adresse qui n est celle de personne.
    banner "INTER-STP (hub SS7) - $(osmo_hub_pc 2>/dev/null || echo 0.0.0)"

    if ! vty_available "$HUB_NODE" 4239; then
        fail "VTY inter-STP (4239) inaccessible"
    else
        ok "VTY accessible"

        # Verifier que l'inter-STP voit tous les operateurs comme AS
        as_output=$(vty_cmd "$HUB_NODE" 4239 "show cs7 instance 0 as all")
        n_as_total=$(echo "$as_output" | grep -acE 'as-op[0-9]+' || true)
        n_as_active=$(echo "$as_output" | grep -acE 'AS_ACTIVE' || true)
        info "AS declares sur le hub : ${n_as_total}"

        if [ "$EXPECT_AS" -gt 0 ]; then
            if [ "$n_as_active" -eq "$EXPECT_AS" ]; then
                ok "AS : ${n_as_active}/${EXPECT_AS} actifs (tous)"
            elif [ "$n_as_active" -gt 0 ]; then
                warn "AS : ${n_as_active}/${EXPECT_AS} actifs"
            else
                fail "AS : aucun actif"
            fi
        elif [ "$n_as_active" -gt 0 ]; then
            # Natif : les operateurs sont des machines distantes, le total
            # attendu n'est pas connaissable d'ici.
            ok "AS : ${n_as_active} actif(s) (total attendu inconnu en natif)"
        else
            fail "AS : aucun actif"
        fi
```

```
        # Verifier les ASP connectes
        asp_output=$(vty_cmd "$HUB_NODE" 4239 "show cs7 instance 0 asp")
        n_asp_active=$(echo "$asp_output" | grep -acE 'ASP_ACTIVE' || true)

        if [ "$EXPECT_AS" -gt 0 ]; then
            if [ "$n_asp_active" -eq "$EXPECT_AS" ]; then
                ok "ASP : ${n_asp_active}/${EXPECT_AS} connectes (tous)"
            else
                warn "ASP : ${n_asp_active}/${EXPECT_AS} connectes"
            fi
        elif [ "$n_asp_active" -gt 0 ]; then
            ok "ASP : ${n_asp_active} connecte(s) (total attendu inconnu en natif)"
        else
            warn "ASP : aucun connecte"
        fi

        # Verifier l'absence de PROHIB (routes bloquees)
        route_output=$(vty_cmd "$HUB_NODE" 4239 "show cs7 instance 0 route")
        n_prohib=$(echo "$route_output" | grep -aciE 'prohib' || true)

        if [ "$n_prohib" -eq 0 ]; then
            ok "Routes : aucun PROHIB"
        else
            fail "Routes : ${n_prohib} PROHIB detectees"
        fi
    fi
elif [ -n "$HUB_REMOTE_IP" ]; then
    # Hub DISTANT (WAN) : "conteneur absent" serait un contresens - le hub
    # tourne, ailleurs. Ce qui est observable d'ici, c'est le LIEN vers lui.
    banner "INTER-STP (hub distant ${HUB_REMOTE_IP})"

    # Le lien operateur → hub est du M3UA sur SCTP (osmo-stp.cfg : "asp
    # asp-to-inter 2908 2910 m3ua"). Une sonde TCP sur 2908 rendrait
    # "Connection refused" sur un lien parfaitement monte : c'est l'association
    # SCTP, vue depuis les noeuds, qui fait foi.
    hub_assoc="$(osmo_hub_assoc "$HUB_REMOTE_IP" 2>/dev/null)" && hub_rc=0 || hub_rc=$?
    case "$hub_rc" in
        0) ok "Lien M3UA/SCTP → ${HUB_REMOTE_IP}:${OSMO_HUB_M3UA_PORT} : ${hub_assoc} noeud(s) associe(s)" ;;
        1) fail "Aucune association SCTP vers ${HUB_REMOTE_IP}:${OSMO_HUB_M3UA_PORT} (${hub_assoc}) - hub arrete,
         ↪  filtre, ou ASP shutdown" ;;
        *) skip "Association SCTP vers ${HUB_REMOTE_IP} non sondable ici (ss --sctp absent des noeuds)" ;;
    esac

    # Pas de faux echec sur l'etat INTERNE du hub : son VTY n'ecoute que sur SA
    # boucle locale. La phrase vient de _mode.sh et nomme l'IP reelle.
    skip "$(osmo_hub_hint)"
else
    banner "INTER-STP"
    if [ "$MODE" = docker ]; then
        skip "Container ${INTER_STP} absent"
    else
        # Pas de faux echec : le hub est peut-etre parfaitement vivant, mais son
        # VTY n'ecoute que sur SA boucle locale. La phrase vient de _mode.sh et
        # nomme l'IP reelle du hub quand /etc/osmo-role la donne.
        skip "$(osmo_hub_hint)"
    fi
fi

# ═══════════════════════════════════════════════════════════════════
# CHECK 2 : Chaque operateur
# ═══════════════════════════════════════════════════════════════════
for op in ${OPS[@]+"${OPS[@]}"}; do
    op_id="$op"
    # osmo_node garde l'etiquette "osmo-operator-N" meme en natif : c'est le
    # nom d'affichage que les autres checks correlent.
    container="$(osmo_node "$op")"
```

```
# ── LE LIBELLE DIT CE QUI EST, PAS UNE FORMULE ──────────────────────────
# [2026-08-31] Le point code etait ECRIT EN DUR comme "${op_id}.23.2" -
# d ou "STP 1.23.2", "2.23.2", "3.23.2", des adresses qui ne sont celles de
# personne (le plan local donne 1.<op>.2). Et le nom affichait toujours
# "osmo-operator-N", y compris pour l operateur NATIF, qui n est pas un
# conteneur : on lisait "OPERATEUR 1 (osmo-operator-1)" pour une pile qui
# tourne sur l hote, et on cherchait un conteneur inexistant.
# On lit desormais la topologie (osmo-multi.conf) quand elle existe.
_pc="$(osmo_op_pc "$op_id" 2>/dev/null || true)"
if [ "$(osmo_op_mode "$op_id" 2>/dev/null)" = "native" ]; then
    _label="natif (hote)"
else
    _label="$container"
fi
banner "OPERATEUR ${op_id} (${_label}) - STP ${_pc:-inconnu}"

# ── STP local (4239) ────────────────────────────────────────────────────
echo -e "  ${BOLD}STP local :${NC}"
if ! vty_available "$op" 4239; then
    fail "STP VTY (4239) inaccessible"
else
    # Verifier la connexion vers l'inter-STP
    as_out=$(vty_cmd "$op" 4239 "show cs7 instance 0 as all")
    inter_state=$(echo "$as_out" | grep -aoP 'as-inter\s+\K(AS_\w+)' || echo "ABSENT")

    if [[ "$inter_state" == "AS_ACTIVE" ]]; then
        ok "as-inter → hub : ACTIVE"
    elif [[ "$inter_state" == "ABSENT" ]] && [ "$HUB_EXPECTED" -eq 1 ]; then
        fail "as-inter absent (pas de connexion au hub)"
    elif [[ "$inter_state" == "ABSENT" ]]; then
        skip "as-inter : ignore (aucun inter-STP connu depuis ce noeud)"
    else
        warn "as-inter : ${inter_state}"
    fi

    # Verifier les ASP locaux (MSC/BSC)
    asp_out=$(vty_cmd "$op" 4239 "show cs7 instance 0 asp")
    n_asp_ok=$(echo "$asp_out" | grep -acE 'ASP_ACTIVE' || true)

    if [ "$n_asp_ok" -ge 2 ]; then
        ok "ASP locaux : au moins 2 actifs (MSC+BSC)"
    elif [ "$n_asp_ok" -eq 1 ]; then
        warn "ASP locaux : 1 seul actif (manque MSC ou BSC)"
    else
        fail "ASP locaux : aucun actif"
    fi

    # Verifier la route par defaut vers l'inter-STP (element cle).
    # Verifiable des qu'un hub est ATTENDU : le hub distant d'un lab natif
    # se voit dans la table de routage du STP local, meme si son VTY a lui
    # est hors de portee.
    route_out=$(vty_cmd "$op" 4239 "show cs7 instance 0 route")
    has_default=$(echo "$route_out" | grep -acE '0\.0\.0/0.*avail' || true)

    if [ "$has_default" -gt 0 ] && [ "$HUB_EXPECTED" -eq 1 ]; then
        ok "Route par defaut → inter-STP : presente et avail"
    elif [ "$HUB_EXPECTED" -eq 1 ]; then
        fail "Route par defaut → inter-STP absente ou non avail"
    elif [ "$MODE" != docker ]; then
        skip "Route par defaut → inter-STP : ignore (natif, aucun hub declare ici)"
    fi

    # Verifier l'absence de PROHIB
    n_prohib=$(echo "$route_out" | grep -aciE 'prohib' || true)
    if [ "$n_prohib" -gt 0 ]; then
        fail "Routes PROHIB detectees : ${n_prohib}"
```

```
        fi

        # Compter les routes dynamiques (juste pour info)
        n_dyn=$(echo "$route_out" | grep -ac 'dyn' || true)
        info "Routes dynamiques apprises : ${n_dyn} (normal: devrait augmenter avec le nombre d'operateurs)"
    fi

    # En mode quick, on passe au prochain operateur
    if [ "$QUICK" -eq 1 ]; then
        continue
    fi

    # ── MSC (4254) ──────────────────────────────────────────────
    echo -e "  ${BOLD}MSC :${NC}"
    if ! vty_available "$op" 4254; then
        fail "MSC VTY (4254) inaccessible"
    else
        msc_asp=$(vty_cmd "$op" 4254 "show cs7 instance 0 asp")
        msc_active=$(echo "$msc_asp" | grep -acE 'ASP_ACTIVE' || true)
        [ "$msc_active" -gt 0 ] && ok "MSC→STP : ASP_ACTIVE" || fail "MSC→STP : pas de ASP actif"

        # SCCP - osmo_qgrep, et non "grep -aq" : en bout de tube, grep -aq sort
        # au premier match et tue le producteur en SIGPIPE (141 sous pipefail).
        msc_sccp=$(vty_cmd "$op" 4254 "show cs7 instance 0 sccp users")
        echo "$msc_sccp" | osmo_qgrep -aE 'SSN 254' && ok "SCCP SSN 254 (MSC-A) enregistre" || fail "SCCP SSN 254
         ↪  absent"
    fi

    # ── BSC (4242) ──────────────────────────────────────────────
    echo -e "  ${BOLD}BSC :${NC}"
    if ! vty_available "$op" 4242; then
        fail "BSC VTY (4242) inaccessible"
    else
        bsc_asp=$(vty_cmd "$op" 4242 "show cs7 instance 0 asp")
        bsc_active=$(echo "$bsc_asp" | grep -acE 'ASP_ACTIVE' || true)
        [ "$bsc_active" -gt 0 ] && ok "BSC→STP : ASP_ACTIVE" || fail "BSC→STP : pas de ASP actif"

        # SCCP
        bsc_sccp=$(vty_cmd "$op" 4242 "show cs7 instance 0 sccp users")
        echo "$bsc_sccp" | osmo_qgrep -aE 'SSN 254' && ok "SCCP SSN 254 (BSC) enregistre" || fail "SCCP SSN 254
         ↪  absent"

        # BTS state (optionnel)
        bsc_bts=$(vty_cmd "$op" 4242 "show bts 0" 2>/dev/null || echo "")
        if echo "$bsc_bts" | osmo_qgrep -aE "Oper 'Enabled'.*Avail 'OK'"; then
            ok "BTS 0 : OK"
        fi
    fi

    # ── HLR (4258) ──────────────────────────────────────────────
    echo -e "  ${BOLD}HLR :${NC}"
    if ! vty_available "$op" 4258; then
        fail "HLR VTY (4258) inaccessible"
    else
        hlr_gsup=$(vty_cmd "$op" 4258 "show gsup-connections")
        vlr_conn=$(echo "$hlr_gsup" | grep -acE "VLR" || true)
        [ "$vlr_conn" -gt 0 ] && ok "GSUP : VLR connecte" || fail "GSUP : VLR absent"
    fi
done

# ═══════════════════════════════════════════════════════════════
# CHECK 3 : Matrice de connectivite simple
# ═══════════════════════════════════════════════════════════════
# La matrice croise les operateurs LOCAUX entre eux via le hub : elle a du sens
# des qu'un hub est ATTENDU (local, ou distant en WAN - le trafic op1↔op2 le
# traverse dans les deux cas) et qu'il y a plus d'un operateur ici. Sur un
```

```
# mono-operateur elle se reduirait a une case "self" - on l'annonce ignoree
# plutot que de la dessiner vide, et on renvoie vers le check qui, lui, sait
# regarder entre machines.
if [ "$HUB_EXPECTED" -eq 1 ] && [ "$N_OPS" -gt 1 ]; then
    if [ -n "$HUB_REMOTE_IP" ]; then
        banner "MATRICE DE CONNECTIVITE (via inter-STP distant ${HUB_REMOTE_IP})"
    else
        banner "MATRICE DE CONNECTIVITE (via inter-STP)"
    fi

    printf "  %-8s" ""
    for j in "${OPS[@]}"; do printf "  Op%-3s" "$j"; done
    echo ""

    for op in "${OPS[@]}"; do
        src_id="$op"
        printf "  Op%-5s" "${src_id}"

        # On ne teste que la connectivite de base : l'operateur a-t-il une route par defaut ?
        route_out=$(vty_cmd "$op" 4239 "show cs7 instance 0 route" 2>/dev/null || echo "")
        has_default=$(echo "$route_out" | grep -acE '0\.0\.0/0.*avail' || true)

        for j in "${OPS[@]}"; do
            if [ "$j" -eq "$src_id" ]; then
                printf "  ${CYAN}%-5s${NC}" "self"
            elif [ "$has_default" -gt 0 ]; then
                # Si route par defaut, on suppose que la connectivite est possible via l'inter-STP
                printf "  ${GREEN}%-5s${NC}" "via"
            else
                printf "  ${RED}%-5s${NC}" "FAIL"
                FAIL=$((FAIL+1))
            fi
        done
        echo ""
    done
    echo ""
    echo "  Note: 'via' signifie que l'operateur a une route par defaut vers l'inter-STP"
    echo "  Les routes dynamiques specifiques sont apprises au fur et a mesure"
elif [ "$MODE" != docker ]; then
    banner "MATRICE DE CONNECTIVITE (via inter-STP)"
    skip "ignore (natif) : ${N_OPS} operateur(s) local(aux) - la connectivite entre operateurs passe par le WAN,
     ↪  voir checks/wan_ss7_check.sh"
fi

# ═══════════════════════════════════════════════════════════════════
# RESUME
# ═══════════════════════════════════════════════════════════════════
echo ""
echo -e "$(printf '═%.0s' {1..60})"

if [ "$FAIL" -eq 0 ]; then
    echo -e "  ${GREEN}${BOLD}SS7 OK${NC}  -  ${GREEN}${PASS} pass${NC}  ${YELLOW}${WARN} warn${NC}  ${SKIP} skip"
elif [ "$FAIL" -le 3 ]; then
    echo -e "  ${YELLOW}${BOLD}SS7 DEGRADE${NC}  -  ${GREEN}${PASS} pass${NC}  ${RED}${FAIL} fail${NC}
     ↪  ${YELLOW}${WARN} warn${NC}  ${SKIP} skip"
else
    echo -e "  ${RED}${BOLD}SS7 KO${NC}  -  ${GREEN}${PASS} pass${NC}  ${RED}${FAIL} fail${NC}  ${YELLOW}${WARN}
     ↪  warn${NC}  ${SKIP} skip"
fi
echo -e "$(printf '═%.0s' {1..60})"
echo ""

exit "$FAIL"
```

3.43 /opt/GSM/osmo-operator/checks/vty-debug-dump.sh

18243 octets, 479 lignes → 479 lignes

```
#!/bin/bash
# vty-debug-dump.sh
#
# Parcourt tous les VTY Osmocom de chaque operateur ET de l'inter-STP,
# active le niveau de log DEBUG, execute les commandes "show" pertinentes,
# et consolide dans un log.
#
# DEUX MONDES, detection automatique (checks/_mode.sh) :
#   docker : un conteneur osmo-operator-N par operateur, le hub SS7 est le
#            conteneur osmo-inter-stp ; on interroge les VTY par "docker exec".
#   natif  : ISO, VM ou machine nue - un seul operateur (sauf topologie netns),
#            les VTY sont en telnet sur 127.0.0.1. Ce qui n'existe que sous
#            docker (le conteneur inter-STP) est IGNORE EXPLICITEMENT : un dump
#            qui inventerait un hub absent se lirait comme une panne SS7.
#
# L'etiquette "container=osmo-operator-N" est ecrite dans le dump DANS LES
# DEUX MODES : checks/operator_summary.sh y accroche toutes ses extractions
# (grep -A20 "container=..."). En natif c'est un nom d'affichage, une cle de
# correlation, pas un objet interrogeable.
#
# VTY couverts par operateur :
#   OsmoSTP          4239
#   OsmoHLR          4258
#   OsmoMGW          4243
#   OsmoMSC          4254
#   OsmoBSC          4242
#   OsmoBTS          4241
#   OsmoPCU          4240
#   OsmoSGSN         4245
#   OsmoGGSN         4260
#   OsmoSIP-conn     4255
#
# VTY de l'inter-STP (hub SS7) :
#   OsmoSTP (hub)    4239
#
# Usage :
#   sudo ./vty-debug-dump.sh [--out /chemin/log.txt] [--docker|--native]

set -euo pipefail
GREEN='\033[0;32m'; CYAN='\033[0;36m'; YELLOW='\033[1;33m'; RED='\033[0;31m'; NC='\033[0m'

# Memorise AVANT le sourcing : la bibliotheque ecrit dans OSMO_MODE (memoisation),
# on ne saurait plus apres coup si la valeur venait de l'environnement.
MODE_ENV="${OSMO_MODE:-}"

# ── Bibliotheque commune docker/natif ──────────────────────────────
# Chemin RELATIF au script (BASH_SOURCE) : le depot et l'ISO n'ont pas la meme
# racine, et l'ISO embarque checks/ en entier - donc _mode.sh est toujours a
# cote de ce fichier, jamais a un chemin absolu du depot.
_here="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
for _c in "$_here/_mode.sh" /opt/GSM/osmo-operator/checks/_mode.sh; do
    [ -r "$_c" ] && { . "$_c"; break; }
done
command -v osmo_mode >/dev/null || { echo "checks/_mode.sh introuvable" >&2; exit 1; }

# ── Options ────────────────────────────────────────────────────────
# --out reste accepte A N'IMPORTE QUELLE POSITION : operator_summary.sh --live
# lance "./vty-debug-dump.sh --out /tmp/vty-live-$$.txt" et doit continuer a
# marcher meme precede de --native.
OUTFILE="${OUTFILE:-/tmp/vty-debug-dump.txt}"
MODE_FORCED=0
while [ $# -gt 0 ]; do
```

```
    case "$1" in
        --out)
            [ -n "${2:-}" ] || { echo "--out attend un chemin" >&2; exit 1; }
            OUTFILE="$2"; shift 2 ;;
        --out=*)
            OUTFILE="${1#*=}"; shift ;;
        --docker|--native|--mode=*)
            osmo_mode_force "$1" || { echo "mode inconnu : $1" >&2; exit 1; }
            MODE_FORCED=1; shift ;;
        -h|--help)
            echo "Usage : sudo ./vty-debug-dump.sh [--out /chemin/log.txt] [--docker|--native]"
            echo "  --docker  force le mode conteneurs (docker exec)"
            echo "  --native  force le mode natif (VTY locaux sur 127.0.0.1)"
            echo "  sans option : detection automatique"
            exit 0 ;;
        *)
            echo "option inconnue : $1" >&2
            echo "Usage : sudo ./vty-debug-dump.sh [--out /chemin/log.txt] [--docker|--native]" >&2
            exit 1 ;;
    esac
done

INTER_STP_CONTAINER="$(osmo_node hub)"   # "osmo-inter-stp" dans les deux mondes

# ── Tables VTY : deux jeux (operateur / inter-stp) ──────────────────────────
declare -A OP_VTY_NAME OP_VTY_SHOW
declare -A ISTP_VTY_NAME ISTP_VTY_SHOW

register_op_vty() {
    local port="$1" name="$2" show_cmds="$3"
    OP_VTY_NAME[$port]="$name"
    OP_VTY_SHOW[$port]="$show_cmds"
}

register_istp_vty() {
    local port="$1" name="$2" show_cmds="$3"
    ISTP_VTY_NAME[$port]="$name"
    ISTP_VTY_SHOW[$port]="$show_cmds"
}

# ── Commandes show operateur - enrichies ────────────────────────────────────

register_op_vty 4239 "OsmoSTP" "\
show cs7 instance 0 users|\
show cs7 instance 0 as all|\
show cs7 instance 0 asp|\
show cs7 instance 0 route|\
show cs7 instance 0 sccp users|\
show cs7 instance 0 sccp connections|\
show cs7 instance 0 sccp timers|\
show logging level|\
show talloc-context all brief"

register_op_vty 4258 "OsmoHLR" "\
show subscriber all|\
show subscriber count|\
show gsup-connections|\
show logging level|\
show talloc-context all brief"

register_op_vty 4243 "OsmoMGW" "\
show mgcp stats|\
show mgcp active|\
show mgcp endpoint Name trunk-0|\
show logging level|\
show talloc-context all brief"
```

```
register_op_vty 4254 "OsmoMSC" "\
show network|\
show subscriber all|\
show subscriber count|\
show transaction all|\
show call-legs|\
show connection|\
show sgs-connections|\
show cs7 instance 0 users|\
show cs7 instance 0 asp|\
show cs7 instance 0 as all|\
show cs7 instance 0 sccp users|\
show cs7 instance 0 sccp connections|\
show logging level|\
show stats|\
show talloc-context all brief"

register_op_vty 4242 "OsmoBSC" "\
show network|\
show bts all|\
show bts 0|\
show trx 0 0|\
show timeslot all|\
show lchan all|\
show lchan summary|\
show paging all|\
show paging-group all|\
show conns|\
show cs7 instance 0 users|\
show cs7 instance 0 asp|\
show cs7 instance 0 as all|\
show cs7 instance 0 sccp users|\
show cs7 instance 0 sccp connections|\
show logging level|\
show stats|\
show talloc-context all brief"

register_op_vty 4241 "OsmoBTS" "\
show bts 0|\
show trx 0 0|\
show ts all|\
show lchan all|\
show lchan summary|\
show phy 0 inst all|\
show logging level|\
show talloc-context all brief"

register_op_vty 4240 "OsmoPCU" "\
show bts all|\
show tbf all|\
show ms all|\
show logging level|\
show talloc-context all brief"

register_op_vty 4245 "OsmoSGSN" "\
show sgsn|\
show mm-context all|\
show pdp-context all|\
show subscriber cache|\
show ns|\
show ns stats|\
show ns entities|\
show bssgp|\
show logging level|\
show stats|\
```

```
show talloc-context all brief"

register_op_vty 4260 "OsmoGGSN" "\
show ggsn|\
show pdp-context all|\
show apn all|\
show logging level|\
show talloc-context all brief"

register_op_vty 4255 "OsmoSIPconn" "\
show logging level|\
show talloc-context all brief"

OP_VTY_PORTS=(4239 4258 4243 4254 4242 4241 4240 4245 4260 4255)

# ── Commandes show inter-STP - balayage instances 0..9 ──────────────────
# L'inter-STP hub peut avoir N instances CS7 (une par lien operateur).

ISTP_SHOW_CMDS="show cs7 instance 0 users|\
show cs7 instance 0 as all|\
show cs7 instance 0 as all active|\
show cs7 instance 0 asp|\
show cs7 instance 0 route|\
show cs7 instance 0 sccp users|\
show cs7 instance 0 sccp connections|\
show cs7 instance 0 sccp timers|\
show cs7 instance 1 users|\
show cs7 instance 1 as all|\
show cs7 instance 1 asp|\
show cs7 instance 1 route|\
show cs7 instance 1 sccp users|\
show cs7 instance 1 sccp connections|\
show cs7 instance 2 users|\
show cs7 instance 2 as all|\
show cs7 instance 2 asp|\
show cs7 instance 2 route|\
show cs7 instance 2 sccp users|\
show cs7 instance 2 sccp connections|\
show cs7 instance 3 users|\
show cs7 instance 3 as all|\
show cs7 instance 3 asp|\
show cs7 instance 3 route|\
show cs7 instance 4 as all|\
show cs7 instance 4 asp|\
show cs7 instance 4 route|\
show cs7 instance 5 as all|\
show cs7 instance 5 asp|\
show cs7 instance 5 route|\
show cs7 instance 6 as all|\
show cs7 instance 6 asp|\
show cs7 instance 6 route|\
show cs7 instance 7 as all|\
show cs7 instance 7 asp|\
show cs7 instance 7 route|\
show cs7 instance 8 as all|\
show cs7 instance 8 asp|\
show cs7 instance 8 route|\
show cs7 instance 9 as all|\
show cs7 instance 9 asp|\
show cs7 instance 9 route|\
show logging level|\
show stats|\
show talloc-context all brief"

# Le hub peut avoir ete pousse hors de 4239 (co-localise avec le STP
# operateur, cf. start-interstp.sh) : relire son VTY dans sa config. En
```

```
# docker le hub reste a 4239 (interroge par docker exec) et le fichier
# hote est absent : sed rend vide, on garde 4239.
ISTP_HUB_VTY=4239
_hubp="$(sed -n 's/^ *bind 127\.0\.0\.1 \([0-9]*\)$/\1/p' /etc/osmocom/osmo-stp-interop.cfg 2>/dev/null | head
↪  -1)"
[ -n "$_hubp" ] && ISTP_HUB_VTY="$_hubp"
register_istp_vty "$ISTP_HUB_VTY" "OsmoSTP-hub" "$ISTP_SHOW_CMDS"

ISTP_VTY_PORTS=("$ISTP_HUB_VTY")

# ── Inventaire des noeuds - docker ou natif ──────────────────────────────
MODE="$(osmo_mode)"
# Un OSMO_MODE d'environnement qui ne vaut ni docker ni native est ignore par la
# bibliotheque (elle redetecte) : on ne l'annonce donc pas comme un forcage.
if   [ "$MODE_FORCED" -eq 1 ]; then MODE_SRC="force par option"
elif [ "$MODE_ENV" = docker ] || [ "$MODE_ENV" = native ]; then MODE_SRC="force par OSMO_MODE"
else MODE_SRC="detection automatique"
fi

# osmo_ops : conteneurs osmo-operator-N en docker ; OSMO_OP_IDS / netns /
# N_OPERATORS (sinon 1) en natif. Deja trie numeriquement et dedoublonne, d'ou
# la disparition du "sort -t - -k 3 -n".
mapfile -t OPS < <(osmo_ops)
OP_CONTAINERS=()
for op in "${OPS[@]:-}"; do
    [ -n "$op" ] && OP_CONTAINERS+=("$(osmo_node "$op")")
done

# Inter-STP : present SI on peut l'interroger D'ICI.
#   docker : le conteneur osmo-inter-stp tourne.
#   natif  : cette machine EST le hub (OSMO_ROLE=interstp, ou un osmo-stp lance
#            sur osmo-stp-interop.cfg). Si le hub est une autre machine, son VTY
#            n'ecoute que sur SON 127.0.0.1 : rien n'est dumpable d'ici, et on le
#            dit - un bloc "OsmoSTP-hub" vide serait lu comme un hub en panne.
HAS_INTER_STP=0
osmo_hub >/dev/null 2>&1 && HAS_INTER_STP=1

if [ ${#OP_CONTAINERS[@]} -eq 0 ] && [ "$HAS_INTER_STP" -eq 0 ]; then
    echo -e "${RED}Aucun container osmo-operator-N ni ${INTER_STP_CONTAINER} trouve.${NC}"
    if [ "$MODE" = docker ]; then
        echo -e "  Containers actifs : $(docker ps --format '{{.Names}}' | tr '\n' ' ')"
    else
        echo -e "  Mode natif : aucun operateur declare (OSMO_OP_IDS, netns ${OSMO_NETNS_PREFIX}N, N_OPERATORS)."
    fi
    exit 1
fi

N_OPS=${#OP_CONTAINERS[@]}
# Le mode est annonce AVANT tout le reste : un diagnostic qui ne dit pas dans
# quel monde il a regarde n'est pas exploitable.
echo -e "${CYAN}Mode : ${MODE} (${MODE_SRC})${NC}"
echo -e "${GREEN}=== VTY Debug Dump - ${N_OPS} operateur(s) + inter-stp=${HAS_INTER_STP} → ${OUTFILE} ===${NC}"
echo ""

# ── Vider / initialiser le fichier de sortie ─────────────────────────────
: > "$OUTFILE"
log() { echo -e "$*" | tee -a "$OUTFILE"; }
logsep() {
    local title="$1"
    log ""
    log "$(printf '=%.0s' {1..72})"
    log "  ${title}"
    log "$(printf '=%.0s' {1..72})"
}
logsubsep() {
    local title="$1"
```

```
    log ""
    log "  $(printf '─%.0s' {1..68})"
    log "  ${title}"
    log "  $(printf '─%.0s' {1..68})"
}

log "VTY DEBUG DUMP - $(date '+%Y-%m-%d %H:%M:%S')"
log "Mode       : ${MODE} (${MODE_SRC})"
log "Operateurs : ${N_OPS}   Containers : ${OP_CONTAINERS[*]:-aucun}"
if   [ "$HAS_INTER_STP" -eq 1 ]; then log "Inter-STP  : present"
elif [ "$MODE" = docker        ]; then log "Inter-STP  : absent"
else                                    log "Inter-STP  : ignore (natif) - $(osmo_hub_hint)"
fi

# ── Interroger un VTY sur un noeud ──────────────────────────────────────
# $1=<op> (numero d'operateur ou "hub")  $2=port  $3=nom  $4=commandes show
# osmo_vty_up / osmo_vty remplacent le couple "docker exec ... echo
# >/dev/tcp/127.0.0.1/PORT" + "docker exec ... nc" : meme dialogue, mais local
# en natif. Le champ "container=" du dump garde l'etiquette du noeud.
query_vty() {
    local op="$1"
    local port="$2"
    local name="$3"
    local show_str="$4"
    local container
    container="$(osmo_node "$op")"

    # Test de connectivite rapide. En natif la sonde passive (ss) passe d'abord :
    # elle n'ouvre aucune session VTY.
    if ! osmo_vty_up "$op" "$port"; then
        echo -e "  ${YELLOW}[SKIP]${NC} ${name} (port ${port}) - non disponible"
        log "  [SKIP] ${name} port ${port} - non disponible"
        return 1
    fi

    echo -ne "  ${CYAN}${name}${NC} (${port}) ... "

    # Construire le script VTY : passage en DEBUG, puis les show, puis end.
    local -a VTY_CMDS=(enable "logging level all debug" "logging filter all 1")
    IFS='|' read -ra SHOW_CMDS <<< "$show_str"
    for cmd in "${SHOW_CMDS[@]}"; do
        VTY_CMDS+=("$cmd")
    done
    VTY_CMDS+=(end)

    # Temporisations d'origine conservees telles quelles (nc -q2, 0,5 s avant
    # d'ecrire, 2 s de lecture) : ce sont de gros jeux de commandes, et changer
    # ces valeurs changerait la QUANTITE de sortie capturee - donc le compteur
    # de lignes affiche juste apres.
    local raw_output
    raw_output="$(OSMO_VTY_OPEN=0.5 OSMO_VTY_READ=2 OSMO_VTY_Q=2 \
                  osmo_vty "$op" "$port" "${VTY_CMDS[@]}" || true)"

    # Filtrer banniere/prompt VTY - meme motif et meme ordre qu'avant.
    # Sans argument : ce dump GARDE la ligne "Free Software lives", contrairement
    # a global_check.sh et ss7_check.sh.
    local clean_output
    clean_output="$(printf '%s\n' "$raw_output" | osmo_vty_clean)"

    local n_lines
    n_lines=$(echo "$clean_output" | wc -l)
    echo -e "${GREEN}${n_lines} lignes${NC}"

    {
        echo ""
        echo "  [${name}] port=${port}  container=${container}"
```

```
        echo "  Commandes show : $(echo "$show_str" | tr '|' ', ')"
        echo ""
        echo "$clean_output" | sed 's/^/    /'
    } >> "$OUTFILE"

    return 0
}

# ── Boucle principale : noeuds operateur ──────────────────────────────
TOTAL_OK=0
TOTAL_SKIP=0

for op in "${OPS[@]:-}"; do
    [ -n "$op" ] || continue
    container="$(osmo_node "$op")"
    logsep "Container : ${container}  -  $(date '+%H:%M:%S')"
    echo -e "${GREEN}--- ${container} ---${NC}"

    for port in "${OP_VTY_PORTS[@]}"; do
        logsubsep "${OP_VTY_NAME[$port]} (port ${port})"
        if query_vty "$op" "$port" \
                "${OP_VTY_NAME[$port]}" "${OP_VTY_SHOW[$port]}"; then
            TOTAL_OK=$(( TOTAL_OK + 1 ))
        else
            TOTAL_SKIP=$(( TOTAL_SKIP + 1 ))
        fi
    done
    echo ""
done

# ── Inter-STP ─────────────────────────────────────────────────────────
if [ "$HAS_INTER_STP" -eq 1 ]; then
    logsep "Container : ${INTER_STP_CONTAINER} (hub SS7)  -  $(date '+%H:%M:%S')"
    echo -e "${GREEN}--- ${INTER_STP_CONTAINER} ---${NC}"

    for port in "${ISTP_VTY_PORTS[@]}"; do
        logsubsep "${ISTP_VTY_NAME[$port]} (port ${port})"
        if query_vty hub "$port" \
                "${ISTP_VTY_NAME[$port]}" "${ISTP_VTY_SHOW[$port]}"; then
            TOTAL_OK=$(( TOTAL_OK + 1 ))
        else
            TOTAL_SKIP=$(( TOTAL_SKIP + 1 ))
        fi
    done
    echo ""
elif [ "$MODE" = docker ]; then
    log ""
    log "  [INFO] Container ${INTER_STP_CONTAINER} absent - skipped"
else
    # Natif sans hub local : ni OK ni echec, et surtout pas de compteur touche -
    # il n'y a rien a interroger ici, ce n'est pas une panne.
    echo -e "  ${YELLOW}${INTER_STP_CONTAINER} - ignore (natif)${NC}"
    echo -e "  ${YELLOW}  $(osmo_hub_hint)${NC}"
    log ""
    log "  [INFO] ${INTER_STP_CONTAINER} - ignore (natif) : $(osmo_hub_hint)"
fi

# ── Resume ────────────────────────────────────────────────────────────
logsep "RESUME - $(date '+%Y-%m-%d %H:%M:%S')"
log "  Mode                  : ${MODE} (${MODE_SRC})"
log "  VTY interroges        : ${TOTAL_OK}"
log "  VTY non disponibles   : ${TOTAL_SKIP}"
log "  Operateurs            : ${N_OPS}"
if   [ "$HAS_INTER_STP" -eq 1 ]; then log "  Inter-STP             : oui"
elif [ "$MODE" = docker        ]; then log "  Inter-STP             : non"
else                                   log "  Inter-STP             : ignore (natif)"
```

```
fi
log "  Fichier               : ${OUTFILE}"
log "  Taille                : $(wc -c < "$OUTFILE") octets  /  $(wc -l < "$OUTFILE") lignes"

echo ""
echo -e "${GREEN}╔══════════════════════════════════════════════════════════════╗${NC}"
echo -e "${GREEN}║  Dump complet → ${OUTFILE}${NC}"
echo -e "${GREEN}╚══════════════════════════════════════════════════════════════╝${NC}"
echo ""
echo -e "  ${CYAN}Consulter :${NC}  less ${OUTFILE}"
echo -e "  ${CYAN}Filtrer   :${NC}  grep -A5 'OsmoBSC' ${OUTFILE}"
echo -e "  ${CYAN}SS7 hub   :${NC}  grep -A10 'OsmoSTP-hub' ${OUTFILE}"
echo -e "  ${CYAN}Routes    :${NC}  grep -A20 'show cs7.*route' ${OUTFILE}"
echo -e "  ${CYAN}ASP etat  :${NC}  grep -iE 'asp|active|inactive|down' ${OUTFILE}"
echo -e "  ${CYAN}SCCP      :${NC}  grep -A10 'sccp' ${OUTFILE}"
echo -e "  ${CYAN}Erreurs   :${NC}  grep -iE 'error|fail|warn|prohib|timeout|refused' ${OUTFILE}"

# Copie locale si lance depuis l'hote
if [ -w "$(pwd)" ]; then
    cp "$OUTFILE" "$(pwd)/vty-debug-dump-$(date '+%Y%m%d-%H%M%S').txt" 2>/dev/null && \
        echo -e "  ${CYAN}Copie locale :${NC} $(pwd)/vty-debug-dump-$(date '+%Y%m%d-%H%M%S').txt" || true
fi
```

3.44 /opt/GSM/osmo-operator/checks/wan_ss7_check.sh

18530 octets, 333 lignes → 333 lignes

```
#!/bin/bash
# ═══════════════════════════════════════════════════════════════════════
# checks/wan_ss7_check.sh - etat du WAN, et etat du SS7 VIS-A-VIS du WAN
#
# checks/ss7_check.sh verifie le SS7 D'UN noeud (inter-STP, ASP, routes). Ce
# script-ci repond a la question qui vient juste apres : "et entre noeuds ?"
#
# CE QU'IL FAUT SAVOIR AVANT DE LIRE LA SORTIE
# --------------------------------------------
# Le WAN d'osmo-operator ne transporte PAS de SS7. Entre deux noeuds il n'y a que :
#     • SIP/RTP  (Asterisk ↔ Asterisk)  → la voix
#     • TCP 789x (sms-interop-relay)    → les SMS
# Le SS7/M3UA (osmo-stp, point codes 1.N.2, inter-STP 0.0.0) reste INTERNE a un
# noeud : l'ASP de chaque operateur pointe sur 172.20.0.10, une adresse du plan
# docker prive, injoignable depuis l'exterieur - et le catch-all "route 0.0.0
# 0.0.0 → as-inter" envoie tout PC inconnu vers CET inter-STP local.
#
# Ce n'est donc pas une panne a reparer, c'est la frontiere du montage : un
# appel inter-noeud est une TERMINAISON SIP, pas un relais MAP/ISUP. Ce script
# le VERIFIE au lieu de le supposer - et signale les deux cas ou l'on croit a
# tort avoir du SS7 inter-noeud :
#   1. deux noeuds qui reutilisent les MEMES point codes (ils le font tous :
#      1.1.2, 1.2.2... sont recalcules a l'identique sur chaque machine) ;
#   2. un port M3UA (2905/2908) expose sur l'interface publique.
#
# Usage : sudo checks/wan_ss7_check.sh [--verbose] [--conf /etc/osmo-wan.conf]
# ═══════════════════════════════════════════════════════════════════════
set -u

GREEN='\033[0;32m'; CYAN='\033[0;36m'; YELLOW='\033[1;33m'
RED='\033[0;31m'; BOLD='\033[1m'; NC='\033[0m'

VERBOSE=0
CONF="/etc/osmo-wan.conf"
while [ $# -gt 0 ]; do
    case "$1" in
        --verbose|-v) VERBOSE=1 ;;
```

```
        --conf)       CONF="${2:-}"; shift ;;
        --conf=*)     CONF="${1#*=}" ;;
        -h|--help)    sed -n '2,25p' "$0" | sed 's/^# \{0,1\}//'; exit 0 ;;
    esac
    shift
done

PASS=0; FAIL=0; WARN=0; SKIP=0
ok()   { echo -e "  ${GREEN}✓${NC} $*"; PASS=$((PASS+1)); }
fail() { echo -e "  ${RED}✗${NC} $*"; FAIL=$((FAIL+1)); }
warn() { echo -e "  ${YELLOW}⚠${NC} $*"; WARN=$((WARN+1)); }
skip() { echo -e "  ${YELLOW}-${NC} $*"; SKIP=$((SKIP+1)); }
info() { [ $VERBOSE -eq 1 ] && echo -e "    ${CYAN}|${NC} $*" || true; }
banner(){ echo ""; echo -e "${BOLD}$*${NC}"; printf '—%.0s' $(seq 1 64); echo ""; }

SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")/.." && pwd)"

echo -e "${CYAN}${BOLD}"
echo "╔══════════════════════════════════════════════════════════════╗"
echo "║   WAN - diagnostic voix / SMS / SS7 entre noeuds osmo-operator      ║"
echo "╚══════════════════════════════════════════════════════════════╝"
echo -e "${NC}"

# ── 0. Table des noeuds ──────────────────────────────────────────
banner "0. Table des noeuds WAN"
if [ ! -r "$SCRIPT_DIR/network/wan-nodes.sh" ]; then
    fail "network/wan-nodes.sh introuvable - depot incomplet"; exit 1
fi
# shellcheck source=network/wan-nodes.sh
. "$SCRIPT_DIR/network/wan-nodes.sh"

if ! wan_nodes_load "$CONF" 2>/dev/null; then
    fail "$CONF illisible - aucun WAN configure sur ce noeud"
    echo -e "     ${CYAN}sudo ./launch/start-docker.sh --wan${NC}  ou  ${CYAN}sudo ./start-native.sh --wan${NC}"
    exit 1
fi
[ "${WAN_NODE_ID:-0}" = "0" ] && wan_nodes_detect_self 2>/dev/null
if wan_nodes_validate 2>/dev/null; then ok "table valide : ${WAN_NODE_COUNT} noeud(s)"
else fail "table incoherente (voir messages ci-dessus)"; fi
wan_nodes_summary

LOCAL_IND="$(wan_local_ind)"
REMOTES=(); for id in "${WAN_NODE_LIST[@]}"; do [ "$id" = "${WAN_NODE_ID:-0}" ] || REMOTES+=("$id"); done
if [ "${#REMOTES[@]}" -eq 0 ]; then
    warn "table a un seul noeud : aucun pair, rien a verifier cote WAN"
    echo -e "     ${CYAN}Ajoutez des noeuds : sudo ./launch/start-docker.sh --wan${NC}"
    exit 0
fi

# ── Mode ─────────────────────────────────────────────────────────
if command -v docker >/dev/null 2>&1 && docker ps --format '{{.Names}}' 2>/dev/null | grep -q '^osmo-operator-1$';
 ↪ then
    MODE=docker
    N_OPS=$(docker ps --format '{{.Names}}' | grep -c '^osmo-operator-[0-9]*$')
else
    MODE=native; N_OPS=1
fi
info "mode=${MODE} operateurs locaux=${N_OPS}"

# ── Ce que porte un pair, et ou il ecoute ────────────────────────
# Ces deux fonctions repliquent ce que network/setup-wan-mesh.sh a REELLEMENT
# ecrit dans les configs. Sans elles, ce controle bouclait sur le nombre
# d'operateurs LOCAUX et cherchait chez chaque pair autant de trunks que nous en
# avons : des qu'un pair en portait moins - le montage un noeud par conteneur,
# soit le banc courant - il sortait "trunk wan_n2_op2 ABSENT, relancez
# setup-wan-mesh.sh" et rendait 1. Un echec faux, avec un conseil qui n'y
```

```
# changeait rien : le trunk manquant n'a jamais eu lieu d'exister.
peer_nops() {   # $1 = id de noeud
    local n="${WAN_NOPS[$1]:-}"
    if [[ "$n" =~ ^[1-9][0-9]*$ ]]; then echo "$n"; return; fi
    # Table sans le 4e champ (une table ecrite a la main, ou anterieure) : un
    # pair du backbone est un conteneur, donc un operateur ; ailleurs, on
    # suppose le pair monte comme nous. C'est la meme regle que node_nops dans
    # network/setup-wan-mesh.sh - il faut lire ce que le maillage a ecrit, pas
    # ce qu'on aurait souhaite qu'il ecrive.
    if peer_is_backbone "${WAN_IP[$1]:-}"; then echo 1; else echo "${N_OPS:-1}"; fi
}
# Un pair du backbone docker est joint a son adresse propre : le conteneur y
# ecoute sur 5060 et sur le port de base du relais SMS. Les 5080/5082 et 7891
# sont les ports PUBLIES sur l'hote, ils ne veulent rien dire de ce cote.
peer_is_backbone() { case "$1" in 172.20.0.*) return 0 ;; *) return 1 ;; esac; }

ast() { # $1=op  $2=commande
    if [ "$MODE" = docker ]; then docker exec "osmo-operator-$1" asterisk -rx "$2" 2>/dev/null
    else asterisk -rx "$2" 2>/dev/null; fi
}
inside() { # $1=op  $2=commande shell
    if [ "$MODE" = docker ]; then docker exec "osmo-operator-$1" bash -c "$2" 2>/dev/null
    else bash -c "$2" 2>/dev/null; fi
}

# ── 1. Joignabilite des pairs ──────────────────────────────────────
banner "1. Pairs - joignabilite IP et ports WAN"
for r in "${REMOTES[@]}"; do
    rip="${WAN_IP[$r]}"
    if ping -c1 -W2 "$rip" >/dev/null 2>&1; then ok "node ${r} ${rip} repond au ping"
    else warn "node ${r} ${rip} ne repond pas au ping (ICMP filtre ? noeud eteint ?)"; fi
    for i in $(seq 1 "$(peer_nops "$r")"); do
        if peer_is_backbone "$rip"; then sip=5060; sms=7890
        else sip=$(( 5080 + (i - 1) * 2 )); sms=$(( 7890 + i - 1 )); fi
        if command -v nc >/dev/null 2>&1; then
            nc -z -w2 "$rip" "$sms" 2>/dev/null \
                && ok "node ${r} op${i} : relais SMS ${rip}:${sms} ouvert" \
                || warn "node ${r} op${i} : relais SMS ${rip}:${sms} ferme - SMS WAN KO dans ce sens"
        else skip "nc absent : test TCP ${rip}:${sms} impossible"; fi
        info "SIP attendu sur ${rip}:${sip}/udp (UDP : pas de test fiable sans trafic)"
    done
done

# ── 2. Asterisk - trunks WAN ───────────────────────────────────────
banner "2. Asterisk - trunks WAN (voix)"
for i in $(seq 1 "$N_OPS"); do
    out="$(ast "$i" 'pjsip show endpoints')"
    if [ -z "$out" ]; then fail "op${i} : Asterisk ne repond pas"; continue; fi
    for r in "${REMOTES[@]}"; do
        for j in $(seq 1 "$(peer_nops "$r")"); do
            ep="wan_n${r}_op${j}"
            if echo "$out" | grep -q "$ep"; then
               state="$(ast "$i" "pjsip show aor $ep" | grep -iE 'Avail|Unavail|Contact' | head -3 | tr '\n' ' ')"
                if echo "$out" | grep -E "Endpoint:[[:space:]]+$ep" | grep -qi 'Unavailable'; then
                    warn "op${i} → ${ep} present mais Unavailable (pair injoignable)"
                else ok "op${i} → ${ep}"; fi
                info "$state"
            else
                fail "op${i} : trunk ${ep} ABSENT - relancez network/setup-wan-mesh.sh"
            fi
        done
    done
done

# ── 3. Dialplan - indicatifs ───────────────────────────────────────
banner "3. Dialplan - un indicatif route par pair"
```

```
for i in $(seq 1 "$N_OPS"); do
    for r in "${REMOTES[@]}"; do
        ind="${WAN_IND[$r]}"
        if ast "$i" "dialplan show wan_out" | grep -q "_${ind}"; then
            ok "op${i} : indicatif ${ind} → node ${r} present dans [wan_out]"
        else
            fail "op${i} : indicatif ${ind} absent de [wan_out]"
        fi
    done
    ast "$i" "dialplan show gsm_in" | grep -q "_${LOCAL_IND}" \
        && ok "op${i} : indicatif local ${LOCAL_IND} traite en interne" \
        || warn "op${i} : indicatif local ${LOCAL_IND} non traite - un appel 'plein numero' echouera"
done

# ── 4. SMS - routes et strip ──────────────────────────────────────────
banner "4. SMS - routes WAN et retrait d'indicatif"
for i in $(seq 1 "$N_OPS"); do
    conf="$(inside "$i" 'cat /etc/osmocom/sms-routing.conf')"
    if [ -z "$conf" ]; then fail "op${i} : sms-routing.conf illisible"; continue; fi
    for r in "${REMOTES[@]}"; do
        ind="${WAN_IND[$r]}"
        line="$(echo "$conf" | grep -E "^${ind}[0-9]+ =" | head -1)"
        if [ -z "$line" ]; then
            fail "op${i} : aucune route SMS pour l'indicatif ${ind}"
        elif echo "$line" | grep -q 'strip='; then
            ok "op${i} : route SMS ${ind} avec strip - le HLR distant verra le numero nu"
        else
            fail "op${i} : route SMS ${ind} SANS strip= - le noeud distant cherchera "${ind}10001" dans son HLR et
             ↪  repondra "not found""
        fi
    done
    # Le relais doit savoir lire strip= : une conf correcte sur un relais qui
    # l'ignore est exactement aussi cassee qu'une conf sans strip.
    if inside "$i" 'grep -q lookup_strip /opt/GSM/osmo-operator/scripts/sms-interop-relay.py 2>/dev/null || grep
     ↪  -q lookup_strip /etc/osmocom/sms-interop-relay.py 2>/dev/null || grep -rq lookup_strip /usr/local/bin
     ↪  2>/dev/null'; then
        ok "op${i} : le relais SMS gere strip="
    else
        warn "op${i} : relais SMS sans lookup_strip - version anterieure, SMS WAN muets"
    fi
done

# ── 5. SS7 - ce qui traverse, ce qui ne traverse pas ──────────────────
banner "5. SS7 - portee reelle vis-a-vis du WAN"
echo -e "  ${CYAN}Attendu : le SS7 s'arrete au bord du noeud. On verifie que c'est bien${NC}"
echo -e "  ${CYAN}le cas, et qu'on ne croit pas avoir un lien SS7 inter-noeud.${NC}"
echo ""

# Le hub change la donne : s'il y en a un, le SS7 TRAVERSE, et ce sont les
# point codes qu'il faut verifier - pas leur absence.
HUB_IP=""
[ -r /etc/osmo-role ] && HUB_IP="$(awk -F= '/^OSMO_HUB_IP=/{print $2}' /etc/osmo-role)"
# En docker, /etc/osmo-role n'existe pas sur l'hote : l'adresse du hub est
# injectee dans les conteneurs par start-docker.sh (-e OSMO_HUB_IP, cf. --hub-ip).
# Sans ce repli, un hub WAN parfaitement declare passait pour "un lien non
# maitrise" faute d'etre reconnu comme LE hub.
if [ -z "$HUB_IP" ]; then
    for _i in $(seq 1 "$N_OPS"); do
        HUB_IP="$(inside "$_i" 'printenv OSMO_HUB_IP' | tr -d '\r' | tail -1)"
        [ -n "$HUB_IP" ] && break
    done
fi

# 5a - ou pointe l'ASP local ?
for i in $(seq 1 "$N_OPS"); do
    cfg="$(inside "$i" 'cat /etc/osmocom/osmo-stp.cfg')"
```

```
    if [ -z "$cfg" ]; then skip "op${i} : osmo-stp.cfg illisible"; continue; fi
    remote="$(echo "$cfg" | awk '/asp asp-to-inter/{f=1} f&&/remote-ip/{print $2; exit}')"
    shut="$(echo "$cfg" | awk '/asp asp-to-inter/{f=1} f&&/shutdown/{print $0; exit}')"
    pc="$(echo "$cfg" | awk '/^ point-code/{print $2; exit}')"
    if [ -n "$HUB_IP" ] && [ "$remote" = "$HUB_IP" ]; then
        if echo "$shut" | grep -q 'no shutdown'; then
            ok "op${i} : ASP → hub inter-STP ${remote} (actif) - le SS7 traverse le WAN, PC ${pc}"
        else
            fail "op${i} : ASP → hub ${remote} mais ${BOLD}shutdown${NC} - aucun SS7 ne passera"
        fi
        # Un hub joignable ou non, ca ne se devine pas depuis la config. Mais
        # le lien est du M3UA sur SCTP : "nc -z" (TCP) rend "Connection
        # refused" sur un hub parfaitement vivant. La preuve, c'est
        # l'association SCTP ETABLIE, vue depuis le noeud lui-meme.
        assoc="$(inside "$i" "ss -an --sctp" | grep -E "ESTAB.*[[:space:]]${remote//./\\.}:2908([[:space:]]|\$)"
        ↪  || true)"
        if [ -n "$assoc" ]; then
            ok "op${i} : association SCTP etablie vers le hub ${remote}:2908"
        elif inside "$i" "ss -an --sctp" | grep -q .; then
            fail "op${i} : aucune association SCTP vers ${remote}:2908 - l'ASP est DOWN"
            echo -e "     ${CYAN}Sur le noeud inter-STP : ./launch/start-interstp.sh --status${NC}"
        else
            skip "op${i} : ss --sctp indisponible - lien vers ${remote}:2908 non verifiable"
        fi
    else
        case "$remote" in
            172.2[0-9].*|127.*)
                ok "op${i} : ASP inter-STP → ${remote} (docker local) - le SS7 reste dans le noeud"
                [ "${#REMOTES[@]}" -ge 1 ] && info "pas de hub commun : voix et SMS traversent, le SS7 non" ;;
            "") skip "op${i} : pas d'ASP inter-STP declare" ;;
           *)  warn "op${i} : ASP inter-STP → ${remote} - M3UA sur un lien non maitrise ? (pas de chiffrement SCTP
            ↪  ici)" ;;
        esac
    fi
done

# 5b - le point code porte-t-il le numero de noeud ?
# Le plan local (1.<op>.<role>) se recalcule a l'identique partout : trois
# noeuds attaches au meme hub y presenteraient trois fois 1.1.2. Le plan WAN
# (1.<noeud><op>.<role>) leve l'ambiguite. On verifie lequel est en place.
if [ "${#REMOTES[@]}" -ge 1 ]; then
    pc_local="$(inside 1 'cat /etc/osmocom/osmo-stp.cfg' | awk '/^ point-code/{print $2; exit}')"
    expected="1.${WAN_NODE_ID}1.2"
    if [ "$pc_local" = "$expected" ]; then
        ok "point code ${pc_local} : le numero de noeud (${WAN_NODE_ID}) y est encode - pas de collision au hub"
    elif [ -n "$HUB_IP" ]; then
        fail "point code ${pc_local} attendu ${expected} : ce noeud entrerait en collision avec ses pairs sur le
        ↪  hub"
        echo -e "     ${CYAN}Reconstruire l'image : ./build-iso.sh --role operator --node ${WAN_NODE_ID}${NC}"
    else
        warn "point code ${pc_local} : plan local, identique sur tous les noeuds"
        echo -e "     ${CYAN}Sans effet tant qu'il n'y a pas de hub commun. Pour du SS7 inter-noeud :${NC}"
        echo -e "     ${CYAN}./build-iso.sh --role interstp   puis   --role operator --node N${NC}"
    fi
fi

# 5c - un port M3UA expose publiquement ?
LOCAL_IP="$(wan_local_ip)"
for port in 2905 2908; do
    if command -v ss >/dev/null 2>&1; then
        # M3UA tourne sur SCTP dans ce depot (osmo-stp.cfg : "listen m3ua ..."),
        # pas sur TCP : "ss -lntp" seul ne verrait RIEN et conclurait a tort
        # "aucun ecouteur". On regarde les deux transports.
        listen="$( { ss -lntp 2>/dev/null; ss -ln --sctp 2>/dev/null; } | grep ":${port} ")"
        if [ -z "$listen" ]; then
            ok "M3UA ${port} : aucun ecouteur sur l'hote"
```

```
        elif echo "$listen" | grep -qE '0\.0\.0\.0:'"${port}"'|\[::\]:'"${port}"; then
            warn "M3UA ${port} ecoute sur 0.0.0.0 - expose si le firewall ne le ferme pas"
            echo -e "     ${CYAN}iptables -A INPUT -p sctp --dport ${port} -j DROP${NC} (ou -p tcp selon le
             ↪  transport)"
        else
            ok "M3UA ${port} : ecoute restreinte ($(echo "$listen" | awk '{print $4}' | tr '\n' ' '))"
        fi
    else skip "ss absent : ecouteurs M3UA non verifies"; fi
done

# 5d - SS7 interne toujours sain ? (delegue au check existant)
if [ -x "$SCRIPT_DIR/checks/ss7_check.sh" ] && [ "$MODE" = docker ]; then
    echo ""
    echo -e "  ${CYAN}SS7 interne : checks/ss7_check.sh --quick${NC}"
else
    skip "SS7 interne : checks/ss7_check.sh (docker uniquement)"
fi

# ── 6. iptables ──────────────────────────────────────────────────────────
banner "6. iptables - chaine WAN"
if iptables -t nat -L OSMO_WAN_MESH -n >/dev/null 2>&1; then
    n=$(iptables -t nat -S OSMO_WAN_MESH 2>/dev/null | grep -c '^-A')
    ok "chaine OSMO_WAN_MESH presente (${n} regle(s))"
    iptables -t nat -S PREROUTING 2>/dev/null | grep -q 'OSMO_WAN_MESH' \
        && ok "chaine branchee sur PREROUTING" \
        || fail "chaine NON branchee sur PREROUTING - rien n'entre"
else
    if iptables -t nat -L OSMO_WAN_INTEROP -n >/dev/null 2>&1; then
        warn "chaine legacy OSMO_WAN_INTEROP active (2 serveurs, prefixe 66)"
    else
        fail "aucune chaine WAN - relancez network/setup-wan-mesh.sh sans --config-only"
    fi
fi

# ── Bilan ────────────────────────────────────────────────────────────────
echo ""
printf '─%.0s' $(seq 1 64); echo ""
echo -e "  ${GREEN}${PASS} ok${NC}   ${RED}${FAIL} echec${NC}   ${YELLOW}${WARN} avertissement${NC}   ${SKIP}
 ↪  ignore"
echo ""
[ "$FAIL" -eq 0 ] || exit 1
```

3.45 /opt/GSM/osmo-operator/configs/asound.conf

1649 octets, 48 lignes → 48 lignes

```
# /etc/asound.conf — Osmocom GSM audio via PulseAudio
#
# gsm_out : osmo-gapk playback → sink PulseAudio "gsm_audio"
# gsm_in  : osmo-gapk capture  → moniteur de "gsm_mic" (silence)
#
# [2026-08-08] gsm_in n'avait PAS de `device` : il retombait sur la source par
# défaut, qui est gsm_audio.monitor — la sortie que les mobiles venaient
# d'écrire. Chaque mobile s'entendait donc lui-même et entendait l'autre :
# boucle fermée par construction (mesuré : RMS ×11 en 10 s sur un appel de 13 s,
# crête 17728). En pointant la capture sur un sink que personne n'alimente, la
# boucle s'ouvre sans changer quoi que ce soit côté mobiles.

pcm.gsm_out {
    # [2026-08-10] NE PAS remettre period_size / buffer_size dans un bloc
    # `slave` : ALSA les REJETTE — « snd_pcm_slave_conf: Unknown field
    # period_size » — et le peripherique ne s'ouvre plus (plus aucun sink-input
    # sur gsm_audio). Essai fait et annule le 10/08. La geometrie du tampon se
    # negocie cote APPLICATION (gapk, snd_pcm_set_params dans pq_alsa), pas ici.
```

```
    #
    # CONSTAT QUI RESTE A EXPLIQUER : sur un appel muet, le flux de lecture du
    # mobile tourne a vide -- 6 300 ecritures/s sur le tube de reveil du greffon
    # ALSA-PulseAudio (fd 17), `Buffer Latency = 0 us`, crete 0, et AUCUNE
    # erreur (ni « execution aborted » de la file gapk, ni message ALSA).
    # L'autre mobile, au repos, fait 6 ecritures/s.
    type pulse
    device gsm_audio
}

ctl.gsm_out {
    type pulse
}

pcm.gsm_in {
    type pulse
    device gsm_mic.monitor
}

ctl.gsm_in {
    type pulse
}

pcm.!default {
    type pulse
}

ctl.!default {
    type pulse
}
```

3.46 /opt/GSM/osmo-operator/configs/extensions.conf

32736 octets, 550 lignes → 550 lignes

```
; ============================================================================
; extensions.conf — Asterisk dialplan — Opérateur 1
;
; ┌──────────────────────────────────────────────────────────────────────────┐
; │  PLAN DE NUMÉROTATION                                                    │
; │  100        Linphone A (softphone local)                                 │
; │  200        Linphone B (softphone local)                                 │
; │  600        Echo test                                                    │
; │  700        Annonce de bienvenue                                         │
; │  NXXXX      Abonnés GSM opérateur N (premier chiffre = identifiant op)   │
; ├──────────────────────────────────────────────────────────────────────────┤
; │  CONTEXTES                                                               │
; │  [internal]    Appels depuis softphones locaux                           │
; │  [gsm_in]      Appels entrants depuis OsmoMSC (GSM → Asterisk)           │
; │  [interop_in]  Appels entrants depuis un opérateur distant (SIP trunk)   │
; │  [gsm_out]     Sous-contexte : sortie vers MSC local                     │
; │  [interop_out] Routage vers l'opérateur distant approprié                │
; │                GÉNÉRÉ par start-docker.sh selon le nombre d'opérateurs   │
; └──────────────────────────────────────────────────────────────────────────┘
;
; SCÉNARIOS COUVERTS
; ──────────────────
; 1. MS → MS  (même opérateur)    : MS → MSC → MNCC → Asterisk → MNCC → MSC → MS
; 2. MS → Linphone                 : MS → MSC → Asterisk SIP → Linphone
; 3. Linphone → MS                 : Linphone SIP → Asterisk → MNCC → MSC → MS
; 4. Linphone A → Linphone B       : SIP local
; 5. MS(OpX) → MS(OpY) inter-op   : MNCC → AstX → SIP trunk → AstY → MNCC → MS
; 6. MS → echo / annonce           : test audio
;
```

```
; NOTE : [interop_out] est ajouté en fin de fichier par start-docker.sh.
;        Il contient une extension par opérateur distant, routée vers
;        le trunk interop_trunk_opX correspondant.
; ============================================================================

[general]
static=yes
writeprotect=no
clearglobalvars=no

[globals]
MSC_LOCAL=127.0.0.1
; [2026-08-31] Était codé en dur à 1 : sur op2 et op3 les enregistrements
; sortaient en « ..._op1_... » (voir sub-record plus bas) et les trois
; opérateurs se mélangeaient sous le même label.
OP_PREFIX=__OPERATOR_ID__
; ── Monitoring : enregistrement horodaté de TOUS les appels (yes/no) ──
; [2026-08-12] PASSÉ À `no` — latence sur le MONTANT vers Zadarma.
; `yes` ne coûte pas que l'écriture d'un .wav, il enclenche DEUX choses dans
; [sub-record], et les deux frappent le montant seul :
;   1. `Set(JITTERBUFFER(adaptive)=default)` (l. ~60). Le Gosub tourne sur le
;      canal APPELANT (le MS) et AVANT le Dial : le tampon est posé sur le côté
;      LECTURE de ce canal, donc sur l'audio MS→Asterisk. La patte Zadarma, née
;      du Dial, n'en a aucune. Asymétrie par construction. Face à la gigue de la
;      chaîne Calypso→gapk→RTP, un JB adaptatif monte à son plafond et y reste.
;   2. L'audiohook MixMonitor sort le pont du RTP natif → transcodage forcé
;      gsm↔alaw avec Zadarma (cf. pjsip.conf : allow=alaw,ulaw,gsm).
; Remettre `yes` rend les deux d'un coup — et avec eux les .wav -ul/-dl qui
; servent à attribuer une dégradation à un sens. Si tu as besoin des
; enregistrements SANS la latence, garde MixMonitor et retire la ligne
; JITTERBUFFER, ou pose-la sur la patte du tronc (là où est vraiment la gigue),
; jamais sur celle du MS.
CALL_RECORDING=yes
RECORDINGS_DIR=/var/spool/asterisk/monitor

; ── Entrant Zadarma : à qui sonne un appel venu de l'extérieur ──────────────
; [2026-09-09] LA VARIABLE N'EXISTAIT NULLE PART. [zadarma_in] faisait déjà
; `Dial(PJSIP/${DEST_ZADARMA_IN}@gsm_msc,30,rT)` mais aucun [globals] ne la
; posait : Asterisk composait littéralement `PJSIP/@gsm_msc`, le Dial échouait
; sur-le-champ et TOUT appel entrant tombait. Le trunk pouvait bien être
; enregistré, l'entrant ne pouvait pas aboutir.
; Valeur = un MSISDN du HLR (voir run_modules/21-abonnes-hlr.sh) :
;   100101  le téléphone du banc (osmocom-bb / VM postmarketOS)
;   100102  le second mobile
; Pour sonner un softphone à la place, viser [internal] plutôt que gsm_msc
; (cf. la ligne Dial de [zadarma_in], en fin de fichier).
DEST_ZADARMA_IN=100101

; ============================================================================
; [sub-record] — Démarre l'enregistrement (MixMonitor) horodaté de l'appel.
;   Appelé en tête de chaque appel : Gosub(sub-record,s,1(<numéro_appelé>))
;   Fichier : ${RECORDINGS_DIR}/<date>_<heure>_op<N>_<appelant>_to_<appelé>_<uid>.wav
;   Option 'b' : n'enregistre que l'audio pendant que l'appel est ponté.
;   Désactivable globalement via CALL_RECORDING=no.
;   Nécessite app_mixmonitor.so (cf modules.conf).
; ============================================================================
[sub-record]
exten => s,1,GotoIf($["${CALL_RECORDING}" != "yes"]?skip)
 ; L'enregistrement (audiohook) sort le pont du RTP natif → transcodage forcé.
 ; Un jitterbuffer adaptatif lisse le flux pour éviter les pertes de trames
 ; (translate slin→gsm) qui coupaient le son. Nécessite func_jitterbuffer.so.
 ; [2026-08-13] RETIRÉE — JB posé sur la lecture de la patte MS = le MONTANT seul (cf.
 ↪  /etc/asterisk/extensions.conf)
 ; same => n,Set(JITTERBUFFER(adaptive)=default)
 same => n,Set(REC_TS=${STRFTIME(${EPOCH},,%Y%m%d_%H%M%S)})
 same => n,Set(REC_NAME=${FILTER(A-Za-z0-9_-,${REC_TS}_op${OP_PREFIX}_${CALLERID(num)}_to_${ARG1}_${UNIQUEID})})
```

```
 ; [2026-08-11] r()/t() : le .wav mixe les DEUX sens et ne permet donc pas
 ; d'attribuer une degradation au montant ou au descendant. Sur le canal
 ; appelant (le MS) :  r() = flux LU = MONTANT ;  t() = flux ECRIT = DESCENDANT.
 ; ⚠ PATCHER ICI, PAS DANS /etc/asterisk : /etc/asterisk/extensions.conf est
 ; une COPIE regeneree au demarrage depuis ce fichier. La modification faite
 ; le 10/08 dans la copie a disparu a la premiere relance (constate le 11/08).
 same => n,MixMonitor(${RECORDINGS_DIR}/${REC_NAME}.wav,br(${RECORDINGS_DIR}/${REC_NAME}-
 ↪  ul.wav)t(${RECORDINGS_DIR}/${REC_NAME}-dl.wav))
 same => n,NoOp(=== REC → ${RECORDINGS_DIR}/${REC_NAME}.wav ===)
 same => n(skip),Return()

; ============================================================================
; [sub-annuaire] — le nom derriere le numero
; ============================================================================
; Le banc ne rend que des MSISDN a cinq chiffres, et « 20001 » ne dit rien a
; personne. L'annuaire du banc (data/noms.txt, tire au hasard par start-docker.sh, ecrit
; dans /etc/osmocom/annuaire.csv) met un etat civil en face de chaque abonne.
; On l'amene ici pour qu'il arrive JUSQU'AU COMBINE : le mobile appele affiche
; alors « Aline Boulanger » et non « 20001 ».
;
; POURQUOI CE SOUS-PROGRAMME EST APPELE APRES Set(CALLERID(all)=<...>) :
; cette ligne-la RECONSTRUIT le CallerID sans nom (« <20001> »). L'appeler avant
; reviendrait a poser un nom pour l'effacer aussitot.
;
; Les fiches elles-memes sont dans annuaire.conf, ecrit par start-docker.sh au
; lancement (une extension par MSISDN). Le motif _X. ci-dessous est le repli :
; Asterisk fait primer une extension exacte sur un motif, donc une fiche connue
; gagne, et un numero inconnu ressort d'ici sans rien casser - c'est aussi ce
; qui se passe quand annuaire.conf n'existe pas du tout.
[sub-annuaire]
exten => _X.,1,NoOp(=== ANNUAIRE : ${EXTEN} inconnu - CallerID inchange ===)
 same => n,Return()
exten => 0,1,NoOp(=== ANNUAIRE : appelant anonyme ===)
 same => n,Return()

#include "annuaire.conf"

; ============================================================================
; [internal] — Appels DEPUIS les softphones locaux
; ============================================================================
[internal]

; ── 500 : lecture audio lithium ──────────────────────────────────────
;   Joue /root/lithium.wav du conteneur (WAV 8 kHz mono recommandé).
;   Playback(/root/lithium) sans extension → Asterisk résout /root/lithium.wav.
;   Fichier à fournir dans chaque conteneur (osmo-operator-1 et distant).
exten => 500,1,NoOp(=== LITHIUM - CallerID: ${CALLERID(all)} ===)
 same => n,Answer()
 same => n,Wait(1)
 same => n,Playback(/root/lithium)
 same => n,Hangup()

; ── Tests audio ──────────────────────────────────────────────────────
exten => 600,1,NoOp(=== ECHO TEST - CallerID: ${CALLERID(all)} ===)
 same => n,Answer()
 ; [2026-09-09] PAS D ANNONCE AVANT L ECHO POUR LES APPELS DU BANC (ECHO_NO_PROMPT=1,
 ; pose par osmo-phonesim-banc.py sur son Originate). Le banc origine par
 ; Local/600@internal, et sa patte ;1 enregistre l appel (sub-record). Avec
 ; MixMonitor dans le pont, Asterisk lachait, a l instant precis ou cette
 ; patte ;2 passait de Playback(demo-echotest) a Echo, une RAFALE de 7 000 a
 ; 28 000 paquets RTP de silence en 60 ms (mesure sur lo le 09/09 : 28 414
 ; paquets, 15 payloads distincts, 568 s d audio) : le jitter-buffer du BTS
 ; se desynchronisait et le son etait mauvais pendant une partie de l appel.
 ; Reproduit a volonte sur le 2e mobile : sans enregistrement, pas de rafale ;
 ; enregistrement + Echo SANS annonce prealable, pas de rafale non plus.
 ; L annonce reste pour les autres appelants (Linphone, mobile via gsm_in).
```

```
 ; Pour le banc, elle est rejouee SUR LA PATTE DU MOBILE : le banc pose aussi
 ; DIAL_OPTS=A(demo-echotest), et le Dial de la jambe ;1 (ci-dessous, ${DIAL_OPTS})
 ; la joue au decroche, avant le pont. Teste : annonce entendue, aucune rafale.
 same => n,GotoIf($["${ECHO_NO_PROMPT}" = "1"]?echo)
 same => n,Playback(demo-echotest)
 same => n(echo),Echo()
 same => n,Playback(demo-echodone)
 same => n,Hangup()

exten => 700,1,NoOp(=== ANNONCE - CallerID: ${CALLERID(all)} ===)
 same => n,Answer()
 same => n,Playback(hello-world)
 same => n,Hangup()

; ── Softphone → Softphone ──────────────────────────────────────────
exten => 100,1,NoOp(=== SIP: → Linphone A - CallerID: ${CALLERID(all)} ===)
 same => n,Gosub(sub-record,s,1(${EXTEN}))
 same => n,Dial(PJSIP/linphone_A,,rT)
 same => n,Congestion()
 same => n,Hangup()

exten => 200,1,NoOp(=== SIP: → Linphone B - CallerID: ${CALLERID(all)} ===)
 same => n,Gosub(sub-record,s,1(${EXTEN}))
 same => n,Dial(PJSIP/linphone_B,,rT)
 same => n,Congestion()
 same => n,Hangup()

; ── Softphone → GSM local ──────────────────────────────────────────
; Numéros de l'opérateur local : <numéro d'opérateur> suivi de 4 chiffres.
; Le plan est celui du HLR (start-docker.sh, run_modules/21-abonnes-hlr.sh et
; scripts/sms-routing-setup.sh) : MSISDN = opérateur * 10000 + rang du mobile.
; Le motif était figé sur « 1 » : sur l'opérateur 2, dont les abonnés sont
; 20001 et 20002, aucun appel local ne matchait plus. Il tombait dans le
; fourre-tout _X. « routage inter-op » et partait chercher hors du nœud un
; abonné qui était sur place.
exten => ___MSISDN_PFX____OPERATOR_ID__XX,1,NoOp(=== SIP -> GSM local (Op__OPERATOR_ID__): ${EXTEN} - CallerID:
 ↪  ${CALLERID(all)} ===)
 same => n,Set(CALLERID(all)=<${CALLERID(num)}>)
 ; Le nom de l'appelant, depuis l'annuaire du banc. ANNU_NUM plutot que
 ; CALLERID(num) directement : un appelant anonyme donnerait une extension vide
 ; et un Gosub en echec bruyant a chaque appel.
 same => n,Set(ANNU_NUM=${IF($["${CALLERID(num)}" != ""]?${CALLERID(num)}:0)})
 same => n,Gosub(sub-annuaire,${ANNU_NUM},1)
 same => n,Gosub(sub-record,s,1(${EXTEN}))
 same => n,Dial(PJSIP/${EXTEN}@gsm_msc,,rT${DIAL_OPTS})
 same => n,Congestion()
 same => n,Hangup()

; ── Softphone → GSM inter-opérateur ────────────────────────────────
; Composer 9 + numéro complet (ex: 920001 pour joindre 20001 sur Op2)
exten => _9.,1,NoOp(=== SIP -> INTER-OP: ${EXTEN:1} - CallerID: ${CALLERID(all)} ===)
 same => n,Gosub(sub-record,s,1(${EXTEN:1}))
 same => n,Goto(interop_out,${EXTEN:1},1)

; ── Catch-all : essayer le MSC local ───────────────────────────────
;
; ── Sortant vers Zadarma : 00 + numéro international ───────────────
; Plus spécifique que le catch-all _X. ci-dessous, donc prioritaire.
; Le 00 est retiré et remplacé par le format E.164 attendu par Zadarma.
exten => _00.,1,NoOp(=== SIP -> ZADARMA: +${EXTEN:2} - CallerID: ${CALLERID(all)} ===)
 same => n,Gosub(sub-record,s,1(zadarma_${EXTEN:2}))
; [2026-08-09] LE « + » EST OBLIGATOIRE. Mesure A/B sur le tronc, même session :
;   INVITE sip:33656678074@sip.zadarma.com   -> SIP/2.0 404 Not Found
;   INVITE sip:+33656678074@sip.zadarma.com  -> 100 trying, 183 Session Progress
;   INVITE sip:0656678074@sip.zadarma.com    -> 180 Ringing, "zadarma answered"
; ${EXTEN:2} retirait le 00 sans rien remettre : Zadarma refuse l'E.164 nu. Tous
```

```
; les appels sortants via Asterisk échouaient donc en 404 -> Congestion() -> 503
; vers le MSC, MSC comme Linphone. Et côté MS ça se voyait comme un raccrochage
; qui « ne marche pas » : sans appel côté réseau, ni CALL PROCEEDING ni RELEASE
; COMPLETE, donc T308 jusqu'à son terme (30 s).
 same => n,Dial(PJSIP/+${EXTEN:2}@zadarma,60,rT)
 same => n,NoOp(=== ZADARMA OUT: échec ${DIALSTATUS} ===)
 same => n,Congestion()
 same => n,Hangup()


; ── Sortant vers Zadarma : numéro NATIONAL français, composé tel quel ────────
; [2026-09-09] AJOUTÉ. Jusqu'ici il fallait composer 0033612345678 pour joindre
; un numéro français. La forme naturelle — 0612345678 — n'était attrapée par
; aucun motif Zadarma : elle tombait dans le catch-all _X. plus bas, partait
; vers le MSC local, et celui-ci répondait « unknown subscriber number ».
; On compose désormais comme sur n'importe quel téléphone : 01 02 03 04 05 06
; 07 08 09 + les huit chiffres.
;
; MOTIF : 0, puis UN chiffre 1-9, puis 8 chiffres = les dix chiffres du plan
; français. Le deuxième chiffre exclut explicitement le 0 : aucun recouvrement
; possible avec _00. (international) ci-dessus — les deux motifs ne peuvent pas
; matcher la même chaîne, leur ordre relatif est donc sans effet. Face au
; catch-all _X., c'est le plus spécifique (longueur fixe) qui gagne.
;
; CONVERSION : 0612345678 -> +33612345678. Le 0 de tête est le préfixe
; NATIONAL : il se REMPLACE par l'indicatif pays, il ne s'ajoute pas devant.
; Et le « + » est obligatoire — cf. la mesure A/B du 09/08 rappelée dans le
; bloc _00. ci-dessus : sans lui, Zadarma rend 404 Not Found.
exten => _0[1-9]XXXXXXXX,1,NoOp(=== SIP -> ZADARMA (national): +33${EXTEN:1} - CallerID: ${CALLERID(all)} ===)
 same => n,Gosub(sub-record,s,1(zadarma_33${EXTEN:1}))
 same => n,Dial(PJSIP/+33${EXTEN:1}@zadarma,60,rT)
 same => n,NoOp(=== ZADARMA OUT (national): échec ${DIALSTATUS} ===)
 same => n,Congestion()
 same => n,Hangup()

exten => _X.,1,NoOp(=== SIP -> GSM fallback: ${EXTEN} - CallerID: ${CALLERID(all)} ===)
 same => n,Gosub(sub-record,s,1(${EXTEN}))
 same => n,Dial(PJSIP/${EXTEN}@gsm_msc,,rT${DIAL_OPTS})
 same => n,Congestion()
 same => n,Hangup()

; ============================================================================
; [gsm_in] — Appels entrants DEPUIS OsmoMSC (GSM → Asterisk)
;
; ARCHITECTURE DES PATTERNS (du plus spécifique au moins spécifique) :
;   1. Numéros fixes (100, 200, 600)       → services locaux
;   2. _NXXXX  (5 chiffres, préfixe op)    → MS local même opérateur
;   3. _X.     (catch-all)                  → routage inter-op ou rejet
;
; Asterisk matche le pattern le PLUS SPÉCIFIQUE en premier.
; _NXXXX (5 chars fixes) bat _X. (wildcard) — pas besoin de priorité.
;
; NOTE : les patterns _[0-9][0-9]XXXX et _[0-9]XXXXX sont SUPPRIMÉS
; car ils sont des alias l'un de l'autre (6 chiffres quelconques)
; et causent des conflits de priorité avec Asterisk.
; ============================================================================
[gsm_in]

; ── 500 : lecture audio lithium (/root/lithium.wav du conteneur) ──
exten => 500,1,NoOp(=== GSM -> LITHIUM - CallerID: ${CALLERID(all)} ===)
 same => n,Answer()
 same => n,Wait(1)
 same => n,Playback(/root/lithium)
 same => n,Hangup()

; ── Tests audio ──────────────────────────────────────────────────────────────
```

```
exten => 600,1,NoOp(=== GSM -> ECHO TEST - CallerID: ${CALLERID(all)} ===)
 same => n,Answer()
 same => n,Playback(demo-echotest)
 same => n,Echo()
 same => n,Playback(demo-echodone)
 same => n,Hangup()

; ── MS → Linphone ──────────────────────────────────────────────────
exten => 100,1,NoOp(=== GSM -> Linphone A - CallerID: ${CALLERID(all)} ===)
 same => n,Gosub(sub-record,s,1(${EXTEN}))
 same => n,Dial(PJSIP/linphone_A,,rT)
 same => n,Congestion()
 same => n,Hangup()

exten => 200,1,NoOp(=== GSM -> Linphone B - CallerID: ${CALLERID(all)} ===)
 same => n,Gosub(sub-record,s,1(${EXTEN}))
 same => n,Dial(PJSIP/linphone_B,,rT)
 same => n,Congestion()
 same => n,Hangup()

; ── MS → MS local (même opérateur) ─────────────────────────────────
; Motif : le numéro de l'opérateur suivi de 4 chiffres (10001 pour l'Op1,
; 20001 pour l'Op2...). Voir plus haut : ce motif était figé sur « 1 ».
exten => ___MSISDN_PFX____OPERATOR_ID__XX,1,NoOp(=== GSM -> GSM local (Op__OPERATOR_ID__): ${EXTEN} - CallerID:
 ↪  ${CALLERID(all)} ===)
 same => n,Set(CALLERID(all)=<${CALLERID(num)}>)
 ; Le nom de l'appelant, depuis l'annuaire du banc. ANNU_NUM plutot que
 ; CALLERID(num) directement : un appelant anonyme donnerait une extension vide
 ; et un Gosub en echec bruyant a chaque appel.
 same => n,Set(ANNU_NUM=${IF($["${CALLERID(num)}" != ""]?${CALLERID(num)}:0)})
 same => n,Gosub(sub-annuaire,${ANNU_NUM},1)
 same => n,Gosub(sub-record,s,1(${EXTEN}))
 same => n,Dial(PJSIP/${EXTEN}@gsm_msc,,rT)
 same => n,Congestion()
 same => n,Hangup()

; ── Routage inter-opérateur (catch-all) ────────────────────────────
; Tout ce qui n'est pas un numéro local ou un service fixe arrive ici.
; On tente le routage inter-op via [interop_out].
; Si le numéro commence par notre propre préfixe → invalide (évite boucle).
;
; ── Sortant vers Zadarma : 00 + numéro international ──────────────
; Plus spécifique que le catch-all _X. ci-dessous, donc prioritaire.
; Le 00 est retiré et remplacé par le format E.164 attendu par Zadarma.
exten => _00.,1,NoOp(=== GSM -> ZADARMA: +${EXTEN:2} - CallerID: ${CALLERID(all)} ===)
 same => n,Gosub(sub-record,s,1(zadarma_${EXTEN:2}))
; [2026-08-09] LE « + » EST OBLIGATOIRE. Mesure A/B sur le tronc, même session :
;   INVITE sip:33656678074@sip.zadarma.com   -> SIP/2.0 404 Not Found
;   INVITE sip:+33656678074@sip.zadarma.com  -> 100 trying, 183 Session Progress
;   INVITE sip:0656678074@sip.zadarma.com    -> 180 Ringing, "zadarma answered"
; ${EXTEN:2} retirait le 00 sans rien remettre : Zadarma refuse l'E.164 nu. Tous
; les appels sortants via Asterisk échouaient donc en 404 -> Congestion() -> 503
; vers le MSC, MSC comme Linphone. Et côté MS ça se voyait comme un raccrochage
; qui « ne marche pas » : sans appel côté réseau, ni CALL PROCEEDING ni RELEASE
; COMPLETE, donc T308 jusqu'à son terme (30 s).
 same => n,Dial(PJSIP/+${EXTEN:2}@zadarma,60,rT)
 same => n,NoOp(=== ZADARMA OUT: échec ${DIALSTATUS} ===)
 same => n,Congestion()
 same => n,Hangup()

; ── E.164 avec « + » : ce que le MSC présente RÉELLEMENT ──────────
; [2026-08-09] MESURE, pas supposition : composer 0033656678074 depuis le MS
; faisait arriver ici « +33656678074 ». osmo-msc normalise le 00 international
; en E.164 et pose le « + ». Or AUCUN motif de ce contexte ne commençait par
; « + » : ni _00. (le préfixe a disparu), ni le catch-all _X. (X = un CHIFFRE,
; pas « + »). Résultat mesuré, journal Asterisk :
```

```
;   res_pjsip_session.c: Call to extension '+33656678074' rejected because
;   extension not found in context 'gsm_in'
; Et côté MS ça se voit comme un raccrochage qui « ne marche pas » : l'appel
; n'existant pas côté réseau, il n'y a ni CALL PROCEEDING ni RELEASE COMPLETE,
; donc le mobile attend l'expiration de T308 (30 s) avant de libérer seul.
; Placé APRÈS _00. et AVANT _X. — un motif littéral « + » n'entre en conflit
; avec aucun des deux, ils ne peuvent pas matcher la même chaîne.
; ⚠ Ceci ouvre la route ; il reste que le tronc n'est pas configuré
; (ZADARMA_LOGIN_A_REMPLIR -> 401 à chaque enregistrement) : sans identifiants
; valides dans pjsip.conf, l'appel échouera plus loin, en Congestion().

; ── Sortant vers Zadarma : numéro NATIONAL français, composé tel quel ────────
; [2026-09-09] AJOUTÉ. Jusqu'ici il fallait composer 0033612345678 pour joindre
; un numéro français. La forme naturelle — 0612345678 — n'était attrapée par
; aucun motif Zadarma : elle tombait dans le catch-all _X. plus bas, partait
; vers le MSC local, et celui-ci répondait « unknown subscriber number ».
; On compose désormais comme sur n'importe quel téléphone : 01 02 03 04 05 06
; 07 08 09 + les huit chiffres.
;
; MOTIF : 0, puis UN chiffre 1-9, puis 8 chiffres = les dix chiffres du plan
; français. Le deuxième chiffre exclut explicitement le 0 : aucun recouvrement
; possible avec _00. (international) ci-dessus — les deux motifs ne peuvent pas
; matcher la même chaîne, leur ordre relatif est donc sans effet. Face au
; catch-all _X., c'est le plus spécifique (longueur fixe) qui gagne.
;
; CONVERSION : 0612345678 -> +33612345678. Le 0 de tête est le préfixe
; NATIONAL : il se REMPLACE par l'indicatif pays, il ne s'ajoute pas devant.
; Et le « + » est obligatoire — cf. la mesure A/B du 09/08 rappelée dans le
; bloc _00. ci-dessus : sans lui, Zadarma rend 404 Not Found.
exten => _0[1-9]XXXXXXXX,1,NoOp(=== GSM -> ZADARMA (national): +33${EXTEN:1} - CallerID: ${CALLERID(all)} ===)
 same => n,Gosub(sub-record,s,1(zadarma_33${EXTEN:1}))
 same => n,Dial(PJSIP/+33${EXTEN:1}@zadarma,60,rT)
 same => n,NoOp(=== ZADARMA OUT (national): échec ${DIALSTATUS} ===)
 same => n,Congestion()
 same => n,Hangup()

exten => _+X.,1,NoOp(=== GSM -> ZADARMA (E.164): ${EXTEN} - CallerID: ${CALLERID(all)} ===)
 same => n,Gosub(sub-record,s,1(zadarma_${EXTEN:1}))
; Le « + » est déjà là : on le GARDE (cf. la mesure A/B ci-dessus). L'ancienne
; forme ${EXTEN:1} le retirait et retombait dans le cas qui rend 404.
 same => n,Dial(PJSIP/${EXTEN}@zadarma,60,rT)
 same => n,NoOp(=== ZADARMA OUT (E.164): échec ${DIALSTATUS} ===)
 same => n,Congestion()
 same => n,Hangup()

; [2026-08-31] ICI MOURAIT TOUT APPEL GSM INTER-OPÉRATEUR.
; La ligne était :
;     same => n,GotoIf($["${EXTEN:0:1}" = "1"]?local_fallback)
; « premier chiffre = 1 » voulait dire « numéro local » à l'époque du plan à
; CINQ chiffres, où le premier chiffre ÉTAIT l'opérateur. Depuis le passage à
; MSISDN = <nœud>00<opérateur><rang>, TOUS les numéros du banc commencent par
; le numéro de nœud — 100201, 100301… — - donc ce test attrapait tout et
; envoyait chaque appel sortant en Congestion() avant même d'atteindre
; [interop_out]. Symptôme : le MSC sortait bien l'appel
;     rx MNCC_SETUP_REQ for unknown subscriber number '100201'
; et osmo-sip-connector recevait
;     INVITE got status(503), releasing leg
; Les appels depuis un softphone passaient, eux : [internal] n'a pas ce test.
;
; LE TEST DEVIENT UN MOTIF. On ne bricole plus le numéro à coups de sous-chaîne :
; « local » se dit dans le plan de numérotation, et Asterisk sait déjà comparer
; des motifs. _<pfx><op>. est PLUS spécifique que _X. et MOINS que le
; _<pfx><op>XX déclaré plus haut : un numéro local exact part au MSC, un numéro
; local mal formé tombe ici, tout le reste sort en inter-opérateur.
exten => ___MSISDN_PFX____OPERATOR_ID__.,1,NoOp(=== GSM -> numéro local inexistant: ${EXTEN} ===)
 same => n,Congestion()
```

```
 same => n,Hangup()

exten => _X.,1,NoOp(=== GSM -> routage inter-op: ${EXTEN} - CallerID: ${CALLERID(all)} ===)
 same => n,Gosub(sub-record,s,1(${EXTEN}))
 same => n,Goto(interop_out,${EXTEN},1)

; ============================================================================
; [interop_in] — Appels entrants DEPUIS un opérateur distant (SIP trunk)
; ============================================================================
[interop_in]

; MS distant → MS local
exten => ___MSISDN_PFX____OPERATOR_ID__XX,1,NoOp(=== INTER-OP -> GSM local: ${EXTEN} - CallerID distant:
 ↪  ${CALLERID(all)} ===)
 same => n,Set(CALLERID(all)=<${CALLERID(num)}>)
 ; Le nom de l'appelant, depuis l'annuaire du banc. ANNU_NUM plutot que
 ; CALLERID(num) directement : un appelant anonyme donnerait une extension vide
 ; et un Gosub en echec bruyant a chaque appel.
 same => n,Set(ANNU_NUM=${IF($["${CALLERID(num)}" != ""]?${CALLERID(num)}:0)})
 same => n,Gosub(sub-annuaire,${ANNU_NUM},1)
 same => n,Gosub(sub-record,s,1(${EXTEN}))
 same => n,Dial(PJSIP/${EXTEN}@gsm_msc,,rT)
 same => n,Congestion()
 same => n,Hangup()

; Le motif a CINQ chiffres est supprime : depuis le passage au plan 600xxx,
; tous les MSISDN locaux font six chiffres. Deux motifs pour une seule longueur
; donneraient une extension en double dans le meme contexte, et Asterisk
; n'en garderait qu'une - au hasard de l'ordre de lecture.

; Opérateur distant → Linphone local
exten => 100,1,NoOp(=== INTER-OP -> Linphone A - CallerID distant: ${CALLERID(all)} ===)
 same => n,Gosub(sub-record,s,1(${EXTEN}))
 same => n,Dial(PJSIP/linphone_A,,rT)
 same => n,Hangup()

exten => 200,1,NoOp(=== INTER-OP -> Linphone B - CallerID distant: ${CALLERID(all)} ===)
 same => n,Gosub(sub-record,s,1(${EXTEN}))
 same => n,Dial(PJSIP/linphone_B,,rT)
 same => n,Hangup()

; Refus propre
exten => _X.,1,NoOp(=== INTER-OP -> numéro inconnu: ${EXTEN} - CallerID distant: ${CALLERID(all)} ===)
 same => n,Congestion()
 same => n,Hangup()

; ============================================================================
; [interop_out] — Routage sortant vers les opérateurs distants
;
; Ce contexte est GÉNÉRÉ et AJOUTÉ en fin de fichier par start-docker.sh.
; Il contient une extension _NXXXX par opérateur distant (N ≠ 1)
; qui route vers interop_trunk_opN via PJSIP.
;
; Exemple pour Op1 avec 3 opérateurs :
;   exten => _2XXXX → interop_trunk_op2
;   exten => _3XXXX → interop_trunk_op3
;   exten => _X.    → Congestion
; ============================================================================
;
; ============================================================================
; ZADARMA — entrant depuis l'opérateur externe
;
; Zadarma présente l'appel sur votre login SIP ou sur un de vos DID. On route
; tout vers le mobile GSM par défaut ; adaptez DEST_ZADARMA_IN si vous préférez
; sonner un softphone (100 / 200) ou un SVI.
; ============================================================================
```

```
[zadarma_in]
; ⚠ LE CANAL DOIT GARDER LE NUMERO DE DESTINATION COMME EXTENSION.
; [2026-09-09] Ce contexte routait tout vers une extension « s ». Mesure a l AMI,
; deux Originate identiques a un contexte pres, evenement DialBegin brut :
;     via internal    Exten=100101  DestExten=100101   -> le telephone sonne
;     via zadarma_in  Exten=s       DestExten=s        -> il ne sonne pas
; Le DestExten du canal PJSIP cree par le Dial HERITE de l extension du canal
; APPELANT ; il ne vient PAS de la chaine composee (« 100101@gsm_msc », qui est
; dans DialString). Or le modem du banc (tools/osmo-phonesim-banc.py) ne fait
; sonner le telephone que sur « DialBegin dont DestExten == notre MSISDN ».
; Poser « s » rendait donc l appel entrant invisible pour le combine : le mobile
; etait bien pagine par le MSC, mais aucun RING ne partait vers ModemManager,
; personne ne decrochait, et l appelant exterieur entendait la tonalite jusqu au
; relachement. On saute donc directement sur ${DEST_ZADARMA_IN}, et c est LUI
; qui porte le traitement : le canal garde le MSISDN comme extension, comme un
; appel venu de [internal] ou de [gsm_in].
;
; Le motif d entree ne doit JAMAIS pouvoir matcher un seul caractere : « h » est
; l extension speciale du RACCROCHAGE, et un motif qui l attrape rejoue tout le
; traitement sur un canal mort, en boucle - mesure le 09/09 sur le premier appel
; entrant reel : un nouveau MixMonitor et un nouveau Dial a chaque tour. Il
; aurait aussi avale « i », « t » et « T ». Un numero appele fait toujours au
; moins deux caracteres, et la classe se limite a « + » et aux chiffres : aucune
; extension speciale (toutes en lettres) ne peut entrer ici. Le tiret de
; « 578163-100 » n a pas besoin d y figurer, il n est pas le PREMIER caractere.
exten => _[+0-9].,1,Goto(${DEST_ZADARMA_IN},1)
exten => s,1,Goto(${DEST_ZADARMA_IN},1)
exten => i,1,NoOp(=== ZADARMA IN: numéro appelé invalide (${INVALID_EXTEN}) ===)
 same => n,Goto(${DEST_ZADARMA_IN},1)

; Le traitement, porte par le MSISDN local lui-meme. Le motif est a longueur
; fixe, donc plus specifique que le _[+0-9]. ci-dessus : pas d ambiguite quand
; le Goto arrive sur 100101 (meme mecanique que face au _X. ailleurs dans ce
; fichier). Si DEST_ZADARMA_IN sortait de ce plan, c est ce motif qu il faudrait
; elargir.
exten => ___MSISDN_PFX____OPERATOR_ID__XX,1,NoOp(=== ZADARMA IN: appelé=${EXTEN} - CallerID: ${CALLERID(all)} ===)
 ; Le « + » de l E.164 sur le numero APPELANT : le MSC ne sait pas le porter
 ; jusqu a l ecran du mobile et certains chemins le rejettent. On garde les
 ; chiffres seuls. FILTER plutot qu un ${CALLERID(num):1} aveugle : un appelant
 ; qui arrive deja sans « + » ne doit pas perdre son premier chiffre.
 same => n,Set(CALLERID(num)=${FILTER(0-9,${CALLERID(num)})})
 same => n,Set(CALLERID(name)=${CALLERID(num)})
 same => n,Gosub(sub-record,s,1(zadarma_in_${CALLERID(num)}))
 ; 45 s et non 30 : l etablissement d un appel TERMINE passe par un paging sur
 ; l air, et la chaine QEMU/Calypso met facilement 10 a 15 s avant la premiere
 ; sonnerie. A 30 s Zadarma raccrochait avant que le mobile ne sonne vraiment.
 same => n,Dial(PJSIP/${EXTEN}@gsm_msc,45,rT)
 same => n,NoOp(=== ZADARMA IN: échec ${DIALSTATUS} vers ${EXTEN} ===)
 same => n,Hangup()

[interop_out]

; [2026-08-26] CE BLOC ETAIT ECRIT DEUX FOIS, et la seconde en-tete avait perdu
; son retour a la ligne : " same => n,Hangup()[interop_out]". Asterisk ne voyait
; donc PAS une nouvelle section - il ajoutait un second _X. dans le contexte
; deja ouvert, et refusait les trois priorites :
;     WARNING pbx.c:7148 add_priority: Unable to register extension '_X.'
;             priority 1 in 'interop_out', already in use
; Le fourre-tout restait quand meme en place (la premiere copie), donc rien ne
; cassait visiblement - juste six avertissements a chaque chargement, qui
; noyaient ceux qui comptent.
exten => _X.,1,NoOp(=== INTEROP OUT: inconnu ${EXTEN} ===)
 same => n,Congestion()
 same => n,Hangup()
```

3.47 /opt/GSM/osmo-operator/configs/info.html

16791 octets, 196 lignes → 196 lignes

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width,initial-scale=1">
<title>osmo-operator · help</title>
<!-- [2026-10-23] Rewritten. Served by the dashboard (http://127.0.0.1:8080/info.html,
 ↪  iso_modules/60-dashboard.sh)
     and opened by /usr/local/bin/osmo-info : Firefox is a snap and cannot read file:///usr/share (the old icon
     opened an empty page). The copy in /usr/share/osmo-operator is only the offline fallback. -->
<style>
:root { --bg:#f6f7f9; --card:#fff; --fg:#1d2330; --mute:#667085; --line:#d9dde5; --acc:#2f6fde; --code:#eef1f6;
 ↪  --warn:#a15c00; }
@media (prefers-color-scheme: dark) {
  :root { --bg:#12151b; --card:#1b1f27; --fg:#e3e6eb; --mute:#8d96a5; --line:#2c3340; --acc:#6b9cff;
   ↪  --code:#242a35; --warn:#f0b35a; }
}
* { box-sizing:border-box; }
body { margin:0; background:var(--bg); color:var(--fg); font:15px/1.55 system-ui,sans-serif; }
header { position:sticky; top:0; z-index:2; background:var(--card); border-bottom:1px solid var(--line);
 ↪  padding:10px 16px; display:flex; flex-wrap:wrap; gap:6px 14px; align-items:center; }
header b { font-size:16px; margin-right:8px; }
header a { color:var(--mute); text-decoration:none; font-size:14px; }
header a:hover { color:var(--acc); }
main { max-width:1100px; margin:auto; padding:16px; }
section { background:var(--card); border:1px solid var(--line); border-radius:10px; padding:16px 20px; margin:0 0
 ↪  16px; }
h1 { font-size:24px; margin:8px 0 4px; }
h2 { font-size:18px; margin:0 0 10px; }
h3 { font-size:15px; margin:16px 0 6px; }
p { margin:6px 0; }
a { color:var(--acc); }
code, pre { font:13px/1.5 ui-monospace,monospace; background:var(--code); border-radius:5px; }
code { padding:1px 5px; }
pre { padding:10px 12px; overflow-x:auto; margin:8px 0; }
table { width:100%; border-collapse:collapse; font-size:14px; }
th, td { text-align:left; vertical-align:top; padding:7px 8px; border-top:1px solid var(--line); }
th { color:var(--mute); font-weight:600; border-top:0; }
td:first-child { white-space:nowrap; }
.lead { color:var(--mute); }
.warn { color:var(--warn); }
.ref { font-size:12px; vertical-align:super; text-decoration:none; }
ol.refs { padding-left:22px; font-size:14px; } ol.refs li { margin:3px 0; }
.grid { display:grid; grid-template-columns:repeat(auto-fill,minmax(250px,1fr)); gap:10px; }
.tile { border:1px solid var(--line); border-radius:8px; padding:10px 12px; }
.tile a { font-weight:600; text-decoration:none; } .tile p { color:var(--mute); font-size:13px; margin:4px 0 0; }
</style>
</head>
<body>
<header>
  <b>osmo-operator · help</b>
  <a href="#modes">Modes</a><a href="#modules">Modules</a><a href="#notebooks">Notebooks</a>
  <a href="#local">On this machine</a><a href="#sites">Sites</a><a href="#repos">Repositories</a><a
   ↪  href="#refs">References</a>
</header>
<main>

<h1>Running the bench</h1>
<p class="lead">A complete 2G network with no radio hardware: Osmocom core, an emulated Calypso phone (QEMU for
 ↪  the ARM,
<code>c54x_exe</code> for the TI DSP and its mask ROM), plus a 4G cell, a SIP softphone and a postmarketOS phone.
Everything lives in <code>/opt/GSM/osmo-operator</code><a class="ref" href="#r1">[1]</a>.
```

```
Nothing telecom starts at boot: starting a network is your gesture<a class="ref" href="#r12">[12]</a>.</p>

<section id="modes">
<h2>Modes: one command per bench</h2>
<p>All modes go through <code>./start.sh</code> (<code>start-native.sh</code> is the same script, kept as a link).
Add <code>--regen</code> to rebuild <code>/etc/osmocom</code> from the templates, <code>--no-attach</code> to
 ↪  stay out of tmux.
Full option list: <code>./start.sh --help</code><a class="ref" href="#r2">[2]</a>.</p>
<table>
<tr><th>Command</th><th>What runs</th><th>State</th></tr>
<tr><td><code>./start.sh --bts-classique --ms</code></td>
<td><b>The default (red phone icon).</b> Classic BTS: <code>osmo-bts-trx</code> talks to
 ↪  <code>pont/pont_dsp.py</code>; the emulated
phone runs the osmocom-bb <code>layer1</code> firmware and <code>mobile</code>, on the real DSP ROM<a class="ref"
 ↪  href="#r3">[3]</a>.</td>
<td>LU, SMS, calls</td></tr>
<tr><td><code>./start.sh --bts-classique --ms-magnetite</code></td>
<td>Same network, but the phone is a Motorola C139 running the <b>real TI firmware</b> (FreeCalypso Magnetite):
 ↪  its own
L2/L3 and UI, colour screen in <code>tools/osmo-c123-gtk.py --modele c139</code>, traces by
 ↪  <code>rvinterf</code><a class="ref" href="#r4">[4]</a>.
Add <code>--gapk</code> for voice on the host.</td>
<td>LU, SMS, calls in A5/0: <code>ENCRYPTION="a5 0" ./start.sh --regen …</code></td></tr>
<tr><td><code>./start.sh --bts --ms --ts 2</code></td>
<td>The Calypso of QEMU <b>is the BTS radio</b> (firmware <code>trx</code> + host <code>transceiver</code>); a
 ↪  second Calypso is the
phone and camps on it over the air (<code>pont/air.py</code>). <code>--ts 2</code>: one Calypso per timeslot (TS0
 ↪  CCCH/SDCCH, TS1 TCH/F).</td>
<td class="warn">camps, LU; no call yet (2nd TRX not frame-aligned)</td></tr>
<tr><td><code>./start.sh --bts --ms --ts 2/1</code></td><td>TS0 and TS1 served by a single Calypso TRX (also the
 ↪  default of <code>--bts</code>).</td><td>calls (audible on 20/10)</td></tr>
<tr><td><code>./start.sh --bts --ms --ts 1</code></td><td>TS0 only (CCCH + SDCCH/4).</td><td>LU, SMS, no
 ↪  call</td></tr>
<tr><td><code>./start.sh --bts --ms-magnetite</code></td><td>Calypso TRX + the C139 Magnetite over the
 ↪  air.</td><td>experimental</td></tr>
<tr><td><code>./start.sh --grgsm</code></td><td>gr-gsm layer 1 in QEMU + <code>grgsm_exe</code> instead of the
 ↪  DSP<a class="ref" href="#r5">[5]</a>.</td><td>end to end</td></tr>
<tr><td><code>--vocoder-rom</code> / <code>--gapk</code></td><td>Speech codec in the DSP mask ROM (default) or on
 ↪  the host (osmo-gapk / libgsm).</td><td>--gapk keeps real time</td></tr>
<tr><td><code>--sap</code>, <code>--sap=pcsc</code></td><td>SIM over SAP: the card in the PC/SC reader
 ↪  (sysmoISIM) or the soft SIM<a class="ref" href="#r6">[6]</a>.</td><td></td></tr>
<tr><td><code>--wan</code>, <code>--payload-ism</code></td><td>Link several nodes (SS7 hub, WireGuard); SMS text
 ↪  over an ISM radio link<a class="ref" href="#r7">[7]</a>.</td><td></td></tr>
<tr><td><code>banc-rssi</code></td><td>The osmocom-bb <code>rssi</code> app on a 4th Calypso (spectrum, sync,
 ↪  per-timeslot levels), fed by the pont in
<code>--bts-classique</code>; <code>banc-rssi --stop</code>. Screen: <code>c123-3</code>.</td><td>DL OK, UL view
 ↪  in progress</td></tr>
<tr><td><code>./start.sh --stop</code> · <code>--status</code></td><td>Stop / query the whole
 ↪  stack.</td><td></td></tr>
</table>
<h3>Safeguards</h3>
<p>BTS#0 runs under a watchdog (<code>BTS_GARDIEN=1</code>, <code>qosmo/run_modules/60-bts.sh</code>): if
 ↪  <code>osmo-bts-trx</code> exits on
“PC clock skew too high” it is restarted after 3&amp;nbsp;s. The pont caps the pause it asks of the BTS
(<code>PONT_CLCK_PAUSE_PLAFOND=38</code>, <code>pont/trx.py</code>). Do not compile during a run: a loaded host
 ↪  still trips the clock.</p>
</section>

<section id="modules">
<h2>Modules: icon, launcher, CLI</h2>
<table>
<tr><th>Dock icon</th><th>CLI</th><th>Code</th></tr>
<tr><td>Red phone: the GSM bench</td><td><code>launch.sh --bts-classique --ms</code> · <code>systemctl start
 ↪  osmo-banc</code></td><td><code>launch.sh</code>, <code>services/osmo-banc.service</code></td></tr>
<tr><td>Antenna: SS7 multi-operator</td><td><code>systemctl start osmo-multi</code> · <code>launch/start-multi.sh
 ↪  --status</code></td><td>operators 2 and 3 in Docker, inter-STP hub<a class="ref" href="#r8">[8]</a></td></tr>
```

```
<tr><td>4G</td><td><code>osmo-lte toggle | status</code> · <code>osmo-epc
 ↪  start</code></td><td><code>tools/osmo-lte.sh</code>, Open5GS + srsRAN<a class="ref"
 ↪  href="#r9">[9]</a></td></tr>
<tr><td>Supplements</td><td><code>sudo ./addition.sh --multi | --opencl | --claude | --extras | --sim</code><br>
<code>--gnuarm | --magnetite | --trx | --fernvale | --simcom |
 ↪  --firmwares</code></td><td><code>addition.sh</code>, <code>tools/osmo-firmwares-install.sh</code></td></tr>
<tr><td>postmarketOS phone</td><td><code>osmo-pmos up | stop |
 ↪  status</code></td><td><code>tools/osmo-pmos.sh</code></td></tr>
<tr><td>Linphone, Wireshark (bench)</td><td><code>linphone</code> · <code>osmo-wireshark-root</code> (GSMTAP
 ↪  udp/4729)</td><td><code>tools/osmo-wireshark-root.sh</code></td></tr>
<tr><td>Update</td><td><code>sudo ./update.sh</code></td><td>git pull of the embedded trees</td></tr>
<tr><td>Tutorial</td><td><code>osmo-tutorial</code></td><td><a href="tutorial.html">tutorial.html</a></td></tr>
<tr><td>? (this page)</td><td><code>osmo-info</code></td><td><code>configs/info.html</code></td></tr>
</table>
<h3>Day to day</h3>
<pre>sudo tmux attach -t calypso          # the bench consoles
telnet 127.0.0.1 4254               # MSC VTY: show subscriber cache
telnet 127.0.0.1 4242               # BSC VTY
telnet 127.0.0.1 4247               # mobile VTY (osmocom phone; not with Magnetite)
./run.sh --logs   (in /opt/GSM/c54x_exe)   # DSP bench logs: /tmp/c54x-pont/*.log</pre>
<p>Settings: <code>globals.conf</code> (MCC, MNC, ARFCN, A5…) or the settings page below; configs are generated
 ↪  into <code>/etc/osmocom</code> by <code>--regen</code><a class="ref" href="#r10">[10]</a>.</p>
</section>

<section id="notebooks">
<h2>Jupyter notebooks</h2>
<div class="grid">
<div class="tile"><a href="https://github.com/bbaranoff/osmo-operator/blob/main/wiki/installer-osmo-jusquau-
 ↪  dsp.ipynb">installer-osmo-jusquau-dsp.ipynb</a>
<p>From a fresh Ubuntu 24.04 to a call on the DSP, one cell per step, with checks; optional Magnetite build.
<a href="https://nbviewer.org/github/bbaranoff/osmo-operator/blob/main/wiki/installer-osmo-jusquau-dsp.ipynb">
 ↪  nbviewer</a></p></div>
<div class="tile"><a href="https://github.com/bbaranoff/osmo-operator/blob/main/wiki/lancer-pas-a-pas.ipynb">
 ↪  lancer-pas-a-pas.ipynb</a>
<p>The bench started process by process: HLR, MGW, MSC, BSC, BTS, QEMU, osmocon, mobile, pont; classic BTS, rssi.
<a href="https://nbviewer.org/github/bbaranoff/osmo-operator/blob/main/wiki/lancer-pas-a-pas.ipynb">nbviewer</a
 ↪  ></p></div>
<div class="tile"><a
 ↪  href="https://github.com/bbaranoff/osmo-operator/blob/main/wiki/README-notebook.md">README-notebook</a>
<p>How to run them: disposable container, fresh VM, or unattended with <code>nbconvert</code><a class="ref"
 ↪  href="#r11">[11]</a>.</p></div>
</div>
<pre># locally (the notebooks are in /opt/GSM/osmo-operator/wiki)
python3 -m venv ~/.venv-jupyter && ~/.venv-jupyter/bin/pip install jupyterlab
~/.venv-jupyter/bin/jupyter lab /opt/GSM/osmo-operator/wiki</pre>
</section>

<section id="local">
<h2>On this machine</h2>
<div class="grid">
<div class="tile"><a href="http://127.0.0.1:8080/">Dashboard</a><p>http://127.0.0.1:8080 · subscribers, VTY,
 ↪  calls, logs (osmo-egprs-web)</p></div>
<div class="tile"><a href="http://127.0.0.1:8090/">Settings</a><p>http://127.0.0.1:8090 · globals.conf, Osmocom
 ↪  configs, Calypso instances (osmo-reglages)</p></div>
<div class="tile"><a href="tutorial.html">Tutorial</a><p>The quick start, served by the dashboard</p></div>
<div class="tile"><a href="https://github.com/bbaranoff/osmo-operator/wiki">Wiki</a><p>The detailed reference,
 ↪  page by page<a class="ref" href="#r13">[13]</a></p></div>
</div>
</section>

<section id="sites">
<h2>Sites</h2>
<div class="grid">
<div class="tile"><a href="https://bbaranoff.github.io">bbaranoff.github.io</a><p>Bastien Baranoff · resources,
 ↪  articles and the related projects</p></div>
```

```
<div class="tile"><a href="https://pl4y.store">pl4y.store</a><p>pl4y (<a
 ↪  href="https://github.com/bbaranoff/pl4y">github.com/bbaranoff/pl4y</a>)</p></div>
<div class="tile"><a href="https://github.com/bbaranoff">github.com/bbaranoff</a><p>All the
 ↪  repositories</p></div>
<div class="tile"><a href="https://github.com/bbaranoff/osmo-operator/releases/latest">Latest release</a><p>The
 ↪  desktop ISO, in parts, with SHA256SUMS</p></div>
</div>
</section>

<section id="repos">
<h2>Repositories</h2>
<table>
<tr><th>Repository</th><th>Role in the bench</th></tr>
<tr><td><a href="https://github.com/bbaranoff/osmo-operator">osmo-operator</a></td><td>This bench: launchers,
 ↪  configs, pont, ISO, supplements, docs</td></tr>
<tr><td><a href="https://github.com/bbaranoff/qosmo">qosmo</a></td><td>QEMU with the Calypso machine (ARM, TPU,
 ↪  TSP, BSP, ST7558/C139 screens), run modules</td></tr>
<tr><td><a href="https://github.com/bbaranoff/c54x_exe">c54x_exe</a></td><td>The TI C54x DSP outside QEMU, on the
 ↪  mask ROM; <code>run.sh</code> drives the instances</td></tr>
<tr><td><a href="https://github.com/bbaranoff/grgsm_exe">grgsm_exe</a></td><td>The gr-gsm layer 1
 ↪  bridge</td></tr>
<tr><td><a href="https://github.com/bbaranoff/firmware">firmware</a></td><td>Prebuilt osmocom-bb Calypso
 ↪  firmware</td></tr>
<tr><td><a href="https://github.com/bbaranoff/osmocom-bb-trx">osmocom-bb-trx</a></td><td>osmocom-bb jolly/testing
 ↪  with TX (Docker image <code>ghcr.io/bbaranoff/osmocom-bb-trx:main</code>)</td></tr>
<tr><td><a href="https://github.com/bbaranoff/osmo-egprs-web">osmo-egprs-web</a> · <a
 ↪  href="https://github.com/bbaranoff/osmo_egprs">osmo_egprs</a></td><td>The dashboard; the EGPRS
 ↪  workflow</td></tr>
<tr><td><a href="https://github.com/bbaranoff/fernly">fernly</a> · <a
 ↪  href="https://github.com/bbaranoff/fernvale-nuttx">fernvale-nuttx</a> · <a
 ↪  href="https://github.com/bbaranoff/osmocom-bb-mtk-layer1">osmocom-bb-mtk-layer1</a></td><td>MediaTek MT6260
 ↪  (Fernvale): reversing OS, NuttX, osmocom-bb layer 1 (<code>addition.sh --fernvale</code>)</td></tr>
<tr><td><a href="https://github.com/bbaranoff/simcom">simcom</a></td><td>SIMCom module documentation and SDKs
 ↪  (<code>addition.sh --simcom</code>)</td></tr>
<tr><td><a href="https://github.com/bbaranoff/deka">deka</a> · <a
 ↪  href="https://github.com/bbaranoff/a51_tools">a51_tools</a></td><td>A5/1 tools (OpenCL supplement)</td></tr>
<tr><td>Upstream</td><td><a href="https://gitea.osmocom.org/">Osmocom</a> · <a
 ↪  href="https://www.freecalypso.org/">FreeCalypso</a> (<a
 ↪  href="https://hg.themyscira.org/fc-magnetite">fc-magnetite</a>) ·
<a href="https://github.com/axilirator/gnu-arm-installer">gnu-arm-installer</a> (arm-elf-gcc 4.5.2)</td></tr>
</table>
</section>

<section id="refs">
<h2>References</h2>
<ol class="refs">
<li id="r1"><a href="https://github.com/bbaranoff/osmo-operator#readme">README</a> — start here, by goal (ISO,
 ↪  Docker, native, ARM, DSP, 4G, SS7)</li>
<li id="r2"><a
 ↪  href="https://github.com/bbaranoff/osmo-operator/blob/main/wiki/Start-native.md">wiki/Start-native</a> — the
 ↪  launcher step by step, options manual</li>
<li id="r3"><a href="https://github.com/bbaranoff/osmo-operator/blob/main/doc/README-dsp.md">doc/README-dsp</a> —
 ↪  the real C54x DSP on the mask ROM</li>
<li id="r4"><a href="https://github.com/bbaranoff/osmo-operator/blob/main/doc/magnetite.md">doc/magnetite</a> —
 ↪  FreeCalypso Magnetite in qosmo: build, flash image, state</li>
<li id="r5"><a
 ↪  href="https://github.com/bbaranoff/osmo-operator/blob/main/doc/README-grgsm.md">doc/README-grgsm</a> — the
 ↪  gr-gsm layer 1</li>
<li id="r6"><a href="https://github.com/bbaranoff/osmo-operator/blob/main/wiki/SIM.md">wiki/SIM</a> — Omnikey
 ↪  reader, sysmoISIM, soft SIM over SAP</li>
<li id="r7"><a href="https://github.com/bbaranoff/osmo-operator/blob/main/wiki/WAN.md">wiki/WAN</a> · <a
 ↪  href="https://github.com/bbaranoff/osmo-operator/blob/main/wiki/Payload-ISM.md">wiki/Payload-ISM</a> —
 ↪  linking nodes</li>
<li id="r8"><a href="https://github.com/bbaranoff/osmo-operator/blob/main/doc/README-ss7.md">doc/README-ss7</a> —
 ↪  multi-operator, SS7 interconnect</li>
```

```
<li id="r9"><a href="https://github.com/bbaranoff/osmo-operator/blob/main/doc/README-lte.md">doc/README-lte</a> —
 ↪  4G, CSFB, the postmarketOS phone</li>
<li id="r10"><a
 ↪  href="https://github.com/bbaranoff/osmo-operator/blob/main/wiki/Environnement.md">wiki/Environnement</a> —
 ↪  which variable wins, what reaches QEMU</li>
<li id="r11"><a
 ↪  href="https://github.com/bbaranoff/osmo-operator/blob/main/wiki/README-notebook.md">wiki/README-notebook</a>
 ↪  — running the notebooks</li>
<li id="r12"><a href="https://github.com/bbaranoff/osmo-operator/blob/main/services/80-osmo-operator.preset">
 ↪  services/80-osmo-operator.preset</a> — the boot policy
 ↪  (<code>tools/osmo-telecom-boot.sh</code>)</li>
<li id="r13"><a href="https://github.com/bbaranoff/osmo-operator/blob/main/wiki/Home.md">wiki/Home</a> · <a
 ↪  href="https://github.com/bbaranoff/osmo-operator/blob/main/doc/README-usage.md">doc/README-usage</a> · <a
 ↪  href="https://github.com/bbaranoff/osmo-operator/blob/main/doc/README-architecture.md">doc/README-
 ↪  architecture</a> · <a
 ↪  href="https://github.com/bbaranoff/osmo-operator/blob/main/doc/README-tests.md">doc/README-tests</a></li>
</ol>
</section>

</main>
</body>
</html>
```

3.48 /opt/GSM/osmo-operator/configs/linphonerc

4556 octets, 121 lignes → 121 lignes

```
# ============================================================================
# linphonerc - compte SIP pre-provisionne du client Linphone (ISO --desktop)
#
# Depose par build-iso.sh dans ~/.config/linphone/linphonerc des comptes root
# et osmocom, et dans /etc/skel. Sans ce fichier l'ISO livrait Linphone VIERGE :
# l'operateur creait le compte a la main, a chaque demarrage, et tombait sur les
# deux pieges ci-dessous.
#
#   PIEGE 1 - LINPHONE NE RELIT PAS SA CONFIG A CHAUD.
#   Le compte ecrit dans linphonerc pendant que Linphone TOURNE n'est jamais
#   charge : l'instance en cours garde son etat en memoire et REECRIT le fichier
#   en sortant. Symptome constate le 2026-08-31 sur ce banc : linphonerc correct
#   (linphone_A / tester / bon proxy, reg_sendregister=1), process bien vivant,
#   socket SIP UDP ouverte -- et pas UN REGISTER en 90 s de capture. Cote
#   Asterisk : endpoint "Unavailable", zero contact, et RIEN dans les logs (pas
#   de 401, pas de "No matching endpoint") puisque aucun paquet n'arrivait.
#   Le serveur etait sain : un REGISTER injecte a la main sur 127.0.0.1:5060
#   recevait bien son "401 Unauthorized" avec challenge digest.
#   -> Il faut REDEMARRER Linphone. D'ou ce provisionnement au build, pose
#      avant le premier lancement.
#
#   PIEGE 2 - publish=0, ET PAS 1 (voir la section [proxy_0]).
#
#   L'identifiant SIP est "linphone_A", PAS "100". Le 100 est l'extension
#   composee dans le dialplan (configs/extensions.conf), pas le nom du compte.
# ============================================================================

[sip]
# -2 = port UDP aleatoire. Surtout PAS 5060 : sur l'ISO, Linphone et Asterisk
# partagent le meme hote, et Asterisk tient deja 0.0.0.0:5060 (pjsip.conf
# [transport-udp] bind=0.0.0.0). Un port fixe a 5060 ici et le client ne
# demarre pas sa pile SIP - sans le dire clairement.
sip_port=-2
# TCP et TLS coupes : pjsip.conf ne declare QU'UN transport, [transport-udp].
# Rien n'ecoute en TCP 5060 (et chan_sip n'est meme pas charge). Un Linphone
# laisse sur son transport par defaut echoue instantanement, sans qu'aucun
# paquet n'arrive jusqu'a Asterisk - donc sans la moindre trace cote serveur.
```

```
sip_tcp_port=-1
sip_tls_port=-1
media_encryption=none
default_proxy=0

# LES DEUX POSTES SONT PROVISIONNES DANS LA MEME INSTANCE. Linphone gere
# plusieurs comptes de front : les deux s enregistrent au demarrage, et l on
# appelle de l un vers l autre (100 <-> 200) sans rien reconfigurer. Le compte
# par defaut (celui qui emet si l on ne choisit pas) est proxy_0, linphone_A,
# fixe par default_proxy=0 dans [sip] ci-dessus.
[auth_info_0]
username=linphone_A
passwd=tester
# 127.0.0.1 est valable dans LES DEUX topologies - conteneur comme ISO :
# Linphone et Asterisk partagent l'hote. Une IP de LAN (192.168.1.x) marcherait
# sur ce banc mais serait fausse des le prochain bail DHCP.
domain=127.0.0.1

[auth_info_1]
username=linphone_B
passwd=testerB
domain=127.0.0.1

[proxy_0]
reg_proxy=<sip:127.0.0.1;transport=udp>
reg_identity=sip:linphone_A@127.0.0.1
reg_expires=3600
reg_sendregister=1
# publish=0 IMPERATIF. A 1, Linphone emet un PUBLISH de presence a chaque
# enregistrement et Asterisk 18 le refuse - il n'embarque aucun gestionnaire
# pour l'evenement "presence" (c'est une brique absente de la version libre) :
#     WARNING res_pjsip_pubsub.c: pubsub_on_rx_publish_request:
#     No registered publish handler for event presence from linphone_A
# Le warning est inoffensif mais il revient a chaque REGISTER et noie
# /var/log/asterisk/messages. Rien a corriger cote Asterisk : c'est le client
# qui doit se taire.
publish=0
avpf=0
quality_reporting_enabled=0

[proxy_1]
reg_proxy=<sip:127.0.0.1;transport=udp>
reg_identity=sip:linphone_B@127.0.0.1
reg_expires=3600
reg_sendregister=1
# publish=0 pour la meme raison que proxy_0 : sans ce reglage, CHAQUE compte
# emet son PUBLISH de presence a chaque enregistrement, et le warning
# "No registered publish handler for event presence" arrive alors en DOUBLE.
publish=0
avpf=0
quality_reporting_enabled=0

# Codecs alignes sur l'endpoint Asterisk (pjsip.conf : disallow=all, puis
# allow=alaw/ulaw/gsm). Opus et speex sont coupes : proposes par defaut, ils
# ne sont jamais retenus et allongent le SDP pour rien.
[audio_codec_0]
mime=PCMA
rate=8000
channels=1
enabled=1

[audio_codec_1]
mime=PCMU
rate=8000
channels=1
enabled=1
```

```
[audio_codec_2]
mime=GSM
rate=8000
channels=1
enabled=1

[audio_codec_3]
mime=opus
rate=48000
channels=2
enabled=0

[audio_codec_4]
mime=speex
rate=16000
channels=1
enabled=0
```

3.49 /opt/GSM/osmo-operator/configs/manager-osmo.conf

820 octets, 18 lignes → 18 lignes

```
; manager-osmo.conf - le compte AMI du raccord mobile.
;
; Pose dans /etc/asterisk/manager.d/osmo.conf par start-native.sh. Il sert a
; tools/osmo-phonesim-banc.py pour ETABLIR LES APPELS demandes par le telephone
; Android : le modem du banc recoit un ATD..., et le seul moyen de faire naitre
; un appel dans Asterisk depuis l exterieur est l action Originate de l AMI.
;
; Pourquoi pas la CLI : « channel originate » vient du module res_clioriginate,
; absent de cette installation (modules.conf est en autoload = no). L AMI, lui,
; est deja active dans manager.conf - il ne lui manquait qu un compte.
;
; L acces est limite a la machine : deny tout, permit la boucle locale.
[osmo]
secret = osmocom1
deny = 0.0.0.0/0.0.0.0
permit = 127.0.0.1/255.255.255.255
read = system,call,verbose
write = system,call,originate
```

3.50 /opt/GSM/osmo-operator/configs/mobile.cfg

3515 octets, 143 lignes → 143 lignes

```
# mobile.cfg.template — avec placeholders
# KI : "ki comp128 XX XX XX ... XX"  16 octets, octet 15=ms_idx, 16=op_id
#
# AUDIO :
#   tch-voice l1phy → mobile décode les TCH-FR depuis les bursts TRXD
#   fake_trx transporte le payload → trxcon → mobile → ALSA
#   Requis dans le container : libasound2-plugins + /dev/snd + PULSE_SERVER
!
line vty
 no login
 bind 127.0.0.1 4247
!
gps device /dev/ttyACM0
gps baudrate default
no gps enable
!
```

```
no hide-default
!
!
log gsmtap 127.0.0.1
 logging filter all 1
 logging color 0
 logging timestamp 0
 logging print category 1
 logging level mm debug
 logging level rr debug
 logging level cc debug
 logging level sms debug
 logging level plmn debug
 logging level sim debug
 logging level sap debug
!
ms 1
 layer2-socket /tmp/osmocom_l2
! [2026-10-14] SIM par SAP : softSIM sert la carte du lecteur PC/SC (Omnikey,
! sysmoISIM) ou la SIM logicielle sur le socket SAP. Sans --sap, start-native.sh
! remet « sim test » (la SIM de test ci-dessous) ; le socket est celui de start_sap.
 sap-socket /tmp/osmocom_sap
 sim sap
 network-selection-mode auto
 imei __IMEI__ 0
 imei-fixed
 no emergency-imsi
 sms-service-center __SMS_SC__
 no call-waiting
 no auto-answer
 no force-rekey
 no clip
 no clir
 tx-power auto
 no simulated-delay
 stick __ARFCN__
 location-updating
 neighbour-measurement
 codec full-speed prefer
 codec half-speed
 no abbrev
 support
  sms
  a5/1
  a5/2
  p-gsm
  e-gsm
  r-gsm
  no gsm-850
  dcs
  no pcs
  class-900 4
  class-850 4
  class-dcs 1
  class-pcs 1
  channel-capability sdcch+tchf+tchh
  full-speech-v1
  full-speech-v2
  half-speech-v1
  min-rxlev -106
  dsc-max 90
  no skip-max-per-band
 test-sim
  imsi __IMSI__
  ki __SIM_ALGO__ __KI__
  no barred-access
```

```
  rplmn __MCC__ __MNC__
 tch-voice
  io-handler gapk
! [2026-08-27] FORMAT DE TRAME TCH : rtp, PAS ti.
! « ti » est le format Texas Instruments des vrais telephones Calypso, ou
! c'est le DSP qui fait le vocodage et rend ses trames dans SA disposition.
! Ici le DSP ne fait pas le codage canal : c'est pont.py qui l'assure, avec
! libosmocoding (gsm0503_tch_fr_encode/decode), laquelle parle la disposition
! RTP/RFC3551 — 33 octets ouverts par la signature 0xD.
! Les deux bouts ne se comprenaient donc pas, DANS LES DEUX SENS, et le son
! sortait robotise sans qu'aucun compteur ne bronche : les trames passaient,
! elles etaient simplement lues avec le mauvais plan.
! MESURE : anneau descendant (ecrit par le pont) d2 2c fd a2 1b ec 20 39...
!          -> signature 0xD, contenu qui varie = vraie parole ;
!          anneau montant (ecrit par le mobile en « ti ») 83 00 05 e1 e0 08
!          10 00 22 01 ff ff ff ff ff ff e0 00... -> pas de 0xD, plages de
!          ff et de 00 = une trame a champs, la disposition TI.
  io-tch-format rtp
  alsa-output-dev gsm_out
  alsa-input-dev gsm_in
 no shutdown
!
! GSMTAP configuration
!
gsmtap
 remote-host __INTER_NET_GATEWAY__
 no local-host
 lchan sacch
 lchan lsacch
 lchan sacch/4
 lchan sacch/8
 lchan sacch/f
 lchan sacch/h
 lchan unknown
 lchan bcch
 lchan ccch
 lchan rach
 lchan agch
 lchan pch
 lchan sdcch
 lchan sdcch/4
 lchan sdcch/8
 lchan facch/f
 lchan facch/h
 lchan pacch
 lchan cbch
 lchan pdch
 lchan pttch
 lchan tch/f
 lchan tch/h
 category gprs dl-unknown
 category gprs dl-dummy
 category gprs dl-ctrl
 category gprs dl-data-gprs
 category gprs dl-data-egprs
 category gprs ul-unknown
 category gprs ul-dummy
 category gprs ul-ctrl
 category gprs ul-data-gprs
 category gprs ul-data-egprs
end
```

3.51 /opt/GSM/osmo-operator/configs/modules.conf

7500 octets, 153 lignes → 153 lignes

```
[modules]
autoload = no

; ── BLOQUÉS — pas de SRTP/DTLS dans un core GSM ──────────────────────
; res_srtp est chargé comme dépendance par res_rtp_asterisk même avec autoload=no.
; Il inspecte les paquets RTP entrants et certains frames GSM-FR déclenchent :
;   "Received SSL traffic on RTP instance without an SSL session"
; → l'audio est jeté → pas de son.
; noload empêche le chargement même par dépendance.
noload = res_srtp.so
noload = res_hep.so
noload = res_hep_pjsip.so
noload = res_hep_rtcp.so

; ── Applications ─────────────────────────────────────────────────────
load = app_bridgewait.so
load = app_dial.so
load = app_playback.so
load = app_stack.so
load = app_verbose.so
load = app_voicemail.so
load = app_directory.so
load = app_confbridge.so
load = app_queue.so
load = app_echo.so
load = app_mixmonitor.so

; ── Functions audio (jitterbuffer : lisse le transcodage forcé par l'enregistrement) ──
load = func_jitterbuffer.so

; ── Bridging ─────────────────────────────────────────────────────────
load = bridge_builtin_features.so
load = bridge_builtin_interval_features.so
load = bridge_holding.so
load = bridge_native_rtp.so
load = bridge_simple.so
load = bridge_softmix.so

; ── CDR ──────────────────────────────────────────────────────────────
load = cdr_custom.so
; [2026-09-09] AJOUTE - AUCUN APPEL N ETAIT ENREGISTRE NULLE PART.
; « cdr show status » listait « Registered Backends » VIDE : cdr_custom sans
; cdr_custom.conf n ecrit rien, et cdr_csv n etait pas charge. Master.csv
; restait donc absent alors qu Asterisk comptait 23 appels traites. Sans trace
; par appel (source, destination, disposition, duree), impossible de comparer
; deux tentatives - « la deuxieme a marche, pas les autres » n a aucun support.
; cdr_csv ecrit /var/log/asterisk/cdr-csv/Master.csv, une ligne par appel.
load = cdr_csv.so

; ── Channels ─────────────────────────────────────────────────────────
load = chan_bridge_media.so
load = chan_pjsip.so
; [2026-08-13] chan_iax2 RETIRÉ — inutilisé, mesuré et non supposé :
;   asterisk -rx "iax2 show peers"        -> 0 iax2 peers
;   asterisk -rx "module show like iax"   -> Use Count 0
;   aucun trunk/extension IAX dans extensions.conf. Tout passe par PJSIP.
;
; ⚠ EFFET DE BORD, à connaître avant de chercher une sonde disparue :
; chan_iax2.so est le SEUL module du répertoire qui importe `jb_setoutput`
; (vérifié : nm -D -u sur tous les .so de /usr/lib/x86_64-linux-gnu/asterisk/
; modules). C'est donc lui qui branche les sorties de journal du moteur
; jitterbuf — moteur qui vit dans /usr/sbin/asterisk (la chaîne "Resyncing the
```

```
; jb" y est, elle n'est PAS dans chan_iax2.so) et qui sert le JITTERBUFFER
; adaptatif de func_jitterbuffer, pas IAX.
; Conséquence : sans chan_iax2, le jitterbuffer TOURNE TOUJOURS mais devient
; MUET — plus une seule ligne « Resyncing the jb ... » dans /var/log/asterisk/
; messages. C'était la seule sonde du JB. Pour la récupérer : remettre `load`.
noload = chan_iax2.so

; ── Codecs ──────────────────────────────────────────────────────────────
load = codec_gsm.so
load = codec_resample.so
load = codec_alaw.so
load = codec_ulaw.so
load = codec_g722.so

; ── Formats ─────────────────────────────────────────────────────────────
load = format_gsm.so
load = format_pcm.so
load = format_wav_gsm.so
load = format_wav.so

; ── Functions ───────────────────────────────────────────────────────────
load = func_callerid.so
load = func_cdr.so
load = func_pjsip_endpoint.so
load = func_sorcery.so
load = func_devstate.so
load = func_strings.so
; [2026-09-09] MANQUAIT - ET IL CASSAIT TOUT APPEL VERS UN MOBILE LOCAL.
; func_logic fournit ${IF()}, utilise TROIS fois dans extensions.conf :
;     same => n,Set(ANNU_NUM=${IF($["${CALLERID(num)}" != ""]?${CALLERID(num)}:0)})
;     same => n,Gosub(sub-annuaire,${ANNU_NUM},1)
; dans [internal], [gsm_in] et [interop_in], sur le motif _1001XX. Le banc est
; en « autoload = no » : la fonction n etait pas enregistree, la substitution
; rendait une chaine VIDE, et le Gosub partait sur une extension vide - donc en
; echec, donc raccrochage. Trace, a chaque tentative d appel :
;     ERROR pbx_functions.c: Function IF not registered
; Le module etait pourtant bien present sur le disque
; (/usr/lib/x86_64-linux-gnu/asterisk/modules/func_logic.so) : rien a installer,
; juste cette ligne. Verifier : asterisk -rx "core show function IF".
load = func_logic.so

; ── PBX ─────────────────────────────────────────────────────────────────
load = pbx_config.so

; ── Resources ───────────────────────────────────────────────────────────
load = res_http_websocket.so
load = res_musiconhold.so
load = res_pjproject.so
load = res_pjsip_acl.so
load = res_pjsip_authenticator_digest.so
load = res_pjsip_caller_id.so
load = res_pjsip_dialog_info_body_generator.so
load = res_pjsip_diversion.so
load = res_pjsip_dtmf_info.so
load = res_pjsip_endpoint_identifier_anonymous.so
load = res_pjsip_endpoint_identifier_ip.so
load = res_pjsip_endpoint_identifier_user.so
load = res_pjsip_exten_state.so
load = res_pjsip_header_funcs.so
load = res_pjsip_logger.so
load = res_pjsip_messaging.so
load = res_pjsip_mwi_body_generator.so
load = res_pjsip_mwi.so
load = res_pjsip_nat.so
load = res_pjsip_notify.so
load = res_pjsip_one_touch_record_info.so
```

```
load = res_pjsip_outbound_authenticator_digest.so
load = res_pjsip_outbound_publish.so
load = res_pjsip_outbound_registration.so
load = res_pjsip_path.so
load = res_pjsip_pidf_body_generator.so
load = res_pjsip_pidf_digium_body_supplement.so
load = res_pjsip_pidf_eyebeam_body_supplement.so
load = res_pjsip_publish_asterisk.so
load = res_pjsip_pubsub.so
load = res_pjsip_refer.so
load = res_pjsip_registrar.so
load = res_pjsip_rfc3326.so
load = res_pjsip_sdp_rtp.so
load = res_pjsip_send_to_voicemail.so
load = res_pjsip_session.so
load = res_pjsip.so
load = res_pjsip_t38.so
load = res_pjsip_transport_websocket.so
load = res_pjsip_xpidf_body_generator.so
load = res_rtp_asterisk.so
load = res_sorcery_astdb.so
load = res_sorcery_config.so
load = res_sorcery_memory.so
load = res_sorcery_realtime.so
load = res_timing_timerfd.so
```

3.52 /opt/GSM/osmo-operator/configs/osmo-bsc.cfg

12286 octets, 399 lignes → 393 lignes (1 groupes compactés)

```
!
! osmo-bsc.cfg — BSC opérateur __OPERATOR_ID__ (MCC __MCC__ / MNC __MNC__)
!
! Pour opérateur 1: utiliser __CONTAINER_IP__ = 172.20.0.11
! Pour opérateur 2: utiliser __CONTAINER_IP__ = 172.20.0.12
!
log file /var/log/osmocom/osmo-bsc.log
 logging color 1
 logging timestamp 1
 logging filter all 1
 logging print category-hex 0
 logging print category 1
 logging print thread-id 0
 logging print level 1
 logging print file basename last
log stderr
 logging filter all 1
 logging color 1
 logging print category-hex 0
 logging print category 1
 logging print thread-id 0
 logging timestamp 0
 logging print level 1
 logging print file basename last
!
log gsmtap __GATEWAY_IP__
 logging filter all 1
 logging color 0
 logging timestamp 0
 logging print category 1
 logging print level 1
 logging level rll notice
 logging level mm notice
 logging level rr notice
```

```
 logging level rsl notice
 logging level nm notice
 logging level pag notice
 logging level meas notice
 logging level msc notice
 logging level ho notice
 logging level hodec notice
 logging level ref notice
 logging level ctrl notice
 logging level filter notice
 logging level pcu notice
 logging level lcls notice
 logging level chan notice
 logging level ts notice
 logging level as notice
 logging level cbs notice
 logging level lcs notice
 logging level asci notice
 logging level reset notice
 logging level loop notice
 logging level lglobal notice
 logging level llapd notice
 logging level linp notice
 logging level lmux notice
 logging level lmi notice
 logging level lmib notice
 logging level lsms notice
 logging level lctrl notice
 logging level lgtp notice
 logging level lstats notice
 logging level lgsup notice
 logging level loap notice
 logging level lss7 notice
 logging level lsccp notice
 logging level lsua notice
 logging level lm3ua notice
 logging level lmgcp notice
 logging level ljibuf notice
 logging level lrspro notice
 logging level lns notice
 logging level lbssgp notice
 logging level lnsdata notice
 logging level lnssignal notice
 logging level liuup notice
 logging level lpfcp notice
 logging level lcsn1 notice
 logging level lio notice
 logging level ltcap notice
!
stats interval 5
!
line vty
 no login
!
e1_input
 e1_line 0 driver ipa
 e1_line 0 port 0
!
cs7 instance 0
 network-indicator international
 point-code __PC_BSC__

 asp asp-to-stp 2905 2907 m3ua
  remote-ip 127.0.0.1
  local-ip  127.0.0.1
  role asp
```

```
  sctp-role client
  no shutdown

 as as-bsc m3ua
  asp asp-to-stp
  routing-key __RCTX_BSC__ __PC_BSC__
  traffic-mode override

 sccp-address addr-msc
  routing-indicator PC
  point-code __PC_MSC__
  subsystem-number 254

network
 network country code __MCC__
 mobile network code __MNC__
 encryption __ENCRYPTION__
 neci 1
 paging any use tch 0
 handover 0
 handover algorithm 1
 handover1 window rxlev averaging 10
 handover1 window rxqual averaging 1
 handover1 window rxlev neighbor averaging 10
 handover1 power budget interval 6
 handover1 power budget hysteresis 3
 handover1 maximum distance 9999
 timer t3109 __T3109__
 mgw 0
  local-port 2727
  remote-ip 127.0.0.1
  remote-port 2427
 bts 0
  type osmo-bts
  band GSM900
  cell_identity __CELL_ID__
  location_area_code __LAC__
  base_station_id_code __BSIC__
  ms max power __MS_MAX_POWER__
  cell reselection hysteresis __CELL_RESEL_HYST__
  rxlev access min __RXLEV_ACCESS_MIN__
  radio-link-timeout 32
  channel allocator mode chan-req descending
  channel allocator mode assignment ascending
  channel allocator mode handover ascending
  channel allocator mode vgcs-vbs ascending
  rach tx integer 9
  rach max transmission 7
  rach max-delay 63
  rach expiry-timeout 32
  channel-description attach 1
  channel-description bs-pa-mfrms 5
  channel-description bs-ag-blks-res 1
  no nch-position
  no access-control-class-ramping
  early-classmark-sending forbidden
  early-classmark-sending-3g allowed
  ipa unit-id __IPA_UNIT_ID__ 0
  oml ipa stream-id 255 line 0
  ! [2026-10-22] marqueur NEIGHBOR_MODE_BTS0 (generate_configs.sh) : « manual » sans voisin avec --ms-magnetite.
  ! Le 614 de la BTS1 (sidecar) entrait dans la liste BA ; le faux air ne voit pas la L1 TI s'y accorder, ses
  ! fenetres recoivent la cellule, le voisin parait fort -> synchro voisine (PM lues a 0) -> RECOVERY du DSP.
  ! MAGNETITE_VOISINS=1 garde « automatic » (enquete sur le reglage du voisin, trace TSP de QEMU).
  neighbor-list mode __NEIGHBOR_MODE_BTS0__
  codec-support fr
  amr tch-f modes 0 2 5 7
```

```
amr tch-f threshold ms 13 25 37
amr tch-f hysteresis ms 4 4 4
amr tch-f threshold bts 13 25 37
amr tch-f hysteresis bts 4 4 4
amr tch-f start-mode auto
amr tch-h modes 0 2 3 5
amr tch-h threshold ms 16 24 32
amr tch-h hysteresis ms 4 4 4
amr tch-h threshold bts 16 24 32
amr tch-h hysteresis bts 4 4 4
amr tch-h start-mode auto
gprs mode egprs
gprs routing area 0
gprs network-control-order nc0
gprs power-control alpha 0
gprs cell bvci __BVCI__
gprs cell timer blocking-timer 3
gprs cell timer blocking-retries 3
gprs cell timer unblocking-retries 3
gprs cell timer reset-timer 3
gprs cell timer reset-retries 3
gprs cell timer suspend-timer 10
gprs cell timer suspend-retries 3
gprs cell timer resume-timer 10
gprs cell timer resume-retries 3
gprs cell timer capability-update-timer 10
gprs cell timer capability-update-retries 3
gprs nsei __NSEI__
gprs ns timer tns-block 3
gprs ns timer tns-block-retries 3
gprs ns timer tns-reset 3
gprs ns timer tns-reset-retries 3
gprs ns timer tns-test 30
gprs ns timer tns-alive 3
gprs ns timer tns-alive-retries 10
gprs nsvc 0 nsvci __NSVCI__
gprs nsvc 0 local udp port 23001
gprs nsvc 0 remote ip __CONTAINER_IP__
gprs nsvc 0 remote udp port 23000
gprs nsvc 1 nsvci 0
gprs nsvc 1 local udp port 0
gprs nsvc 1 remote ip 0.0.0.0
gprs nsvc 1 remote udp port 0
bs-power-control
 mode static
ms-power-control
 mode dyn-bts
 ctrl-interval 2
 step-size inc 4 red 2
 rxlev-thresh lower 32 upper 38
 rxlev-thresh-comp lower 10 12 upper 19 20
 rxqual-thresh lower 3 upper 0
 rxqual-thresh-comp lower 5 7 upper 15 18
 ci-thresh fr-efr disable
 ci-thresh fr-efr lower 13 upper 17
 ci-thresh-comp fr-efr lower 5 7 upper 15 18
 ci-thresh hr disable
 ci-thresh hr lower 16 upper 21
 ci-thresh-comp hr lower 5 7 upper 15 18
 ci-thresh amr-fr disable
 ci-thresh amr-fr lower 7 upper 11
 ci-thresh-comp amr-fr lower 5 7 upper 15 18
 ci-thresh amr-hr disable
 ci-thresh amr-hr lower 13 upper 17
 ci-thresh-comp amr-hr lower 5 7 upper 15 18
 ci-thresh sdcch disable
```

```
   ci-thresh sdcch lower 12 upper 16
   ci-thresh-comp sdcch lower 5 7 upper 15 18
   ci-thresh gprs disable
   ci-thresh gprs lower 18 upper 24
   ci-thresh-comp gprs lower 5 7 upper 15 18
  trx 0
   rf_locked 0
   arfcn __ARFCN__
   nominal power 23
   max_power_red 0
   rsl e1 tei 0
   timeslot 0
    phys_chan_config CCCH+SDCCH4
    hopping enabled 0
   timeslot 1
    ! [2026-10-21] type pose par generate_configs.sh (marqueur TS1_PCHAN_BTS0) : SDCCH8 en BTS classique (pont,
↪  pas de
    ! trx.highram), TCH/F avec le Calypso TRX (--bts / OSMO_TRX=1). [2026-10-22] Magnetite : SDCCH8 aussi (le pont
↪  annonce le SDCCH8 lu dans l IMM ASS, pont/downlink.py) ; avant : la L1 TI n annonce pas
    ! son canal dedie, le banc ne sert que TS0 ; 21/10 23:06 SDCCH8 TS1 -> « >N200 of SABM »). Avec le TRX un seul
↪  TS etait servi :
    ! le BSC preferait les SDCCH/8 du TS1 pour la LU et le mobile ne les entendait jamais (20/10).
    phys_chan_config __TS1_PCHAN_BTS0__
    hopping enabled 0
   timeslot 2
    phys_chan_config TCH/F
    hopping enabled 0
   timeslot 3
    phys_chan_config TCH/F
    hopping enabled 0
   timeslot 4
    phys_chan_config TCH/F
    hopping enabled 0
   timeslot 5
    phys_chan_config TCH/F
    hopping enabled 0
   timeslot 6
    phys_chan_config PDCH
    hopping enabled 0
   timeslot 7
    phys_chan_config PDCH
    hopping enabled 0
!
! ── Seconde BTS : le side-car fake_trx ──────────────────────────────────────
! Elle manquait ici. osmo-bts-trx-bts1.cfg s'annonce avec l'unit-id calcule par
! generate_configs.sh (6000 + operateur*10 + 2), mais aucun bloc "bts 1" ne
! l'attendait cote BSC : le BSC repondait
!     "Unable to find BTS configuration for 6012/0/0, disconnecting"
! puis la BTS se coupait ("Abis close"), ne demandait jamais POWERON, fake_trx
! n'emettait donc aucune horloge, et le module de demarrage concluait a une
! "course No clock" - un diagnostic qui designait le mauvais coupable. La
! sequence s'arretant la, la BTS PRINCIPALE ne demarrait pas non plus : plus
! aucune radio, pour un identifiant qui ne correspondait pas.
!
! Les valeurs viennent du meme plan que le reste : unit-id et ARFCN decales
! pour ne croiser ni la BTS principale, ni les operateurs voisins.
!
! PLACEMENT - NE PAS DEPLACER CE BLOC EN FIN DE FICHIER.
! [2026-08-27] Il y avait ete ajoute, apres "msc 0" / "bsc" / "end". Le parseur
! VTY est un automate a noeuds : "msc 0" ferme le noeud "network", et "end"
! ramene a la racine. Un " bts 1" indente d'un cran a cet endroit ne designe
! plus rien, d'ou :
!     Error occurred during reading the below line:
!      bts 1
!     DNM FATAL Failed to parse the config file (Invalid argument)
! osmo-bsc sortait en code 1, systemd le relancait en boucle, et la sequence de
```

```
! demarrage rapportait "VTY OsmoBSC (4242) : toujours pas pret apres 40s" -
! encore un symptome qui designait le mauvais coupable. "bts 1" doit rester
! DANS le noeud "network", a la suite de "bts 0" et AVANT "msc 0".
 bts 1
  type osmo-bts
  band __BAND__
  cell_identity __IPA_UNIT_ID_BTS1__
  location_area_code __LAC__
  base_station_id_code __BSIC_BTS1__
  ms max power __MS_MAX_POWER__
  cell reselection hysteresis __CELL_RESEL_HYST__
  rxlev access min __RXLEV_ACCESS_MIN__
  radio-link-timeout 32
  channel allocator mode chan-req descending
  channel allocator mode assignment ascending
  channel allocator mode handover ascending
  rach tx integer 9
  rach max transmission 7
  ipa unit-id __IPA_UNIT_ID_BTS1__ 0
  oml ipa stream-id 255 line 0
  neighbor-list mode automatic
  codec-support fr
  gprs mode egprs
  gprs routing area 0
  gprs network-control-order nc0
  gprs power-control alpha 0
  gprs cell bvci 13
  gprs cell timer blocking-timer 3
  gprs cell timer blocking-retries 3
  gprs cell timer unblocking-retries 3
  gprs cell timer reset-timer 3
  gprs cell timer reset-retries 3
  gprs cell timer suspend-timer 10
  gprs cell timer suspend-retries 3
  gprs cell timer resume-timer 10
  gprs cell timer resume-retries 3
  gprs cell timer capability-update-timer 10
  gprs cell timer capability-update-retries 3
  gprs nsei 11
  gprs ns timer tns-block 3
  gprs ns timer tns-block-retries 3
  gprs ns timer tns-reset 3
  gprs ns timer tns-reset-retries 3
  gprs ns timer tns-test 30
  gprs ns timer tns-alive 3
  gprs ns timer tns-alive-retries 10
  gprs nsvc 0 nsvci 11
  gprs nsvc 0 local udp port 23002
  gprs nsvc 0 remote ip 192.168.2.10
  gprs nsvc 0 remote udp port 23000
  gprs nsvc 1 nsvci 0
  gprs nsvc 1 local udp port 0
  gprs nsvc 1 remote ip 0.0.0.0
  gprs nsvc 1 remote udp port 0
  trx 0
   rf_locked 0
   arfcn __ARFCN_BTS1__
   nominal power 23
   max_power_red 0
   rsl e1 tei 0
   timeslot 0
    phys_chan_config CCCH+SDCCH4
    hopping enabled 0
   timeslot 1
    phys_chan_config SDCCH8
    hopping enabled 0
```

```
   timeslot 2
    phys_chan_config TCH/F
    hopping enabled 0
   timeslot 3
    phys_chan_config TCH/F
    hopping enabled 0
   timeslot 4
    phys_chan_config TCH/F
    hopping enabled 0
   timeslot 5
    phys_chan_config PDCH
    hopping enabled 0
   timeslot 6
    phys_chan_config PDCH
    hopping enabled 0
   timeslot 7
    phys_chan_config PDCH
    hopping enabled 0
msc 0
 codec-list fr1 hr1 fr2 fr3 hr3
 allow-emergency deny
 amr-config ⟨1⟩_⟨2⟩k allowed  ×8
    ⟨⟩ = (12,2) (10,2) (7,95) (7,40) (6,70) (5,90) (5,15) (4,75)
 amr-payload octet-aligned
 msc-addr addr-msc
 asp-protocol m3ua
 lcls-mode bts-loop
 lcls-codec-mismatch allowed
bsc
 mid-call-timeout 0
end
```

3.53 /opt/GSM/osmo-operator/configs/osmo-bts-trx-bts1.cfg

1326 octets, 64 lignes → 64 lignes

```
!
! OsmoBTS BTS#1 (side-car faketrx) — généré par start-native.sh (mode faketrx-qemu)
! NE PAS confondre avec /etc/osmocom/osmo-bts-trx.cfg (= BTS#0 QEMU, INTOUCHÉ).
!
log stderr
 logging color 1
 logging print category-hex 0
 logging print category 1
 logging timestamp 0
 logging print file basename last
 logging print level 1
!
log gsmtap __GATEWAY_IP__
 logging filter all 1
 logging color 0
 logging timestamp 0
 logging print category 1
 logging print level 1
 logging level rsl info
 logging level oml info
 logging level rr notice
 logging level trx notice
 logging level abis debug
!
vty telnet-port 4250
!
ctrl
 bind 127.0.0.2 4239
```

```
!
line vty
 no login
!
e1_input
 e1_line 0 driver ipa
 e1_line 0 port 0
 no e1_line 0 keepalive
!
phy 0
 instance 0
  osmotrx rx-gain 40
  osmotrx tx-attenuation 50
 osmotrx ip local 127.0.0.1
 osmotrx ip remote 127.0.0.1
 osmotrx base-port local 5820
 osmotrx base-port remote 5720
!
bts 0
 band DCS1800
 ipa unit-id __IPA_UNIT_ID_BTS1__ 0
 oml remote-ip 127.0.0.1
 pcu-socket /tmp/pcu_bts2
 rtp jitter-buffer 100
 paging queue-size 200
 paging lifetime 0
 min-qual-rach 50
 min-qual-norm -5
 gsmtap-sapi ccch
 gsmtap-sapi pdtch
 trx 0
  power-ramp max-initial 23000 mdBm
  power-ramp step-size 2000 mdB
  power-ramp step-interval 1
  ms-power-control osmo
  phy 0 instance 0
```

3.54 /opt/GSM/osmo-operator/configs/osmo-bts-trx.cfg

1364 octets, 57 lignes → 57 lignes

```
!
! OsmoBTS (192.168.1.15-bc49-dirty) configuration saved from vty
!!
!
log file /var/log/osmocom/osmo-bts-trx.log
 logging color 1
 logging timestamp 1
 logging print category-hex 0
 logging print category 1
 logging print file basename last
 logging print level 1
log stderr
 logging color 1
 logging print category-hex 0
 logging print category 1
 logging timestamp 0
 logging print file basename last
 logging print level 1
!
line vty
 no login
!
e1_input
```

```
 e1_line 0 driver ipa
 e1_line 0 port 0
 no e1_line 0 keepalive
!
phy 0
 instance 0
  osmotrx rx-gain 40
  osmotrx tx-attenuation 50
 osmotrx ip local 127.0.0.1
 osmotrx ip remote 127.0.0.1
 no osmotrx timing-advance-loop
 ! [2026-10-20] TRX Calypso (transceiver, --osmo-trx) : l horloge IND CLOCK vaut le fn du firmware ;
 ! avec les defauts (fn-advance 2, rts-advance 3) les bursts arrivaient 1 a 4 trames avant leur
 ! emission et le firmware les jetait (STALE BURST). Osmo-trx reel : 20 et 5.
 osmotrx fn-advance 20
 osmotrx rts-advance 5

bts 0
 band GSM900
 ipa unit-id __IPA_UNIT_ID__ 0
 oml remote-ip 127.0.0.1
 rtp jitter-buffer 100
 paging queue-size 200
 paging lifetime 0
 min-qual-rach 50
 min-qual-norm -5
 gsmtap-sapi ccch
 gsmtap-sapi pdtch
 trx 0
  power-ramp max-initial 23000 mdBm
  power-ramp step-size 2000 mdB
  power-ramp step-interval 1
  ms-power-control osmo
  phy 0 instance 0
```

3.55 /opt/GSM/osmo-operator/configs/osmo-bts-virtual.cfg

3270 octets, 135 lignes → 135 lignes

```
!
! osmo-bts-virtual.cfg — BTS virtuelle (mode virtphy)
! Opérateur __OPERATOR_ID__
!
! Contrairement à osmo-bts-trx.cfg (qui utilise TRXD/fake_trx),
! osmo-bts-virtual communique avec virtphy via multicast UDP.
! Pas besoin de trxcon ni de fake_trx.
!
! Chaîne : mobile ↔ virtphy ↔ (multicast) ↔ osmo-bts-virtual ↔ BSC
!
log file /var/log/osmocom/osmo-bts-virtual.log
 logging color 1
 logging timestamp 1
 logging print category-hex 0
 logging print category 1
 logging print file basename last
 logging print level 1
 logging level rsl info
 logging level oml info
 logging level rll notice
 logging level rr notice
 logging level meas notice
 logging level pag info
 logging level l1c info
 logging level l1p info
```

```
 logging level dsp error
 logging level pcu notice
 logging level ho debug
 logging level trx notice
 logging level loop notice
 logging level abis debug
 logging level rtp notice
 logging level lglobal notice
 logging level llapd notice
 logging level linp notice
 logging level lmux notice
 logging level lmi notice
 logging level lmib notice
 logging level lsms notice
 logging level lctrl notice
 logging level lgtp notice
 logging level lstats error
log stderr
 logging color 1
 logging print category-hex 0
 logging print category 1
 logging timestamp 0
 logging print file basename last
 logging print level 1
 logging level rsl info
 logging level oml info
 logging level rll notice
 logging level rr notice
 logging level meas notice
 logging level pag info
 logging level l1c info
 logging level l1p info
 logging level dsp error
 logging level pcu notice
 logging level ho debug
 logging level trx notice
 logging level loop notice
 logging level abis debug
 logging level rtp notice
 logging level lglobal notice
 logging level llapd notice
 logging level linp notice
 logging level lmux notice
 logging level lmi notice
 logging level lmib notice
 logging level lsms notice
 logging level lctrl notice
 logging level lgtp notice
 logging level lstats error
!
log gsmtap __GATEWAY_IP__
 logging filter all 1
 logging color 0
 logging timestamp 0
 logging print category 1
 logging print level 1
 logging level rsl info
 logging level oml info
 logging level rll notice
 logging level rr notice
 logging level meas notice
 logging level pag info
 logging level l1c info
 logging level l1p info
 logging level dsp error
 logging level pcu notice
```

```
 logging level ho debug
 logging level trx notice
 logging level loop notice
 logging level abis debug
 logging level rtp notice
 logging level lglobal notice
 logging level llapd notice
 logging level linp notice
 logging level lmux notice
 logging level lmi notice
 logging level lmib notice
 logging level lsms notice
 logging level lctrl notice
 logging level lgtp notice
 logging level lstats error
!
line vty
 no login
!
e1_input
 e1_line 0 driver ipa
 e1_line 0 port 0
 no e1_line 0 keepalive
!
phy 0
 instance 0
!
bts 0
 band DCS1800
 ipa unit-id __IPA_UNIT_ID__ 0
 oml remote-ip 127.0.0.1
 rtp jitter-buffer 100
 paging queue-size 200
 paging lifetime 0
 min-qual-rach 50
 min-qual-norm -5
 gsmtap-sapi ccch
 gsmtap-sapi pdtch
 trx 0
  power-ramp max-initial 23000 mdBm
  power-ramp step-size 2000 mdB
  power-ramp step-interval 1
  ms-power-control osmo
  phy 0 instance 0
```

3.56 /opt/GSM/osmo-operator/configs/osmo-bts.cfg

1919 octets, 89 lignes → 89 lignes

```
!
! OsmoBTS (192.168.1.15-bc49-dirty) configuration saved from vty
!!
!
log file /var/log/osmocom/osmo-bts.log
 logging color 1
 logging timestamp 1
 logging print category-hex 0
 logging print category 1
 logging print file basename last
 logging print level 1
log stderr
 logging color 1
 logging print category-hex 0
 logging print category 1
```

```
 logging timestamp 0
 logging print file basename last
 logging print level 1
!
log gsmtap __GATEWAY_IP__
 logging filter all 1
 logging color 0
 logging timestamp 0
 logging print category 1
 logging print level 1
 logging level rsl info
 logging level oml info
 logging level rll notice
 logging level rr notice
 logging level meas notice
 logging level pag info
 logging level l1c info
 logging level l1p info
 logging level dsp error
 logging level pcu notice
 logging level ho debug
 logging level trx notice
 logging level loop notice
 logging level abis debug
 logging level rtp notice
 logging level lglobal notice
 logging level llapd notice
 logging level linp notice
 logging level lmux notice
 logging level lmi notice
 logging level lmib notice
 logging level lsms notice
 logging level lctrl notice
 logging level lgtp notice
 logging level lstats error
!
line vty
 no login
!
e1_input
 e1_line 0 driver ipa
 e1_line 0 port 0
 no e1_line 0 keepalive
!
phy 0
 instance 0
  osmotrx rx-gain 40
  osmotrx tx-attenuation 50
 osmotrx ip local 127.0.0.1
 osmotrx ip remote 127.0.0.1
 no osmotrx timing-advance-loop

bts 0
 band GSM900
 ipa unit-id __IPA_UNIT_ID__ 0
 oml remote-ip 127.0.0.1
 rtp jitter-buffer 100
 paging queue-size 200
 paging lifetime 0
 min-qual-rach 50
 min-qual-norm -5
 gsmtap-sapi ccch
 gsmtap-sapi bcch
 gsmtap-sapi rach
 gsmtap-sapi agch
 gsmtap-sapi pch
```

```
 gsmtap-sapi sdcch
 gsmtap-sapi sacch
 trx 0
  power-ramp max-initial 23000 mdBm
  power-ramp step-size 2000 mdB
  power-ramp step-interval 1
  ms-power-control osmo
  phy 0 instance 0
```

3.57 /opt/GSM/osmo-operator/configs/osmo-ggsn.cfg

1921 octets, 81 lignes → 81 lignes

```
!
! OpenGGSN configuration
!
log file /var/log/osmocom/osmo-ggsn.log
 logging color 1
 logging timestamp 1
 logging print category-hex 0
 logging print category 1
 logging print file basename last
 logging print level 1
 logging level ip info
 logging level tun info
 logging level ggsn info
 logging level sgsn notice
 logging level icmp6 notice
 logging level lglobal notice
 logging level llapd notice
 logging level linp notice
 logging level lmux notice
 logging level lmi notice
 logging level lmib notice
 logging level lsms notice
 logging level lctrl notice
 logging level lgtp info
 logging level lstats notice
 logging level lgsup notice
 logging level loap notice
 logging level lss7 notice
 logging level lsccp notice
 logging level lsua notice
 logging level lm3ua notice
 logging level lmgcp notice
log stderr
 logging color 1
 logging print category-hex 0
 logging print category 1
 logging timestamp 0
 logging print file basename last
 logging print level 1
 logging level ip info
 logging level tun info
 logging level ggsn info
 logging level sgsn notice
 logging level icmp6 notice
 logging level lglobal notice
 logging level llapd notice
 logging level linp notice
 logging level lmux notice
 logging level lmi notice
 logging level lmib notice
 logging level lsms notice
```

```
 logging level lctrl notice
 logging level lgtp info
 logging level lstats notice
 logging level lgsup notice
 logging level loap notice
 logging level lss7 notice
 logging level lsccp notice
 logging level lsua notice
 logging level lm3ua notice
 logging level lmgcp notice
!
stats interval 5
!
line vty
 no login
!
ggsn ggsn0
 gtp state-dir /var/lib/osmocom/osmo-ggsn
 gtp bind-ip __CONTAINER_IP__
 apn __APN__
  gtpu-mode tun
  tun-device apn0
  type-support v4
  ip prefix dynamic 176.16.32.0/24
  ip dns 0 192.168.1.254
  ip dns 1 1.1.1.1
  ip ifconfig 176.16.32.0/24
  no shutdown
 default-apn __APN__
 no shutdown ggsn
```

3.58 /opt/GSM/osmo-operator/configs/osmo-hlr.cfg

3890 octets, 145 lignes → 145 lignes

```
!
! OsmoHLR configuration — Opérateur __OPERATOR_ID__
! SMS-over-GSUP activé via gsup-smsc-proto
!
log file /var/log/osmocom/osmo-hlr.log
 logging color 1
 logging timestamp 1
 logging filter all 1
 logging print category-hex 0
 logging print category 1
 logging print extended-timestamp 1
 logging print level 1
 logging print file basename
 logging level main debug
 logging level db debug
 logging level auc debug
 logging level ss debug
 logging level lglobal debug
 logging level llapd debug
 logging level linp debug
 logging level lmux debug
 logging level lmi debug
 logging level lmib debug
 logging level lsms debug
 logging level lctrl debug
 logging level lgtp debug
 logging level lstats debug
 logging level lgsup debug
 logging level loap debug
```

```
 logging level lss7 debug
 logging level lsccp debug
 logging level lsua debug
 logging level lm3ua debug
 logging level lmgcp debug
 logging level ljibuf debug
 logging level lrspro debug
 logging filter all 0
 logging print category-hex 1
 logging print category 0
 logging print file 1
 logging level main notice
 logging level db notice
 logging level auc notice
 logging level ss notice
 logging level lglobal notice
 logging level llapd notice
 logging level linp notice
 logging level lmux notice
 logging level lmi notice
 logging level lmib notice
 logging level lsms notice
 logging level lctrl notice
 logging level lgtp notice
 logging level lstats notice
 logging level lgsup notice
 logging level loap notice
 logging level lss7 notice
 logging level lsccp notice
 logging level lsua notice
 logging level lm3ua notice
 logging level lmgcp notice
 logging level ljibuf notice
 logging level lrspro notice
log stderr
 logging filter all 1
 logging color 1
 logging print category-hex 0
 logging print category 1
 logging print extended-timestamp 1
 logging print level 1
 logging print file basename
 logging level main debug
 logging level db debug
 logging level auc debug
 logging level ss debug
 logging level lglobal debug
 logging level llapd debug
 logging level linp debug
 logging level lmux debug
 logging level lmi debug
 logging level lmib debug
 logging level lsms debug
 logging level lctrl debug
 logging level lgtp debug
 logging level lstats debug
 logging level lgsup debug
 logging level loap debug
 logging level lss7 debug
 logging level lsccp debug
 logging level lsua debug
 logging level lm3ua debug
 logging level lmgcp debug
 logging level ljibuf debug
 logging level lrspro debug
 logging filter all 0
```

```
 logging color 1
 logging print category-hex 1
 logging print category 0
 logging timestamp 0
 logging print file 1
 logging level main notice
 logging level db notice
 logging level auc notice
 logging level ss notice
 logging level lglobal notice
 logging level llapd notice
 logging level linp notice
 logging level lmux notice
 logging level lmi notice
 logging level lmib notice
 logging level lsms notice
 logging level lctrl notice
 logging level lgtp notice
 logging level lstats notice
 logging level lgsup notice
 logging level loap notice
 logging level lss7 notice
 logging level lsccp notice
 logging level lsua notice
 logging level lm3ua notice
 logging level lmgcp notice
 logging level ljibuf notice
 logging level lrspro notice
!
stats interval 5
!
line vty
 no login
!
hlr
 subscriber-create-on-demand 5 cs+ps
 store-imei
 database /var/lib/osmocom/hlr.db
 gsup
  bind ip 127.0.0.2
  ipa-name HLR-OP__OPERATOR_ID__
 ussd route prefix *#100# internal own-msisdn
 ussd route prefix *#101# internal own-imsi
 !
 ! ── SMS-over-GSUP : proto-SMSC connecté via GSUP ──────────────────────────
 ! Le proto-smsc-daemon se connecte au HLR avec l'IPA name SMSC-OP<N>
 ! SC-address = numéro fictif unique par opérateur (réservé NANP)
 !
 smsc entity SMSC-OP__OPERATOR_ID__
 smsc default-route SMSC-OP__OPERATOR_ID__
```

3.59 /opt/GSM/osmo-operator/configs/osmo-mgw.cfg

2385 octets, 92 lignes → 92 lignes

```
!
! OsmoMGW (1.7.0) configuration saved from vty
!!
!
log file /var/log/osmocom/osmo-mgw.log
 logging color 1
 logging timestamp 1
 logging filter all 1
 logging print category-hex 1
```

```
 logging print category 0
 logging print file 1
 logging level rtp notice
 logging level lglobal notice
 logging level llapd notice
 logging level linp notice
 logging level lmux notice
 logging level lmi notice
 logging level lmib notice
 logging level lsms notice
 logging level lctrl notice
 logging level lgtp notice
 logging level lstats notice
 logging level lgsup notice
 logging level loap notice
 logging level lss7 notice
 logging level lsccp notice
 logging level lsua notice
 logging level lm3ua notice
 logging level lmgcp notice
 logging level ljibuf notice
 logging level lrspro notice
log stderr
 logging filter all 1
 logging color 0
 logging print category-hex 1
 logging print category 0
 logging timestamp 0
 logging print file 1
 logging level rtp notice
 logging level lglobal notice
 logging level llapd notice
 logging level linp notice
 logging level lmux notice
 logging level lmi notice
 logging level lmib notice
 logging level lsms notice
 logging level lctrl notice
 logging level lgtp notice
 logging level lstats notice
 logging level lgsup notice
 logging level loap notice
 logging level lss7 notice
 logging level lsccp notice
 logging level lsua notice
 logging level lm3ua notice
 logging level lmgcp notice
 logging level ljibuf notice
 logging level lrspro notice
!
stats interval 5
!
line vty
 no login
!
!
mgcp
  domain mgw
  bind ip 127.0.0.1
  bind port 2427
  rtp port-range 4002 16001
  rtp bind-ip 127.0.0.1
  rtp ip-probing
!
! [2026-08-12] Keep-alive RTP DESACTIVE. osmo-mgw emettait 1 paquet dummy d'un
! octet (0x23 = MGCP_DUMMY_LOAD) par connexion, pour percer un NAT. Asterisk 18
```

```
! classe tout premier octet dans [20,63] comme un enregistrement DTLS, d'ou un
! "Received SSL traffic on RTP instance without an SSL session"
! (res_rtp_asterisk.c:3133) a chaque appel. Inutile ici : rtp bind-ip 127.0.0.1,
! le pair (Asterisk) est en loopback, il n'y a aucun NAT a percer.
  no rtp keep-alive
  rtcp-omit
  rtp-patch ssrc
  rtp-patch timestamp
  no rtp-patch rfc5993hr
  sdp audio-payload number 98
  sdp audio-payload name GSM
  sdp audio-payload send-ptime
  sdp audio-payload send-name
  loop 0
  number endpoints 31
  allow-transcoding
  osmux off
```

3.60 /opt/GSM/osmo-operator/configs/osmo-msc.cfg

5762 octets, 212 lignes → 212 lignes

```
!
! osmo-msc.cfg — MSC opérateur __OPERATOR_ID__ (MCC __MCC__ / MNC __MNC__)
!
log file /var/log/osmocom/osmo-msc.log
# [2026-09-08] cc, mm, mncc et lsms en info dans le fichier : c est la
# seule facon de voir DISCONNECT / RELEASE et le sort d un RP-DATA quand
# un mobile reste bloque en appel (« MO SMS without prior CM Service
# Request », cause 96). Les « notice » plus bas ecrasaient les « info »
# des premieres lignes : meme cible, la derniere ligne gagne.
 logging level mncc info
 logging level cc info
 logging color 1
 logging timestamp 1
 logging filter all 1
 logging print category-hex 1
 logging print category 0
 logging print thread-id 0
 logging print file 1
 logging level rll notice
 logging level cc info
 logging level bcc notice
 logging level gcc notice
 logging level mm info
 logging level rr notice
 logging level mncc info
 logging level pag notice
 logging level msc notice
 logging level mgcp notice
 logging level ho notice
 logging level db notice
 logging level ref notice
 logging level ctrl notice
 logging level smpp notice
 logging level ranap notice
 logging level vlr notice
 logging level iucs notice
 logging level bssap notice
 logging level sgs notice
 logging level ss notice
 logging level asci notice
 logging level lglobal notice
 logging level llapd notice
```

```
 logging level linp notice
 logging level lmux notice
 logging level lmi notice
 logging level lmib notice
 logging level lsms info
 logging level lctrl notice
 logging level lgtp notice
 logging level lstats notice
 logging level lgsup notice
 logging level loap notice
 logging level lss7 info
 logging level lsccp notice
 logging level lsua notice
 logging level lm3ua info
 logging level lmgcp notice
 logging level ljibuf notice
 logging level lrspro notice
 logging level lns notice
 logging level lbssgp notice
 logging level lnsdata notice
 logging level lnssignal notice
 logging level liuup notice
 logging level lpfcp notice
 logging level lcsn1 notice
 logging level lio notice
 logging level ltcap notice
log stderr
 logging filter all 1
 logging color 1
 logging print category-hex 1
 logging print category 0
 logging print thread-id 0
 logging timestamp 0
 logging print file 1
 logging level rll notice
 logging level cc notice
 logging level bcc notice
 logging level gcc notice
 logging level mm notice
 logging level rr notice
 logging level mncc notice
 logging level pag notice
 logging level msc notice
 logging level mgcp notice
 logging level ho notice
 logging level db notice
 logging level ref notice
 logging level ctrl notice
 logging level smpp notice
 logging level ranap notice
 logging level vlr notice
 logging level iucs notice
 logging level bssap notice
 logging level sgs notice
 logging level ss notice
 logging level asci notice
 logging level lglobal notice
 logging level llapd notice
 logging level linp notice
 logging level lmux notice
 logging level lmi notice
 logging level lmib notice
 logging level lsms notice
 logging level lctrl notice
 logging level lgtp notice
 logging level lstats notice
```

```
 logging level lgsup notice
 logging level loap notice
 logging level lss7 info
 logging level lsccp notice
 logging level lsua notice
 logging level lm3ua info
 logging level lmgcp notice
 logging level ljibuf notice
 logging level lrspro notice
 logging level lns notice
 logging level lbssgp notice
 logging level lnsdata notice
 logging level lnssignal notice
 logging level liuup notice
 logging level lpfcp notice
 logging level lcsn1 notice
 logging level lio notice
 logging level ltcap notice
!
stats interval 5
!
line vty
 no login
!
cs7 instance 0
 network-indicator international
 point-code __PC_MSC__
!
 ! MSC → STP : même container → 127.0.0.1
 asp asp-to-stp 2905 2906 m3ua
  remote-ip 127.0.0.2
  local-ip  127.0.0.1
  role asp
  sctp-role client
  no shutdown
!
 as as-msc m3ua
  asp asp-to-stp
  routing-key __RCTX_MSC__ __PC_MSC__
  traffic-mode override
!
 sccp-address addr-bsc
  routing-indicator PC
  point-code __PC_BSC__
  subsystem-number 254
!
network
 network country code __MCC__
 mobile network code __MNC__
 short name __OP_NAME__
 long name __OP_NAME__
 encryption __ENCRYPTION__
 authentication optional
 rrlp mode none
 mm info 0
 mgw 0
  remote-ip 127.0.0.1
msc
 mncc external /tmp/msc_mncc
 mncc guard-timeout 180
 ncss guard-timeout 30
 assign-tmsi
 ! [2026-09-23] banc DSP plus lent que le temps reel : le LU complet (auth, chiffrement,
 ! TMSI REALLOC COMPLETE) depasse les 5 s par defaut de X1 -> LU REJECT en RELEASING.
 timer geran X1 30
 lcls-permitted
```

```
 check-imei-rqd early
 ! SMS-over-GSUP : redirige les SM-RP vers le HLR → proto-smsc-daemon
 sms-over-gsup
 auth-tuple-max-reuse-count 3
 auth-tuple-reuse-on-error 1

mncc-int
 default-codec tch-f fr
 default-codec tch-h hr
smpp
 local-tcp-port 2775
 policy closed
 # [2026-09-06] smpp-first : le MSC propose le SMS a l ESME AVANT la livraison
 # interne. C est ce qui permet au telephone du banc (la VM postmarketOS) de recevoir les
 # SMS de l abonne 100101 - il EST cet abonne, au meme titre que le mobile
 # osmocom-bb. tools/osmo-phonesim-banc.py se lie ici comme ESME et remet
 # ensuite une copie au mobile par le VTY, pour que les deux sonnent.
 smpp-first
 esme msc_tester
  password osmocom1
  route prefix national isdn 33
  # Le numero du telephone Android, dans les quatre formes de numerotation
  # qu un SMS peut porter selon son emetteur.
  route prefix unknown unknown 100101
  route prefix unknown isdn 100101
  route prefix national isdn 100101
  route prefix international isdn 100101
hlr
 remote-ip __HLR_IP__
 remote-port 4222
 ipa-name VLR-SDR-OP__OPERATOR_ID__
sgs
 local-port 29118
 local-ip 127.0.0.1
 vlr-name vlr.msc001.mnc001.mcc001.3gppnetwork.org
asci
 disable
 gcr
```

3.61 /opt/GSM/osmo-operator/configs/osmo-pcu.cfg

370 octets, 20 lignes → 20 lignes

```
log file /var/log/osmocom/osmo-pcu.log
 logging color 1
 logging timestamp 1
 logging filter all 1
 logging print category 1
 logging print level 1
log gsmtap __GATEWAY_IP__
 logging filter all 1
 logging color 0
 logging timestamp 0
 logging print category 1
 logging print level 1
!
pcu
 flow-control-interval 10
 cs 1
 cs max 1 1
 alloc-algorithm a
 alpha 0
 gamma 0
```

3.62 /opt/GSM/osmo-operator/configs/osmo-sgsn.cfg

893 octets, 45 lignes → 45 lignes

```
!
! Osmocom SGSN configuration
!
log file /var/log/osmocom/osmo-sgsn.log
 logging color 1
 logging timestamp 1
 logging print category-hex 0
 logging print category 1
 logging print file basename last
 logging print level 1
log stderr
 logging color 1
 logging print category-hex 0
 logging print category 1
 logging timestamp 0
 logging print file basename last
 logging print level 1

line vty
 no login
!
sgsn
 gtp state-dir /var/lib/osmocom/osmo-sgsn
 gtp local-ip 127.0.0.1
 ggsn 0 remote-ip __CONTAINER_IP__
 ggsn 0 gtp-version 1
 ggsn 0 echo-interval 60
 authentication optional
 auth-policy remote
 gsup remote-ip __HLR_IP__
 gsup remote-port 4222
!
ns
 timer tns-block 3
 timer tns-block-retries 3
 timer tns-reset 3
 timer tns-reset-retries 3
 timer tns-test 30
 timer tns-alive 3
 timer tns-alive-retries 10
 bind udp local
  listen __CONTAINER_IP__ 23000
  accept-ipaccess
!
bssgp
```

3.63 /opt/GSM/osmo-operator/configs/osmo-sip-connector.cfg

1839 octets, 37 lignes → 37 lignes

```
log file /var/log/osmocom/osmo-sip-connector.log
 logging level sip info
 logging level mncc info
 logging color 1
 logging timestamp 1
 logging filter all 1
 logging print category 1
 logging print level 1
!
```

```
mncc
  socket-path /tmp/msc_mncc
sip
  local 127.0.0.1 5061
  ! ── LA PATTE SIP EST LOCALE ──────────────────────────────────────────
  ! Elle portait le placeholder de dorsale INTER-OPERATEURS (celui d'osmo-stp.cfg)
  ! -- op_backbone_ip, soit 172.20.0.11 pour l'operateur 1. C'est la bonne valeur
  ! pour le local-ip de l'ASP ; c'est la mauvaise ici, et pour deux raisons
  ! independantes :
  !
  !   1. Sur l'ISO/VM cette adresse N'EXISTE PAS - build-iso.sh la retire meme
  !      volontairement ("172.20.0.11 est RETIREE : c'est l'adresse du PREMIER
  !      CONTENEUR operateur"). sofia-sip, lie en local sur 127.0.0.1, ne peut pas
  !      y emettre :
  !          DSIP tport_vsend(...): Invalid argument with (s=7 */172.20.0.11:5060)
  !          DSIP nta: INVITE (...): Invalid argument (22)
  !          DSIP INVITE got status(503), releasing leg
  !      -> TOUS les appels meurent en 503, quel que soit l'etat de la radio.
  !
  !   2. Meme joignable, Asterisk refuserait : pjsip.conf n'identifie cette patte
  !      que sur la boucle locale - [gsm_msc-identify] match=127.0.0.1, et
  !      [gsm_msc] contact=sip:127.0.0.1:5061. Les deux sens sont loopback.
  !
  ! 127.0.0.1 est correct dans LES DEUX topologies : en conteneur comme sur l'ISO,
  ! osmo-sip-connector et Asterisk partagent le meme espace reseau, et Asterisk
  ! ecoute sur 0.0.0.0:5060 (pjsip.conf [transport-udp] bind=0.0.0.0).
  ! Surchargeable par SIP_REMOTE_IP_OVERRIDE si la patte devait un jour sortir.
  remote __SIP_REMOTE_IP__ 5060
```

3.64 /opt/GSM/osmo-operator/configs/osmo-stp.cfg

2714 octets, 79 lignes → 79 lignes

```
!
! osmo-stp.cfg — Signal Transfer Point opérateur __OPERATOR_ID__
!
! Architecture :
!   [BSC __PC_BSC__] ─►┐
!                      ├─ STP __PC_STP__ @ 127.0.0.1:2905 (local)
!   [MSC __PC_MSC__] ─►┘        │
!                               │ asp client → inter-STP __INTER_STP_IP__:2908
!                               ▼
!                        [Inter-STP 0.0.0]
!
! BSC/MSC se connectent au STP via 127.0.0.1 (même container).
! Le STP écoute sur 0.0.0.0:2905 pour accepter connexions locales ET
! éviter les race conditions Docker (l'IP réseau privé n'existe pas
! encore au moment du démarrage des services).
!
log file /var/log/osmocom/osmo-stp.log
 logging color 1
 logging timestamp 1
 logging filter all 1
 logging print category-hex 0
 logging print category 1
 logging print extended-timestamp 1
 logging print level 1
 logging print file basename
 logging level lss7 info
 logging level lsccp info
 logging level lm3ua info
 logging level linp notice
log stderr
 logging filter all 1
```

```
 logging color 1
 logging print category-hex 0
 logging print category 1
 logging print extended-timestamp 1
 logging print level 1
 logging print file basename
 logging level lss7 info
 logging level lsccp info
 logging level lm3ua info
 logging level linp notice
!
stats interval 5
!
line vty
 no login
!
cs7 instance 0
 network-indicator international
 point-code __PC_STP__
!
 ! ── Listener local ──────────────────────────────────────────────────
 ! 0.0.0.0 : accepte les connexions sur toutes les interfaces
 ! → fonctionne immédiatement, avant même que Docker attache le réseau privé
 ! BSC/MSC se connectent via 127.0.0.1
 xua rkm routing-key-allocation dynamic-permitted
 listen m3ua 2905
  accept-asp-connections dynamic-permitted
  local-ip 127.0.0.1
  local-ip 127.0.0.2
!
 ! ── Lien client vers l'inter-STP central ────────────────────────────
 asp asp-to-inter 2908 2910 m3ua
  remote-ip __INTER_STP_IP__
  local-ip __INTER_LOCAL_IP__
  role asp
  sctp-role client
  __INTER_STP_SHUTDOWN__
!
 as as-inter m3ua
  asp asp-to-inter
  routing-key __RCTX_INTER__ __PC_STP__
  traffic-mode override
!
 ! ── Routage ─────────────────────────────────────────────────────────
 ! Routes locales (BSC/MSC) : créées dynamiquement par accept-asp.
 ! Catch-all : tout le reste → inter-STP.
 route-table system
  update route 0.0.0 0.0.0 linkset as-inter
```

3.65 /opt/GSM/osmo-operator/configs/osmo-trx.cfg

481 octets, 27 lignes → 27 lignes

```
log file /var/log/osmocom/osmo-trx.log
 logging color 1
 logging timestamp 1
 logging filter all 1
 logging print category 1
 logging print file basename
 logging level set-all info
log stderr
 logging filter all 1
 logging color 1
 logging print category 1
```

```
 logging timestamp 1
 logging print file basename
 logging level set-all info
!
line vty
 no login
!
trx
 bind-ip 127.0.0.1
 remote-ip 127.0.0.1
 base-port 5700
 egprs enable
 tx-sps 4
 rx-sps 4
 clock-ref internal
 chan 0
```

3.66 /opt/GSM/osmo-operator/configs/osmo-upf.cfg

238 octets, 11 lignes → 11 lignes

```
log file /var/log/osmocom/osmo-upf.log
 logging color 1
 logging timestamp 1
 logging filter all 1
 logging print category 1
 logging print level 1
pfcp
 local-addr __CONTAINER_IP__
nft
 # netfilter requires no specific configuration
gtp
```

3.67 /opt/GSM/osmo-operator/configs/pjsip.conf

13534 octets, 354 lignes → 354 lignes

```
; ===============================================================================
; pjsip.conf — Asterisk PJSIP — Opérateur __OPERATOR_ID__
;
; [2026-08-31] CE GABARIT ÉTAIT LE SEUL DES TROIS CONFIGS ASTERISK À NE PORTER
; AUCUN MARQUEUR (grep -c "__" : extensions.conf 3, osmo-sip-connector.cfg 1,
; celui-ci 0). Il était donc recopié à l'identique dans TOUS les opérateurs, et
; op2 comme op3 s'annonçaient « Op1 ». La substitution, elle, passait déjà sur
; ce fichier (generate_configs.sh:481 couvre "$dest/asterisk"/*.conf) : il
; suffisait de poser les marqueurs.
;
; Linphone depuis l'hôte :
;   Direct : __CONTAINER_IP__:5060  (Docker bridge)
;   linphone_A / tester → 100  |  linphone_B / testerB → 200
;   (100/200 sont des postes SIP locaux, HORS plan MSISDN <nœud>00<op><ms> :
;    le dialplan les traite dans [internal], à côté de 500/600/700.)
; ===============================================================================

[global]
; [2026-08-12] debug=no par DEFAUT (demande explicite). Cette option journalise
; TOUS les messages SIP, REGISTER compris, et sature /var/log/asterisk/messages.
; Pour une capture ponctuelle : `asterisk -rx "pjsip set logger on"` — reglage
; d execution qui ne survit PAS au redemarrage d Asterisk (mesure du 09/08 :
; capture armee a 02:52, perdue au redemarrage de 07:32, INVITE Linphone rate).
; Pour une trace qui survit a un redemarrage : repasser debug=yes ICI, puis le
; remettre a no une fois la capture faite.
```

```
debug=no
type=global
user_agent=OsmoAsterisk-Op__OPERATOR_ID__
; Les clients (Linphone) et le trunk WAN arrivent tous via la gateway Docker
; 172.20.0.1 (NAT). L'identify IP du trunk WAN (network/setup-wan-interop.sh, match=
; 172.20.0.1) avalerait sinon les REGISTER Linphone. On identifie donc par
; username d'abord : Linphone -> par username, trunk WAN IP-only -> par IP.
endpoint_identifier_order=username,ip,anonymous

; ============================================================================
; TRANSPORT
; ============================================================================

[transport-udp]
type=transport
protocol=udp
bind=0.0.0.0
; [2026-08-31] external_media_address / external_signaling_address = 10.2.0.2
; RETIRÉS. Cette adresse était codée en dur, identique chez tous les opérateurs,
; et n'existait ni sur l'hôte ni sur le réseau du banc (172.20.0.0/16, déclaré
; en local_net juste en dessous). Elle ne sert qu'aux pairs situés HORS des
; local_net — un softphone sur le LAN — auxquels Asterisk annonçait donc une
; adresse injoignable : signalisation OK, RTP dans le vide. Sans ces lignes,
; Asterisk annonce l'adresse réelle de son interface, ce qui est correct ici.
; À ne remettre que derrière un vrai NAT, avec l'adresse publique réelle.
local_net=172.20.0.0/16
local_net=127.0.0.0/8

; ============================================================================
; LINPHONE A — numéro 100
; ============================================================================

[linphone_A]
type=endpoint
transport=transport-udp
context=internal
disallow=all
allow=alaw
allow=ulaw
allow=gsm
auth=linphone_A
aors=linphone_A
direct_media=no
rtp_symmetric=yes
force_rport=yes
rewrite_contact=yes
media_encryption=no
callerid=Linphone A <100>

[linphone_A]
type=auth
auth_type=userpass
username=linphone_A
password=tester

[linphone_A]
type=aor
max_contacts=3
remove_existing=yes
qualify_frequency=30
qualify_timeout=5.0

; ============================================================================
; LINPHONE B — numéro 200
; ============================================================================
```

```
[linphone_B]
type=endpoint
transport=transport-udp
context=internal
disallow=all
allow=alaw
allow=ulaw
allow=gsm
auth=linphone_B
aors=linphone_B
direct_media=no
rtp_symmetric=yes
force_rport=yes
rewrite_contact=yes
media_encryption=no
callerid=Linphone B <200>

[linphone_B]
type=auth
auth_type=userpass
username=linphone_B
password=testerB

[linphone_B]
type=aor
max_contacts=3
remove_existing=yes
qualify_frequency=30
qualify_timeout=5.0

; ============================================================================
; TRUNK GSM ← OsmoMSC via osmo-sip-connector (127.0.0.1:5061)
;
; media_encryption=no : CRITIQUE — osmo-sip-connector envoie du RTP brut,
;   pas du SRTP/DTLS. Sans cette option Asterisk peut interpréter certains
;   paquets RTP comme du DTLS → "Received SSL traffic without SSL session"
;   → audio jeté → pas de son.
; ============================================================================

[gsm_msc-identify]
type=identify
endpoint=gsm_msc
match=127.0.0.1

[gsm_msc]
type=endpoint
transport=transport-udp
context=gsm_in
disallow=all
allow=alaw
allow=ulaw
allow=gsm
aors=gsm_msc
direct_media=no
rtp_symmetric=yes
force_rport=yes
rewrite_contact=yes
media_encryption=no

[gsm_msc]
type=aor
contact=sip:127.0.0.1:5061
qualify_frequency=10

; ============================================================================
; TRUNKS INTER-OPÉRATEURS — GÉNÉRÉ par start-docker.sh
```

```
; ============================================================================
;
; ============================================================================
; TRUNK ZADARMA (opérateur SIP externe)  — https://zadarma.com
;
; ┌─ À REMPLIR ──────────────────────────────────────────────────────────────┐
; │  ZADARMA_LOGIN_A_REMPLIR         ← votre login SIP Zadarma (ex: 123456)  │
; │  ZADARMA_PASSWORD_A_REMPLIR      ← votre mot de passe SIP              │
; └──────────────────────────────────────────────────────────────────────────┘
; Les deux se trouvent dans l'espace client Zadarma, section « Mes SIP ».
; ⚠ Le login SIP N'EST PAS l'identifiant du compte web, et le mot de passe SIP
;    n'est pas celui du compte : ce sont des identifiants dédiés.
;
; Après remplissage :  asterisk -rx "pjsip reload"
; Vérifier :           asterisk -rx "pjsip show registrations"   → Registered
;
; Composition :
;   sortant  → préfixe 00 + numéro international   (ex: 0033612345678)
;   entrant  → contexte [zadarma_in] du dialplan
; ============================================================================

[zadarma_auth]
type=auth
auth_type=userpass
username=533110
password=3eahh8IWZC

[zadarma]
type=aor
contact=sip:sip.zadarma.com
qualify_frequency=60
qualify_timeout=5.0

[zadarma_reg]
type=registration
transport=transport-udp
outbound_auth=zadarma_auth
server_uri=sip:sip.zadarma.com
client_uri=sip:533110@sip.zadarma.com
contact_user=533110
retry_interval=60
forbidden_retry_interval=600
expiration=300
line=yes
endpoint=zadarma

[zadarma]
type=endpoint
transport=transport-udp
context=zadarma_in
disallow=all
; [2026-08-13] ORDRE INVERSÉ : gsm EN TÊTE. PJSIP propose les codecs dans
; l'ordre de ces lignes et le pair prend le premier qu'il accepte. Avec alaw
; en tête, Zadarma répondait alaw — mesuré sur l'appel de 08:15 :
;   PJSIP/gsm_msc  NativeFormats (gsm)   ReadTranscode (gsm)->(slin)
;   PJSIP/zadarma  NativeFormats (alaw)  WriteTranscode (slin)->(alaw)
; soit gsm↔slin↔alaw, DEUX transcodages par sens. La patte MS est en GSM-FR
; par nature (c'est le codec de l'air) : si Zadarma accepte gsm, les deux
; pattes parlent le même codec et bridge_native_rtp peut relayer le RTP tel
; quel, sans repasser par slin.
; ⚠ NON VÉRIFIÉ : que Zadarma accepte le GSM-FR sur ce type de compte. S'il
;    refuse, il retombe sur alaw/ulaw ci-dessous — aucune perte, l'appel passe
;    comme avant. Le juge est `core show channel PJSIP/zadarma-...` pendant un
;    appel : NativeFormats (gsm) = accepté, (alaw) = refusé.
; ⚠ NE PAIE QUE SI LES AUDIOHOOKS PARTENT AUSSI. Tant que [sub-record] pose
;    MixMonitor + JITTERBUFFER, le canal est sorti du pont natif et repasse en
```

```
;    slin quoi qu'il arrive (ReadFormat: slin sur une patte NativeFormats gsm).
;    Ce changement seul ne supprimera donc PAS le transcodage.
allow=gsm
allow=ulaw
allow=alaw
outbound_auth=zadarma_auth
aors=zadarma
from_user=533110
from_domain=sip.zadarma.com
; NAT : Asterisk est derrière la gateway Docker. Sans ces quatre lignes le RTP
; part vers une adresse privée et l'appel est muet dans un sens.
direct_media=no
rtp_symmetric=yes
force_rport=yes
rewrite_contact=yes
; Zadarma ne fait pas de SRTP sur ce type de compte. Laisser 'no' : sinon
; Asterisk peut prendre du RTP brut pour du DTLS et jeter l'audio (même piège
; que le trunk gsm_msc plus haut dans ce fichier).
media_encryption=no
dtmf_mode=rfc4733

[zadarma-identify]
type=identify
endpoint=zadarma
match=sip.zadarma.com
; ⚠ endpoint_identifier_order (en tête de ce fichier) vaut username,ip,anonymous.
;    L'identification par IP n'intervient donc qu'après le username : les REGISTER
;    des Linphone ne peuvent pas être avalés par ce match.

; ============================================================================
; TRUNK ZADARMA PBX (extension du standard)  — pbx.zadarma.com
;
; [2026-09-09] AJOUTÉ POUR L'ENTRANT, PUIS DEVENU UN FILET DE SÉCURITÉ.
; Historique, parce qu'il explique les deux inscriptions : le trunk [zadarma]
; ci-dessus était le compte SIP 474117, et le numéro +33757690612 n'était
; rattaché À AUCUN compte SIP. Zadarma n'avait donc personne à qui présenter
; l'appel entrant : quinze minutes de trace n'ont montré que ses OPTIONS de
; keepalive toutes les 30 s, et pas un seul INVITE. D'où cette inscription de
; secours sur l'extension 100 du standard 578163, au cas où le numéro y aurait
; été rattaché.
;
; CE QUI A RÉELLEMENT DÉBLOQUÉ L'ENTRANT est ailleurs : le numéro a été connecté
; au compte SIP 533110 (« Numbers, connected to SIP : +33757690612 » sur sa page)
; et [zadarma] est passé de 474117 à 533110. L'entrant arrive donc par [zadarma],
; pas par ce trunk-ci. On le garde quand même : les deux tombent dans
; [zadarma_in], et une extension de standard enregistrée ne coûte rien.
;
; D'où cette seconde inscription : l'extension 100 du standard 578163.
;   Serveur : pbx.zadarma.com   (≠ sip.zadarma.com — deux plates-formes)
;   Login   : 578163-100        (numéro de standard + tiret + extension)
;
; CÔTÉ ZADARMA, TROIS CHOSES DOIVENT ÊTRE VRAIES — la config ci-dessous n'en
; couvre AUCUNE, elles se règlent dans l'espace client :
;   1. Le mot de passe de l'extension doit être ENREGISTRÉ (bouton SAVE de la
;      page « PBX extensions settings »). Un mot de passe généré mais non
;      sauvegardé donne un 401 en boucle.
;   2. « Restriction by IP » doit valoir l'IP publique du banc. Elle vaut
;      82.66.231.141 et c'est bien celle du banc (vérifié le 09/09) ; elle
;      change à chaque renumérotation de la box → 403 Forbidden sur REGISTER.
;   3. Le scénario d'appel entrant du numéro doit router vers l'extension 100.
;
; VÉRIFICATION :  asterisk -rx "pjsip show registrations"
;                 → zadarma_pbx_reg/sip:pbx.zadarma.com ... Registered
;                 et la page du standard passe de « offline » à « online ».
; ============================================================================
```

```
[zadarma_pbx_auth]
type=auth
auth_type=userpass
username=578163-100
; [2026-09-09] MOT DE PASSE REMPLACE. Le precedent (0wj5jcKYFC) etait genere
; dans la page du standard mais jamais enregistre : REGISTER -> 401 en boucle,
; extension « offline ». Verifie par un REGISTER nu, hors Asterisk : challenge
; sur realm pbx.zadarma.com puis 200 OK avec celui-ci.
password=QsK3m8fmNE

[zadarma_pbx]
type=aor
contact=sip:pbx.zadarma.com
qualify_frequency=60
qualify_timeout=5.0

[zadarma_pbx_reg]
type=registration
transport=transport-udp
outbound_auth=zadarma_pbx_auth
server_uri=sip:pbx.zadarma.com
client_uri=sip:578163-100@pbx.zadarma.com
contact_user=578163-100
retry_interval=60
; 600 s après un 403/404 : l'extension est limitée à 3 lignes et Zadarma
; blackliste une IP qui martèle des REGISTER refusés.
forbidden_retry_interval=600
expiration=300
; line=yes : Asterisk marque le Contact enregistré et reconnaît l'INVITE
; entrant par cette marque, AVANT toute identification par IP. C'est ce qui
; garantit que l'appel du standard tombe dans [zadarma_in] et pas en
; « anonymous » (endpoint_identifier_order = username,ip,anonymous).
line=yes
endpoint=zadarma_pbx

[zadarma_pbx]
type=endpoint
transport=transport-udp
context=zadarma_in
disallow=all
; Même raisonnement d'ordre des codecs que pour [zadarma] : gsm en tête pour
; éviter le double transcodage gsm↔slin↔alaw avec la patte MS. Si le standard
; refuse le GSM-FR il retombe sur alaw/ulaw, l'appel passe quand même.
allow=gsm
allow=ulaw
allow=alaw
outbound_auth=zadarma_pbx_auth
aors=zadarma_pbx
from_user=578163-100
from_domain=pbx.zadarma.com
; NAT : mêmes quatre lignes que le trunk classique, mêmes raisons.
direct_media=no
rtp_symmetric=yes
force_rport=yes
rewrite_contact=yes
media_encryption=no
dtmf_mode=rfc4733

[zadarma_pbx-identify]
type=identify
endpoint=zadarma_pbx
; pbx.zadarma.com et sip.zadarma.com ne résolvent PAS sur la même adresse
; (185.45.152.164 contre 185.45.152.180, mesuré le 09/09) : les deux blocs
; identify ne se marchent pas dessus. Filet de sécurité derrière line=yes.
match=pbx.zadarma.com
```

3.68 /opt/GSM/osmo-operator/configs/rtp.conf

286 octets, 12 lignes → 12 lignes

```
; rtp.conf — Asterisk RTP configuration
; Plage de ports pour le média RTP (voice)
;
; En mode Linphone externe, ces ports doivent être exposés
; via Docker (-p) et ouverts dans le firewall.

[general]
rtpstart=__RTP_START__
rtpend=__RTP_END__
strictrtp=no
icesupport=no
stunaddr=
```

3.69 /opt/GSM/osmo-operator/configs/sms-routing.conf

1873 octets, 51 lignes → 41 lignes (2 groupes compactés)

```
# ═══════════════════════════════════════════════════════════════════
# sms-routing.conf — Table de routage inter-opérateur SMS
#
# Ce fichier est templaté par start-docker.sh via apply_config_templates()
# Les placeholders __XXX__ sont remplacés au démarrage du container.
#
# Pour ajouter un opérateur :
#   1. Ajouter son IP dans [operators]
#   2. Ajouter les préfixes MSISDN dans [routes]
# ═══════════════════════════════════════════════════════════════════

[local]
# ID de cet opérateur (remplacé dynamiquement)
operator_id = __OPERATOR_ID__

[operators]
# operator_id = container_ip
# Ces IPs correspondent au réseau bridge Docker (172.20.0.0/24)
1 = 172.20.0.11
2 = 172.20.0.12
# 3 = 172.20.0.13

[routes]
# prefix = operator_id
# Longest-prefix match : le préfixe le plus long gagne
#
# ── Préfixes MSISDN par opérateur ──────────────────────────────────
# Adapter selon ton plan de numérotation
#
# Opérateur 1 : extensions 1xxx et MSISDN +33601xxxxxx
⟨1⟩ = 1  ×7
    ⟨⟩ = (10000) (1001) (1002) (1003) (1004) (1005) (33601)
#
# Opérateur 2 : extensions 2xxx et MSISDN +33602xxxxxx
⟨1⟩ = 2  ×7
    ⟨⟩ = (20000) (2001) (2002) (2003) (2004) (2005) (33602)
#
# ── Route par défaut (optionnel) ───────────────────────────────────
# Si aucun préfixe ne matche, le SMS n'est pas routé.
# Décommenter pour un fallback :
# * = 1
```

3.70 /opt/GSM/osmo-operator/configs/conky/dino.html

219265 octets, 1349 lignes → 1349 lignes

```
<!doctype html><html><head><meta charset=utf8><meta name=viewport content="width=device-width,initial-scale=1"><
↪ style>:root{color-scheme:light}body{margin:0;padding:0;font:14px
↪ -apple-system,BlinkMacSystemFont,sans-serif;background:#faf9f5;color:#141413}img{max-width:100%}[hidden]
↪ :not([hidden=until-found]){display:none!important}</style></head><body>
<title>Cavalier d'ombres</title>
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Rye&
↪ family=Barlow+Condensed:wght@400;500;600&family=IBM+Plex+Mono:wght@400;500&display=swap">
<style>
  :root{
    --ink:#131018;        /* nuit d'encre */
    --ink-2:#1c1822;
    --bone:#efe6d2;       /* os */
    --bone-dim:#a99f8d;
    --sun:#e0653a;        /* soleil couchant */
    --moon:#a9b7de;       /* clair de lune */
    --cactus:#4b7a4a;
    --dust:#c8a87a;
    --line:rgba(239,230,210,.14);
  }
  *{box-sizing:border-box}
  body{
    margin:0; background:var(--ink); color:var(--bone);
    font-family:"Barlow Condensed",-apple-system,"Segoe UI",sans-serif;
    min-height:100vh; display:flex; flex-direction:column; align-items:center;
    justify-content:center; gap:14px; padding:18px 14px 26px;
    background-image:radial-gradient(120% 80% at 50% -10%, #241d2b 0%, var(--ink) 62%);
  }

  header{width:min(980px,100%); display:flex; align-items:baseline; justify-content:space-between; gap:16px}
  h1{
    font-family:Rye,"Barlow Condensed",serif; font-weight:400; margin:0;
    font-size:clamp(22px,3.4vw,34px); letter-spacing:.01em; color:var(--bone);
    text-wrap:balance;
  }
  h1 em{font-style:normal; color:var(--sun)}
  header p{margin:0; font-size:14px; color:var(--bone-dim); letter-spacing:.04em; text-align:right;
↪   max-width:42ch}

  .frame{
    position:relative; width:min(980px,100%); aspect-ratio:8/5;
    border:1px solid var(--line); border-radius:3px; overflow:hidden;
    background:#0d0b12; box-shadow:0 24px 60px -30px #000, inset 0 0 0 1px rgba(0,0,0,.5);
    touch-action:none; cursor:crosshair;
  }
  canvas{display:block; width:100%; height:100%}

  /* ---- HUD ---- */
  .hud{position:absolute; inset:0; pointer-events:none; font-variant-numeric:tabular-nums}
  .top{
    position:absolute; top:0; left:0; right:0; display:flex; align-items:center;
    justify-content:space-between; gap:12px; padding:9px 14px;
    background:linear-gradient(#000000a8,#0000);
  }
  .dist{display:flex; align-items:baseline; gap:6px}
  .dist b{font-size:26px; font-weight:600; line-height:1; letter-spacing:.01em}
  .dist span{font-size:12px; color:var(--bone-dim); letter-spacing:.16em; text-transform:uppercase}
  .right{display:flex; align-items:center; gap:16px}
  .sky-read{text-align:right; line-height:1.15}
  .sky-read .ph{font-size:14px; letter-spacing:.03em}
  .sky-read .dy{font-size:11px; color:var(--bone-dim); letter-spacing:.14em; text-transform:uppercase}
```

```
  .lives{display:flex; gap:5px}
  .lives i{
    display:block; width:15px; height:11px; background:var(--bone);
    clip-path:polygon(0 100%,100% 100%,84% 74%,74% 0,26% 0,16% 74%);
    transition:opacity .18s;
  }
  .lives i.off{opacity:.16}

  .snd{
    position:absolute; right:12px; bottom:12px; z-index:3;
    font:500 13px/1 "IBM Plex Mono",monospace; color:var(--bone);
    background:#0a0810a6; border:1px solid var(--line); border-radius:2px;
    padding:6px 9px; cursor:pointer; letter-spacing:.08em;
  }
  .snd:hover{background:#0a0810e0}
  .snd:focus-visible{outline:2px solid var(--sun); outline-offset:2px}
  .snd[aria-pressed="true"]{color:var(--bone-dim)}
  /* Le bouton plein ecran, a gauche du son. ON EST DANS LA FENETRE A LA BASE :
     rien ne change tant qu on ne clique pas. En plein ecran le cadre prend tout
     l ecran en gardant ses proportions (8/5, bandes noires s il le faut), et le
     meme bouton ramene dans la fenetre. */
  .fsb{ right:58px }
  body.full{ padding:0; gap:0; background:#000; background-image:none }
  body.full header{ display:none }
  body.full .frame{ width:min(100vw, 160vh); height:min(100vh, 62.5vw); border:0; border-radius:0;
↪   box-shadow:none }

  .card{
    position:absolute; left:50%; top:34%; transform:translate(-50%,-50%);
    display:grid; justify-items:center; gap:4px; text-align:center; pointer-events:none;
    text-shadow:0 2px 18px #000, 0 0 30px #000;
  }
  .card[hidden]{display:none}
  .card b{font-family:Rye,serif; font-weight:400; font-size:clamp(19px,2.8vw,27px); color:#f6efdd}
  .card span{font-size:12px; letter-spacing:.18em; text-transform:uppercase; color:var(--moon)}

  /* ---- overlays ---- */
  .veil{
    position:absolute; inset:0; display:grid; place-content:center; text-align:center; gap:10px;
    background:radial-gradient(60% 60% at 50% 50%, #0b0910cc, #0b0910f2); padding:24px;
  }
  .veil[hidden]{display:none}
  .veil h2{
    font-family:Rye,serif; font-weight:400; margin:0; font-size:clamp(20px,3vw,30px); color:var(--bone);
  }
  .veil p{margin:0; font-size:15px; color:#cfc6b4; max-width:46ch; line-height:1.5}
  .veil .keys{display:flex; gap:8px; justify-content:center; margin-top:4px; flex-wrap:wrap}
  .veil kbd{
    font-family:"IBM Plex Mono",monospace; font-size:12px; padding:4px 9px; color:var(--bone);
    border:1px solid var(--line); background:#ffffff0e; border-radius:2px;
  }
  .veil .score{font-family:"IBM Plex Mono",monospace; font-size:13px; color:var(--bone-dim); letter-spacing:.06em}
  .veil .score b{color:var(--sun); font-weight:500}

  footer{
    width:min(980px,100%); display:flex; justify-content:space-between; gap:18px;
    font-size:12.5px; color:var(--bone-dim); letter-spacing:.03em; flex-wrap:wrap;
  }
  footer code{font-family:"IBM Plex Mono",monospace; font-size:11.5px; color:#bcb4a3}
  @media (max-width:620px){
    header{flex-direction:column; gap:4px} header p{text-align:left}
    .dist b{font-size:20px}
  }
</style>

<header>
```

```
  <h1>Coureur d'<em>ombres</em></h1>
  <p>Le dino court vers le levant : le Soleil se lève droit devant lui, monte, et sort du cadre par le haut. La
   ↪  Lune fait le même chemin, un cinquième plus vite. Les ombres sont calculées, pas dessinées — même celle du
   ↪  saut.</p>
</header>

<div class="frame" id="frame">
  <canvas id="cv" width="960" height="600"></canvas>
  <div class="hud">
    <div class="top">
      <div class="dist"><b id="dist">0</b><span>mètres</span></div>
      <div class="right">
        <div class="sky-read">
          <div class="ph" id="phase">nouvelle lune</div>
          <div class="dy" id="day">jour 1 · 06 h 00</div>
        </div>
        <div class="lives" id="lives"><i></i><i></i><i></i></div>
      </div>
    </div>
  </div>

  <button class="snd" id="snd" type="button" aria-pressed="false" title="son (M)">♪</button>
  <button class="snd fsb" id="fsb" type="button" aria-pressed="false" title="plein écran (F)">□</button>

  <div class="card" id="card" hidden>
    <b id="cardLvl"></b>
    <span id="cardSub"></span>
  </div>

  <div class="veil" id="start">
    <h2>Il va tout droit</h2>
    <p>Tant que personne ne touche aux commandes, la piste reste vide. Prends la main et le désert se remplit. La
     ↪  Lune rattrape le Soleil tous les cinq jours : l'éclipse est un spectacle. Il n'y a que des mètres à
     ↪  faire.</p>
    <div class="keys"><kbd>←</kbd><kbd>→</kbd><kbd>espace : saut</kbd><kbd>clic / glisser</kbd></div>
  </div>

  <div class="veil" id="over" hidden>
    <h2 id="overTitle">Dans les épines</h2>
    <p class="score">parcouru <b id="finalDist">0</b> m · record <b id="bestDist">0</b> m</p>
    <div class="keys"><kbd>Entrée</kbd><kbd>clic</kbd></div>
  </div>
</div>

<footer>
  <span>Soleil et Lune tournent autour du dino et se lèvent devant lui ; la Lune va 1,2 fois plus vite que le
   ↪  Soleil, et le rattrape tous les cinq jours. L'année dure onze jours : neige, orages, aurores, arcs-en-ciel,
   ↪  étoiles filantes — et la dernière nuit, le père Noël passe.</span>
  <code>ombre : S = Q − l·(Qz □ lz)</code>
</footer>

<script>
(() => {
  "use strict";
  const cv = document.getElementById("cv"), ctx = cv.getContext("2d", {alpha:false});
  const W = 960, H = 600, FOV = 52 * Math.PI/180, FOC = (W/2)/Math.tan(FOV/2);

  // ---------- petite algèbre ----------
  const sub=(a,b)=>[a[0]-b[0],a[1]-b[1],a[2]-b[2]];
  const add=(a,b)=>[a[0]+b[0],a[1]+b[1],a[2]+b[2]];
  const mul=(a,s)=>[a[0]*s,a[1]*s,a[2]*s];
  const dot=(a,b)=>a[0]*b[0]+a[1]*b[1]+a[2]*b[2];
  const cross=(a,b)=>[a[1]*b[2]-a[2]*b[1], a[2]*b[0]-a[0]*b[2], a[0]*b[1]-a[1]*b[0]];
  const norm=a=>{const l=Math.hypot(a[0],a[1],a[2])||1; return [a[0]/l,a[1]/l,a[2]/l];};
  const clamp=(v,a,b)=>v<a?a:v>b?b:v;
  const lerp=(a,b,t)=>a+(b-a)*t;
```

```
const mix=(a,b,t)=>{t=clamp(t,0,1); return [lerp(a[0],b[0],t),lerp(a[1],b[1],t),lerp(a[2],b[2],t)];};
const rgb=c=>`rgb(${c[0]|0},${c[1]|0},${c[2]|0})`;

// ---------- caméra ----------
let EYE=[0,-9.2,2.6], RGT=[1,0,0], UPV=[0,0,1], FWD=[0,1,0], HORIZON=300;
function setCam(camX, lookX){
  EYE=[camX,-9.2,2.6];
  const f=norm(sub([lookX,0,1.5],EYE));
  const r=norm(cross(f,[0,0,1]));
  FWD=f; RGT=r; UPV=cross(r,f);
  HORIZON = proj([camX,8000,0])[1];
}
function proj(P){
  const d=sub(P,EYE), cz=dot(d,FWD);
  if(cz<=0.06) return null;
  return [W/2 + FOC*dot(d,RGT)/cz, H/2 - FOC*dot(d,UPV)/cz, cz];
}

// ---------- boîtes orientées ----------
function box(c,h,col,ax,ay,az){
  return {c, h, col, ax:ax||[1,0,0], ay:ay||[0,1,0], az:az||[0,0,1]};
}
function verts(b){
  const out=[];
  for(const sx of [-1,1]) for(const sy of [-1,1]) for(const sz of [-1,1])
    out.push([
      b.c[0]+b.ax[0]*b.h[0]*sx+b.ay[0]*b.h[1]*sy+b.az[0]*b.h[2]*sz,
      b.c[1]+b.ax[1]*b.h[0]*sx+b.ay[1]*b.h[1]*sy+b.az[1]*b.h[2]*sz,
      b.c[2]+b.ax[2]*b.h[0]*sx+b.ay[2]*b.h[1]*sy+b.az[2]*b.h[2]*sz]);
  return out;
}
function faces(b,out){
  const A=[b.ax,b.ay,b.az];
  for(let i=0;i<3;i++){
    const j=(i+1)%3, k=(i+2)%3;
    for(const s of [-1,1]){
      const n=mul(A[i],s), cf=add(b.c,mul(n,b.h[i]));
      const u=mul(A[j],b.h[j]), v=mul(A[k],b.h[k]);
      out.push({n, col:b.col, q:[sub(sub(cf,u),v), sub(add(cf,u),v), add(add(cf,u),v), add(sub(cf,u),v)]});
    }
  }
}
// enveloppe convexe (chaîne monotone) — winding constant pour une union propre
function hull(pts){
  pts=pts.slice().sort((a,b)=>a[0]-b[0]||a[1]-b[1]);
  if(pts.length<3) return pts;
  const half=P=>{const o=[]; for(const p of P){
    while(o.length>=2 && (o[o.length-1][0]-o[o.length-2][0])*(p[1]-o[o.length-2][1])
                       -(o[o.length-1][1]-o[o.length-2][1])*(p[0]-o[o.length-2][0])<=0) o.pop();
    o.push(p);} return o;};
  const lo=half(pts), hi=half(pts.slice().reverse());
  return lo.slice(0,-1).concat(hi.slice(0,-1));
}

// ---------- le modèle ----------
const DINO=[92,92,92], DINO_D=[72,72,72], DINO_T=[62,62,62], EYE_W=[248,248,248];
const C_CAC=[75,122,74], C_CAC_D=[58,98,58];

// Le dino de Chrome, vu de dos, en boîtes : corps, cou, crâne, queue relevée,
// deux bras minuscules, deux pattes.  z = hauteur du saut.
function dinoBoxes(t, px, lean, z){
  const B=[], ph=t*Math.PI*2, air = z>0.02;
  const Z = z + (air ? 0 : 0.045*Math.sin(2*ph));
  const rl = lean*0.18;
  const P=(x,y,zz)=>[px + x*Math.cos(rl), y, zz + Z + x*Math.sin(rl)];
  B.push(box(P(0,-0.02,1.16),[0.40,0.60,0.42],DINO));            // corps
```

```
  B.push(box(P(0,0.58,1.44),[0.33,0.24,0.34],DINO));             // poitrail
  B.push(box(P(0,0.74,1.84),[0.25,0.20,0.30],DINO));             // cou
  B.push(box(P(0,0.86,2.20),[0.25,0.32,0.21],DINO));             // crâne
  B.push(box(P(0,1.14,2.13),[0.21,0.22,0.13],DINO_D));           // museau
  for(const sx of [-1,1]) B.push(box(P(sx*0.215,0.98,2.28),[0.05,0.05,0.05],EYE_W));
  const c1=Math.cos(0.30), s1=Math.sin(0.30);                    // queue en deux segments
  B.push(box(P(0,-0.76,1.28),[0.24,0.40,0.25],DINO,[1,0,0],[0,c1,s1],[0,-s1,c1]));
  const c2=Math.cos(0.58), s2=Math.sin(0.58);
  B.push(box(P(0,-1.34,1.52),[0.14,0.42,0.15],DINO_T,[1,0,0],[0,c2,s2],[0,-s2,c2]));
  for(const sx of [-1,1])                                        // bras minuscules
    B.push(box(P(sx*0.42,0.50,1.42),[0.07,0.15,0.07],DINO_D,[1,0,0],[0,0.94,0.34],[0,-0.34,0.94]));
  for(let i=0;i<2;i++){                                          // pattes
    const sx = i ? 1 : -1;
    const a = air ? (i ? 0.85 : -0.45) : 0.72*Math.sin(2*Math.PI*(t+i*0.5));
    const L=0.92, d=[0,Math.sin(a),-Math.cos(a)];
    const hip=P(sx*0.24,-0.06,1.00);
    B.push(box(add(hip,mul(d,L/2)),[0.135,0.15,L/2],DINO_D,
               [1,0,0],[0,Math.cos(a),Math.sin(a)],[0,-Math.sin(a),Math.cos(a)]));
    B.push(box(add(add(hip,mul(d,L)),[0,0.10,0.06]),[0.13,0.22,0.06],DINO_T));   // pied
  }
  return B;
}
function cactusBoxes(c){
  const B=[], s=c.s, x=c.x, y=c.y;
  B.push(box([x,y,0.95*s],[0.16*s,0.16*s,0.95*s],C_CAC));
  if(c.a & 1){
    B.push(box([x-0.34*s,y,1.05*s],[0.20*s,0.13*s,0.12*s],C_CAC_D));
    B.push(box([x-0.48*s,y,1.05*s+0.42*s],[0.13*s,0.13*s,0.42*s],C_CAC));
  }
  if(c.a & 2){
    B.push(box([x+0.34*s,y,1.22*s],[0.20*s,0.13*s,0.12*s],C_CAC_D));
    B.push(box([x+0.48*s,y,1.22*s+0.34*s],[0.13*s,0.13*s,0.34*s],C_CAC));
  }
  return B;
}

// ---------- décor fixe ----------
const RND=(()=>{let s=20260906; return ()=>{s=(s*1664525+1013904223)>>>0; return s/4294967296;};})();
const ROCKS=[], STARS=[];
for(let i=0;i<70;i++) ROCKS.push({x:(RND()*2-1)*34, y0:RND()*140, r:0.12+RND()*0.45});
for(let i=0;i<460;i++) STARS.push({x:RND()*W, y:Math.pow(RND(),0.75)*0.99,
                                   m:0.18+Math.pow(RND(),2.2)*0.82, s:RND()<0.13?2:1,
                                   tw:RND()*6.28});

// ---------- état ----------
const S={
  mode:"attract", t:0, dist:0, speed:9, px:0, tgt:0, camX:0, lean:0,
  lives:3, invuln:0, gallop:0, cacti:[], spawn:2.5, shake:0, z:0, vz:0,
  day0: 0, best: 0, flash:0, totT:0,
  eclipsing:false, cardT:0, cardTitle:"", cardSub:"", sky:null,
  meteors:[], santa:-1, santaSeen:false,
  wx:{cloud:0,rain:0,snow:0,storm:0,aurora:0,day:0}, boltT:0, bolt:null, glare:0, skyT:0
};
try{ S.best = +(localStorage.getItem("cavalier-best")||0) || 0; }catch(e){}

const DAY = 24;                        // secondes pour un jour complet
// [2026-09-07] LE DINO COURT VERS LE LEVANT. Les deux astres TOURNENT autour de
// lui, sur le même cercle, et on ne voit que le quart qui MONTE : chacun sort
// de l'horizon droit devant, grimpe à vitesse constante et quitte le cadre par
// le haut — coupé par le bord, pas éteint. Le reste du tour (la descente,
// dans le dos du dino, puis le passage sous terre) est invisible. C'est ce
// qui met la pleine Lune EN FACE la nuit : elle est à l'opposé du Soleil, donc
// elle se lève devant quand lui se couche derrière.
//   hauteur dans le cadre = vitesse angulaire × temps (× K_SKY)
// Le Soleil fait un tour par jour ; la Lune va MOON_K = 1,2 fois plus vite,
// et le rattrape donc tous les 1/(MOON_K−1) = 5 jours : c'est la lunaison du
```

```
// modèle (tant pis pour les 28 jours), et le jour de la conjonction, l'éclipse.
// Le Soleil seul fait le jour et la nuit ; la Lune est où elle est, invisible
// de jour, et se révèle en fondu — déjà en mouvement — quand le ciel s'assombrit.
const ECL_ANG = 34*Math.PI/180;       // angle de la conjonction : bas dans le quart montant
const ECL_DAY = 0.25 + 34/360;        // l'heure où le Soleil y est
const MOON_K = 1.2;                   // vitesse angulaire de la Lune / celle du Soleil
const MOON_PER = 1/(MOON_K-1);        // jours entre deux conjonctions (5)
// À l'équateur, à midi, le Soleil est à la verticale : l'amplitude du cercle
// vaut 90°.  La déclinaison ne fait plus que 5°, de quoi que deux jours ne se
// ressemblent jamais tout à fait : ce sont les saisons.
// L'année du modèle dure SEASON jours ; le creux du froid tombe au bout de
// l'an (jour 0 ≡ jour 11), l'été au milieu — l'hiver, c'est le passage à l'an.
const SUN_MAX = 90*Math.PI/180, SUN_DEC = 5*Math.PI/180, SEASON = 11;
const yearT  = dayT => ((dayT % SEASON) + SEASON) % SEASON;   // jour dans l'an [0, SEASON[
const froidAt= dayT => (1 + Math.cos(2*Math.PI*dayT/SEASON))/2; // 1 au creux de l'hiver
const sunAmp = dayT => SUN_MAX - SUN_DEC*froidAt(dayT);
// Deux angles sur le même cercle, deux vitesses. L'angle lunaire est celui du
// Soleil plus un retard qui croît de (MOON_K−1) tour par jour, nul à la
// conjonction : à ECL_DAY les deux angles coïncident, exactement.
const moonAngle = dayT => (dayT-0.25)*2*Math.PI + 2*Math.PI*(dayT-ECL_DAY)*(MOON_K-1);
// La hauteur ne suit pas un sinus mais une dent de scie : l'astre MONTE à
// vitesse constante, de l'horizon (u = 0) au zénith (u = π/2) — c'est le seul
// quart qu'on voit, devant le dino. Ensuite il redescend derrière lui.
function arcDir(theta, amp){
  const TAU=2*Math.PI, u=((theta%TAU)+TAU)%TAU;
  const tri = u<=Math.PI ?  (1 - Math.abs(u-Math.PI/2)/(Math.PI/2))
                         : -(1 - Math.abs(u-3*Math.PI/2)/(Math.PI/2));
  const alt = amp*tri;
  const front = u<Math.PI/2 || u>3*Math.PI/2;    // devant le dino : la montée
  return [0, (front?1:-1)*Math.cos(alt), Math.sin(alt)];
}
// Modèle géocentré : les deux astres tournent autour du dino, à une distance
// finie.  Les ombres se calculent donc depuis leur POSITION, pas depuis une
// direction — elles rayonnent à partir du point au sol sous l'astre.
const R_SUN_W = 70, R_MOON_W = 95;    // mètres
const R_SUN_PX = 26, R_MOON_PX = 29;   // même taille apparente, à un cheveu près  // la Lune est plus large :
↪  la totalité dure
// Le cowboy voit π/3 : l'horizon (183 px, hauteur 0) est en bas du ciel, le
// haut du cadre est à 60° de hauteur.  Le zénith (90°) est donc bien au-dessus
// du cadre : l'astre sort par le haut, entier, aux deux tiers de sa montée.
const SKY_FOV = Math.PI/3, HZ_PX = 183;
const K_SKY = HZ_PX/SKY_FOV;           // px par radian de hauteur
// Le ciel s'ouvre sur la Lune d'aujourd'hui : pleine lune quand c'est la pleine
// lune, sinon la phase du jour. Âge lunaire réel calculé depuis la nouvelle
// lune de référence du 6 janvier 2000, 18 h 14 UTC, mois synodique de
// 29,530589 j, puis ramené à la lunaison du modèle.
const SYNODIC = 29.530588853;
function moonAgeToday(){
  const ref = Date.UTC(2000,0,6,18,14)/86400000, now = Date.now()/86400000;
  return ((now-ref) % SYNODIC + SYNODIC) % SYNODIC;      // 0 = nouvelle lune
}
const AGE = moonAgeToday();
S.day0 = ECL_DAY + MOON_PER - (SYNODIC-AGE)*MOON_PER/SYNODIC;   // reste avant la conjonction
S.sky = null;

// aigrettes de la couronne : longues à l'équateur, brosses courtes aux pôles
const CORONA=[];
for(let i=0;i<34;i++){
  const a=RND()*Math.PI*2, eq=Math.abs(Math.sin(a));
  CORONA.push({a, l:(0.25+0.75*eq)*(0.55+0.45*RND()), w:0.05+0.09*RND(),
               b:(RND()*2-1)*0.22, o:0.55+0.45*RND()});
}

// ---------- météo : tirée au sort chaque jour ----------
const wh = n => { const x=Math.sin(n*127.1+311.7)*43758.5453; return x-Math.floor(x); };
// Tout le temps qu'il peut faire : dégagé, nuageux, pluie ou NEIGE (la
```

```
// saison froide, quand le Soleil est bas, en fait de la neige plus souvent),
// orage avec ses éclairs, et les AURORES, par nuit claire, une fois sur trois.
function dayWeather(n){
  const a=wh(n), b=wh(n+0.37), c=wh(n+0.71), d=wh(n+0.53), e=wh(n+0.89);
  const froid = froidAt(n);                                  // 1 au creux de la saison
  const aurora = (a<0.64 && e>0.66) ? 0.45+0.55*b : 0;
  // La dernière nuit de l'an, il neige — toujours — et le ciel reste assez
  // ouvert pour qu'on voie qui passe.
  if(Math.floor(yearT(n)) === SEASON-1) return {cloud:0.38+0.14*b, rain:0, snow:0.45+0.30*c, storm:0, aurora:0};
  if(a<0.34) return {cloud:0.04+0.18*b, rain:0, snow:0, storm:0, aurora};       // dégagé
  if(a<0.64) return {cloud:0.34+0.42*b, rain:0, snow:0, storm:0, aurora};       // nuageux
  if(a<0.87){                                                                    // précipitations
    if(d < 0.25+0.45*froid) return {cloud:0.62+0.30*b, rain:0, snow:0.35+0.45*c, storm:0, aurora:0};  // neige
    return {cloud:0.62+0.30*b, rain:0.30+0.40*c, snow:0, storm:0, aurora:0};    // pluie
  }
  return {cloud:0.82+0.18*b, rain:0.68+0.32*c, snow:0, storm:1, aurora:0};      // orage
}
function weatherAt(dayT){                    // bascule douce en fin de journée
  const n=Math.floor(dayT), f=dayT-n;
  const A=dayWeather(n), B=dayWeather(n+1), t=clamp((f-0.80)/0.20,0,1);
  return {cloud:lerp(A.cloud,B.cloud,t), rain:lerp(A.rain,B.rain,t),
          snow:lerp(A.snow,B.snow,t), storm:lerp(A.storm,B.storm,t),
          aurora:lerp(A.aurora,B.aurora,t), day:n};
}
const CLOUDS=[];
for(let i=0;i<26;i++){
  const puffs=[];
  const np=3+((RND()*4)|0);
  for(let k=0;k<np;k++) puffs.push({dx:(RND()*2-1)*46, dy:(RND()*2-1)*9, r:16+RND()*22});
  CLOUDS.push({x:RND(), alt:0.10+RND()*1.15, sp:0.006+RND()*0.016, sc:0.6+RND()*0.9, puffs});
}
const DROPS=[];
for(let i=0;i<240;i++) DROPS.push({x:RND()*W, y:RND()*H, l:9+RND()*16, v:520+RND()*420});
// Les flocons : lents, et ils dérivent au lieu de tomber droit.
const FLAKES=[];
for(let i=0;i<220;i++) FLAKES.push({x:RND()*W, y:RND()*H, r:1.2+RND()*2.2, v:38+RND()*52, ph:RND()*7,
 ↪  sw:12+RND()*22});

  // ---------- le ciel (ne dépend ni du cavalier ni de la caméra) ----------
function computeSky(){
  const dayT = S.skyT + S.day0;
  const thS = (dayT - 0.25) * Math.PI*2;           // rotation de la Terre : 1 tour/jour
  const ampS = sunAmp(dayT);
  const lS = arcDir(thS, ampS);
  const altS = Math.asin(clamp(lS[2],-1,1));
  const thM = moonAngle(dayT);                     // 28 jours pour le même tour
  const lM = arcDir(thM, ampS);                    // exactement le même cercle
  const altM = Math.asin(clamp(lM[2],-1,1));
  // Deux astres à la même hauteur mais l'un devant et l'autre derrière ne se
  // rencontrent pas : il faut qu'ils soient du même côté du ciel.
  const sameBranch = (lS[1]>=0) === (lM[1]>=0);
  // Plus la Lune est près du sommet, plus le Soleil la croise lentement : la
  // couronne n'a le temps de se déployer que là.
  const slow = clamp((0.5-Math.abs(Math.cos(thM)))/0.5, 0, 1);

  // Phase : la vraie, celle de la lunaison, reprise depuis la Lune d'aujourd'hui.
  // Elle ne dépend donc pas de la hauteur — elle est fixe sur une nuit entière et
  // a nettement tourné la nuit suivante.  Le terminateur, lui, est horizontal :
  // les deux astres sont sur le même cercle, dans le plan avant/haut, et le
  // Soleil est donc pile sous la Lune — ou pile au-dessus.
  const untilNew = ((((ECL_DAY - dayT) % MOON_PER) + MOON_PER) % MOON_PER);
  const lunation = 1 - untilNew/MOON_PER;          // 0 et 1 = nouvelle lune
  const kMoon = (1 - Math.cos(2*Math.PI*lunation))/2;
  const sepAng = altM - altS;
  const elong = 2*Math.PI*lunation;
  // Le limbe éclairé pointe vers le Soleil.  Angle Soleil-dino-Lune = 0 : la
```

```
    // Lune est de dos, noire ; il grandit, la Lune prend de l'avance sur le
    // cercle et le Soleil reste DERRIÈRE elle — plus bas, ou sous l'horizon
    // derrière le dino la nuit : limbe en bas.  Passé 180° (pleine lune, Soleil
    // à l'opposé), c'est le Soleil qui est devant, plus haut : limbe en haut,
    // jusqu'à ce qu'il la rattrape par en dessous à la conjonction suivante.
    const pa = elong < Math.PI ? Math.PI/2 : -Math.PI/2;

    // Éclipse : le Soleil ne peut atteindre la Lune qu'au jour de la conjonction,
    // et seulement au sommet de sa course, là où il est le plus lent.
    const sep = (sameBranch && altS>0 && altM>0) ? Math.abs(sepAng)*K_SKY : 1e9;
    const rs=R_SUN_PX, rm=R_MOON_PX;
    let cover = 0;
    if(sep < rs+rm){
      if(sep <= rm-rs) cover = 1;
      else{
        const a1=Math.acos(clamp((sep*sep+rs*rs-rm*rm)/(2*sep*rs),-1,1));
        const a2=Math.acos(clamp((sep*sep+rm*rm-rs*rs)/(2*sep*rm),-1,1));
        cover = clamp((rs*rs*(a1-Math.sin(2*a1)/2)+rm*rm*(a2-Math.sin(2*a2)/2))/(Math.PI*rs*rs),0,1);
      }
    }
    return {dayT, thS, thM, lS, lM, altS, altM, elong, kMoon, cover, pa, slow,
            total: cover>=0.999};
  }
  // dôme céleste : l'astre n'est situé que par sa direction — jamais par le cavalier
  function skyPt(d, hz){
    const alt = Math.asin(clamp(d[2],-1,1));
    if(d[1] <= 0) return null;                    // derrière le dino : jamais vu
    if(alt < -0.30) return null;                  // couché : rien sous l'horizon
    const vx = W/2 + FOC*RGT[1]/(FWD[1] || 1e-6);   // point de fuite des deux traces
    return [vx, hz - alt*K_SKY, alt];
  }

  // ---------- son : 3 mesures tirées de l'extrait, bouclées sans couture ----------
  const LOOP_B64=
   "SUQzBAAAAAAAI1RTU0UAAAAPAAADTGF2ZjYwLjE2LjEwMAAAAAAAAAAAAAA//uAAAAAAAAAAAAA
```

```
t/bRXcfBB2X/1+oFZniWUwIFxCNNEX23QSRMSRlTky++09nECvA+amCzC2yPt4k/9
    ZPGvh5qyQK2Jbw93XkbBgPMHfCt65gzZ3WrdvTnY6nMqtRxQAwtFDpJkFXb2G/Trkm/q/An
    //oY56oGzmtzd77k0KDvgFATd3opAAARoTcQlDxKrr8aJ1MaC3MmJ+9E5qXzKCgObPdShFUMQf/V+r/0f0++e/88sEV2iXIQAAAAeHCo+kR1ABqiWSsb/wzA0GP
ghK86ZA0GDuS9BCm4E7P2twpxeywxxC2OWuIPndV+dDMZUssyGOAxU2OhX8at2pvHccKe/6URyHIo+cPyi+9D7j0Tjzzwstd7/
    QCcOwepCwj6EIQ9fJ9RywlZGleRZ9arm0Z
    /////vPBygLBvF/RR/jmk1DvksiAAAAKhJE5JCMLCGU7bK9TVgOjLdpDUcTjsKgfJ4327Ktki4xduSDqN708BElvJJBhxL6REtso7guxdUwgZZIEyBh6Q1hIU4N
BZVgvvMbs9mw+bSGBTGrNVq7RP5SqDVGiDmp868Qzy/zn9OZc+
    m5Ta7OtgH97zZ7WKGvvBfgoc16wmtItydjACAok4GhAAduH8a9kY0qOVVhQ0XynyG+VmfcJPDbv
    ///cpJy085i3/L/s6bP/EnTUEy+6QgAABTJ2hKRsEFt9FG1m3dicYhtoPSWrECuE8+jcYjY2Z85VzOkN7CAP55szTAVLKj08BeVsO5DCxxz3YTAACIHIUSQgFBh
XaBJAlB0tZxeHctJwDI3jiFCSua0/nq7lMx9qivVx3gt6j2OgjpWkne1zBG7i7rVTPJoNFaUbekFtuxNwelOK1sBOJl+
    eCo8kTiqZKRnuXdoiDbFNsW7bT9PKt/5O/W/I2b1o1qgA1eHZlIAAEzeNDXEWIbZpD7zy14Yo8LdQRAHAiFR8naFCMx12qulV9/
    xHCAeAoSZ1YLAcehI2/xgcMBYUJhOEc2cBaKr3BkFFjYlOeWZ+YTr0WD+Z2/LffyL+wuwuH1i0LI
    //7gmTZAhPIXdh7CRPyMeILXT2CGo7VXVuspFiAsofsrPSUanNuFjaO8wbbf8QH9UvqWB99aBHmJi5CogGOAMTgc6HGoDgvZJTSqg/QQxbf/sbokXn1BxtX/2QK
FCKLKcnSbGSLRFiZNk1LZDolAuOZ9H8cZFByioVEKn6alaDoFw0TekcTJlklpLt2J1MqGaZgghRb
    //vp/0qSVe6SKSSqDP9V203ZXnkWQolxi8b3k5FNlAAAAq16CJSPqmzWDcFRCUVHGhxrzXociMoqvePODdDSyDtToWIQ6iCR9GWY3Yn74S5Voq0pqIWVZ33SZ/m
LnB9UBpACbzFFiVEh3ot3I/+kEOoEAAAAAzKYgA/
    CLItADkkAkOZuPoqfaFyltOOOvL6FiwElACIpDSPDpOitaB7VzEIuTTu8aUUej2NTTdu5qeGhtjXh0WSqfc5330TIom2yo1Rc6laklh5r3h5IJNOk1El7n6A9tx
    +PE4i
    //uAZJOAAyxAW9HpG7AwgNstPSkADUjDaaewxYC/h+y0xA1IiB2ZT/6sx/q0VUH5AAAACmUBBUJSYOdAFwZV9SoJOWGBGWDREqJXpFQ1cZJoojxEzJSG2iEEjnK
797w60OAoOEtynUBVitLaYUBF8LHh1IGYjt167JhEJUBJAEECcLJquZBhgvQME8PwlbOuENJc8KyYiknQMY6LINmrD31C9KlN9vd/2
    //0f+uoBekAAACnHwyAyQ6QWqB8wQiALiqTFYTqRLCVlhA8DF1TKvRK0SNHCMpQLwYDMGpaZgYVPJHkAdjBMQB00dQn/+4JkroIC4yDbUewaNDLCGw09I0gLZNN
dml+64
    PGZoRO1FlT0tj4QQ7xy1jgcempPEMe0WTSMSijZQTSgteiXiAbxQYRdOEeev7mekIs9S5lU8q0YlLME9rjSDLQEysEAEGyYNgBYq83oFWSaZN
    +q0AgQAAAyqFcChREEaMD1qAPc7SdCJjNrKoZbMKwTQYAkSMyaGYwf/+4
    Bk2IICvhfaaewaMDXhqvpl7CaLLGdhTDEkgNOJKzT0jch62kSzYc9LHn7T9lJ6yDzoBetknDJr/1t/6/Z/+oB/1
    SAASnH4IQfpbAgk4uJvq8kRYxXwgxSG4YhWnQ4XFFYZR61UblNtFwaf5n9X8e4Rjlq8PxuCsjWmlI70iIzEEIY6Z5NmdxhGXLl+
    a1OeoMYqbEQkDoXtuoEB5QjJD5spy4vGjxNN9JJgOWUoMMFq8L4GWoVOOsekWs6UIdVxGELV7EXma9bcJT/
    WVFax0lcPee9eSIC52CQZn3333f/d/29TP/zaA6qQAAFKRyKApkgcFAVAt5eCVKo9Q4oGx8tZtxPijZA3/FO8uVkuyGUH/3
    YseiUuuLHd3etNWlse6RgHMOaajMxCyC8sa76KxWtrKnY+Zd3RP06xKhHIhXOUTDBkqi5zbK9M/
    KpGS5qBIIucBWC7hcl3ONmHWXRCxJUadhe3agUE6x0c8v/7
    gmTvAAN2Q9dTCRtQQ4MKamGJYA047WFHpG8Q8oxq9YMdyADof93i9cJzRJnbDiBv6hh5m3/9m
    //ug3SEAARODQxAGXrIKYQYkRBUob7gTQ/1OaBiv1Sf1LRlqA/gTz1/vK9iUpVPYiJlAxM7g6BANpspdeURYRbB5sTHBbL7YtQZ2Rm+iLtWVv0IzLyu9OoasxVZ
quOuSezUYerdaizjkxVFGuzTUTZWJ89W1tRtGU/uQGd1QX0KxL0bUWL7Od9u/pqP1Q3/
    a1WkYwQYpjDCK1TtSm9GTUurUZq57ChSbsrKQBXHmIJgfRQLQaPgbBOGZAhc6gNwtP6Rv/P/MbPP/7
    gmTnggMlT1jR6SrkOqLKzT3mCg1RaVvnjLiA9qBqNPYIaO8z
    ///9t/VTF9r++/9X1Q3/2q3MEZVoAA4JFMgCc7OF8GEbxL2YlR/PiWJTaCjRzIQLkwib16KOePVcfjDtehkCm607N52G0G08iMQwkinRiCpsr3YVCQaLopU/TyS
n1YkCMdH8k08bEOnPdtcpCHpF0er3lQkIJzke0e++7zYrWT71I8nmRdo1qEBhWH30PTjn+2u1ijQobtHvfDqcuvrgZcATY+
    lFChWaeDAdYoDBcPsJb0z3L
    //uCZOsAA29a1vnmFGI9KFqdPMJSDhlrW+ekTwjiIGr5hIkU+WNSf5/K9v7J/////T+v//VuO3o9RJLsrQBN0Nl/yY0jXXKGG3oaetrztuNADX10QDRmRKVUdf
hKetKKZWe7Ueuapyw0txcoTEKZLzAyWZrIseQB5VPljWn79lvZPdTVTz1NW+
    /4iOve29jN6KkTdtqvY3YrUtqYYdn5T7XS//uCZOSCA4laVWnsE/AwoossPSIpjwE1U0ekscDZlat88Rm571qHOWpNLWwAAJAB2EwP9Jp9gkof60wMShX6yvNoz
    +GHWXqWM+04Mc8LZyCykeaxDi+q2DS0S3jw9vFBSxFisKoRcLgsC7vndZ/P9RIm/u+
    84YXMcDnU0GbWH65pnWtfOtxm9zvr/41I/ul2aXesZpqffh2/i5i3tb+NnW6zZxjWM0/3
    nby9omt4pLsSOFBzHgBFBADgCQEANTK2YosjQGxjFjB4eqYHA0ungdZrzjMMddgDkE6LwfioJtAR8i0havL2wsGW00ThbKJ+
    Fp4o1Lh7NelXrBXTOm2hV
    //uAZOEAA+hP1n09YAAwpPqsp4gAWJWVW/mngAIYHup7NPAA2kw8oxPb+7zv9M7E6cVmy0561v/VK3Z79Yc21uW3OB8Zv//9/X/x/CmQ/4YwXt9fww7//+nn1RE
Iu6LFyGjeLMFvSZ7sB9Va6Z4g9RM9HZbkKI0Ovc2TpJ1uwuBEJUA2BJGZdG2BmLlS3TGz/+4
    JkiQIEuktZVz2ABjCDK2znlAAPGO9lTDEtEOYQbCD0iVi4jHnqEyFdgYbbif8lupBDGSSAMJnvOeZl+L9ZcsaReRtXvjlKWYWVaf/0
    p6kgISv5AAM80xon+MV6P5yVrCrEgJ0UgbXWPMkyccO2lxmMYkba3Qt281vmVBRzZOiggMElLDCJ3g+
    n6QDWEAAAAfTuUgJQg4ikyis4KgGQa9SsYFhoUT9lXjrmWwJ7P3S42cKH2G+q/JxTb3yA3Z6mFtHFHLzlCwV+
    E1w7agW0v97tRwPnR2GCN04ldfCwRClA/9E3xP0bjyE8MVNmMuh2avqFEYu48UY9I032ixJZJc4lXx+2xikVLHkyoCkMARlBi/Wg3T/
    PxCJ4rG0BHbjIxisehPPMoa0ysc7LoSd
    //b+6U7m9lrKg7j8mP2v/96bbZ0AE0gAAGZBpQRGp9Sh6tjF3neRpLVGJNrGYHeeer3Y/FP72arPPLLl10Vb/+4JkcQIz3jzYUw9hwDuFGz09gioQgSNfrDBTwM
OhKpiWFhLET2sJVp1liIsaehkki1phIglAOkH2K+lhGzn/+4BkXoMD4UtX0wkUcDLBmug9iSYN1T9hB6RRwNkYbHDwimgNEsjVThe550ohdnR/2
    fsrKkMJxZmhL2tLPN6ys2dIEQqSI0tne/MhqYjZBwOwf4LU0ilNuUUbJDj4x8WxkTsbdWoPyGM6q4u+
    pU5kdmyqiCezTtzexQbUpXzDFUDP7fj2E7bFUkXgDJQs1jsOR0TRXmkVJjJ9GaqjzIrHYNUtBpCgEylv/
    mky0X8ma86D9MrczAifUZOhRfYUgAAF0EYToggQ8JQTseSpGmC/DVqY85Zj/EeVa4UQQlIUOklBO5EJGdZqeMtXUpTtW7vVqvPW+
    aaMYatSYJwuXrkgoKAu6sLECs+QxznLhjGnCPS84pJ5Rl7+
    wotOgYHCMWYn7gEYIipLeLk3oITmjKVIEjaCBjcHmhhDTzRA6B6GEpIEYqAVRMGa995Aa+9/
    cOQFSiFUpilizkV6vGBShHMIWKaAAADe3ZPYnQWnBf/7gmRZAAN+Q9jR6RvkNKIq6z2GYg4VAV1MMQ9AzA/
    sdPYIeAHbXsy5g8HudVntQE9zQdFo7O5rQEyzdx9uTder2BuBcW1s07X1i6YUmuzXJplGiobOdxXpqUg1jGU8Sj4IbVxTv3ihBO7v6Wx46o3pL82K
    +JrgdR5X3kyRYcLIUw3+aHN7a0C461UCGBQAiTRK9DAPnAtNSwpG0CmSqgC7nWFks6JDzHTa18jm
    //6IrOgwz//7IcM76SC01FEiSSnLGMC5Yw0ENVwhxJjKMZtmqboZh1p5UJxuonXMdCw2aHDnhJY0lvD5vS6GdS1I/Yl0YU6MPualn3xIwpiCHowFDLkI+GfROII
```

```
COB4KcZqhIk8B/Cvi8lf/7gmRaAQNTNdrp6BykNsIb/T2CSYwpCWcHjLOQ3IarMYewkI5C3LMSz1D1RHYco+KRM8DgkIFaxjkgTuLVUpH/o3vrM+
    GcntgUEXjrDK6eWVV2ZAk7OzsiTaTOpcyEKx3FQo9VCGAqLWOApGvr5t0azeAI7EmAIQCPsjHlsERiZcPNNhyhwBPsA4NgiRl9bGBHxdg+
    WIxYFw5SRHjgQUfx5AmKB39sV/61
    AFgAgAApx0H4B1EsBZTp4IaX4mptJM11qGKxfXlBOlHSH3k00SRV2bqSPIimQSuZ1p6KFmQeR3dPtULPR1Tl19kWzxh4GoBrIvl3HEFhJ1PPqy1763
    /j+isARyokLAIEAqJp3MVQHO0JHROEdEsJYwhmJWROIx06Y1RK7AMmiVCQunk0U0SWHEdTnfsEJ3/
    rACgQEPJ2QIikJFo3MdOUNYeVkYHKBQgjpZwzMN/0mV8hYkPdTIGDM4e4MQpHjifk86Dwnf/7
    gmRiAQLINFhR6StENwMKzGHlOAq41V8MMGrQ5QhqdYeM4Khnw1QjQ920vG
    //vlfdoONYxN5JhV7onNnf/ad/7peS2FAAA6AAAADeBGSqVFMuyCrP4t9uEEQ9HDQELZx/rB11uhifwD7Yhg3wgx7262Uuky57+uzb/Cbv+6sAAAyhwyFgIqFRI
R3ggK8KFabD0FQOEHaqmHmKgodJ2OnpEkA3QZrdPYNwFxEOTZJkkap3l62ztIgLo8GxFxnA3HQotzUOlkuuz/vhxdVv/9X
    //bQxZ7gAAAKaOkilKVbASKTgtAaITAnCOkWIY61WFBY0YgvXuDk1PGGQ6pvcK56DO/3dzdun2v/tNKHM176UZFwHh4q2X16ov9j3vHFyDcgkYkkXIbO7U9TfUP
0tYaekqsDVhWp0/CSQm4MSTXOtv+TTNvxiGYuc4MAg4EDaHD1AEMjl09ceBwjCwFAb2LNhusPGwsREcAo/
    ZX5EFkNqVJhAQAAsKwFFwqZ6RK8WUbam9FiI2MRForXhTOeEXCAJYqfEqus3BQ+C63t1vd
    //60U2400QQEnKLg9Yz6RVkOF3bzJOpWNyhQ9blj2nZtMYW1kDirnA7e/8wCCbky8L55H3/UznlCkcPhI0XsLHzqj5cBJUZenkDgTcgHEWTa7DykjC36ithWIQc
rqZljf6MrsIjMzZiIppRDo+AWDhMX/URHE2WA3uTxD0IiEzKlRnmqB9IKQxl23LTIycTGE0LHMyzVS5q/0o4Hx83Us6AqS/IzZb/
    l13qn8ZGJZpM5dRaaNO9YkUAUQOLpsvR3utkY5GaJKzkjHRTNbSnVBbq35DtItTLU0rof/
    or0KkuzPScSszwupUlsEAg2gYYwdapFeUxrBpZGNVM9U171KtYzvfqquSfrP9W6Pf97hEjWdD2TY8aLKGG/+
    pppVTsRUBlVJxtpMIkgqSi1HMIc3C3kzBikmOAb78nMcy1OrnzqaCxwjTiBE54EgoNqFKsQQq3E/Ai33Rg+E85lqYF4qa+LuvuacE1Ku5r
    //v+kikeevf75G6+i4ZD5QBcqbFLePCBkZHSwfTzMs5/yH4rM5yMUQGFzEtBGswAgEFiGBEeb/+4Jk5QIDSF/Vaew78jOlKmgxB0wOVX1Vp6SvyPAU6nT2FLDpE
fOqXtSJYtSEXKRC3jeZbOfqOk3
    ///8a+v6U3r/7zbf/7Kn0Q6VB3o9ga7OUSIjhLFamroRkJgEEx4M1eACyFYhahC4wGUiUSEY3mHhIA3YCrS78qiMflxYGUcaHIJNbhsrg2uLGqS/482J8+PPiLj
QBoX+r1BFxmZSG1NuYrTCFD7fNNVrdY8HCNWfVCirjaKRIJToOmKcIoQaBC0tsnDcpUkWEv6QUKfepFWq1aUD3LjUuzaUWKm9Ygml+
    KpsLti0lxQpfOGJmxNUM7PMqIBGUYsqRdViq6k5L+TGs/
    Is0O5dQfvJ5oZIHGEPizDoJFHImFvVyijQmciaIAVdywlEAAqYeTDzaapHIJBRi2GU2rhKMdlSf86+9ZXYAtEtVk4DDL8CGOfKZnb31xTL/
    mJkDDMDRx7PYYhocjF+X1WKqSk5L/sdb8izQ7/UP+T9PSjBEuBgIeIH/+QBu/
    taJAAAUxNJJGGGoNieNdT6w0zeDoGuQFD0qsTCShQY002GZVWo+81pg
    //H8TK/PYtP7D//7gkSBgAN5S1np5hykXmkbPWHjDgypMWGsGG/BpxVstYYY2TkNlfZKFbveZ16zQYQ623kZXOldnwp3/mRXn1F4x+v7fY1v1b28oU6SKqTI39F
fvnySqE7AUQOE88Oy63L6NuIxPmtmxwcuRry83ya9bwPwRWe3cgZawQDgg8MGzKwocILMuC5ooDYwJNdr/+
    PGgKTNiACFgAuM8eSJNw1z6Tx5wEurK2V8tvDnReyrs6WMpLfOf5LnoS3p0YDEOgKaYTE/BzsL9/
    BtZ35IAIZQiIQAAABeEIhGZEBM1hL6scclYz5OPL47CKKdjFtM0mnKVKqaPA7m2o9aJV9Dvdl/mld1J+
    ysHAY7S3QSKEmgmAkDJ6xIsHnHFINlyWJvZR86xR1DCU04IjSx9jmBBCqckH8o8D1Uu7aVrWuxACZwqAAgCQABwxhJR9mMfCKGZKGxeDCSRt4
    /u18ySLzEs
    //uCZEiCA300V3sGQ/A7APrvPYkiDTTBXaw8w0DuFK108InqdDNaCqknRQ+fegGAUHq/Z///y5w0qg4psBgI+usgBTAJgGKOCb0lCIFCfAy36pRyEzPEijD8JAx
Vn/17LvVmK3t0e1FrM2Wrcicouz2SuiM62a1/Vnsl+cerAN3KoEAACRUZMDqISwvFwuhGKzkjL5Xev3pK5HUzLr80+hr/n6uqHBSXl2SF
    ////////R6CMBCSNsEAAB3NdOUCwhKNhzGAjMhAgIpdhLNi1rpiY0pNLvu36I63a+yIy9b62Rn6Tdv70hO/s+spd5303SBo0HVlTwvLH5Ua5LfCWIXaKCugJAIC
85kJJCJuhNXFMqnUqjEGuROszNo7nVWJsrpgVnK
    //1F2OJnATEKBGJ2fmPqVOmw5nv3MeXJhxTw5mgCeAABSs3eXxGsZWsgWg8sIhzhisULqLPt9SpMrauR5jIrs8Ez8qVf2kRBzo7ylSBPGIO3SyIlEFJ1h6N92FQ
C5EkSnLT+BfjtZkOTCTGWaLo31MtK+4wKoeOIC58R7uhU7OC953Mlp+PWOzHP9Fuul50paiyYRVkVnfr6Jqnf/9
    fKtz7ylTz19ZokCRFQdp7aP1gvsCAASxQ7ITRD55Wksjb56G4O9k5F6LQb/0Suun
    ///br0Wpnqk6wzGv9ZZ4gd///SpC7VhpSoly2ySJIhFuqYH4PeIZaQJOXxDVOnnI08aJBKsY8XqeL+y+3u1vOU7du5flsa9nvvfdVe1fXFg3Ckirac3EunVtYIh
1XZc1weOqqiMCrmnvTtNWeZ95QQuSXdxhg4zzTkARzKJKb3VxV73c7RW6STZUa+
    /YtIe44R/7P/uoBuVAAAFdzDDDjEzCnSkSqgslpyy24MMV2/D51bEfj3niosJBPRDOKW4dCxZhuqOWZNNoWxwukt1e02ggKwdVEHClVn9XxIsE4enz/8f/
    +t7N/dzveny3Lrpze2KrJV0vG11COoLIXSLbb54Bn6wAEaAAABYGHGnhXGGMD1dkbCEgWUN3a/KKZ3f/7gmTjggK+
    NNZTAzv0OaKKnWXiHIyo0VGtGFGBG5TotaEZ8Jc8dyI227yu26GutUVRzIhy1c7HEVOh9FRBwjqVbUQ/SEBYM
    //f/+oAARNEOXRCJUOIJBtWj+XY2TA/06KQPCbagPZRUdF+ro1AglL9QW/yxnoaqX4WYSh/PCEF8QwHKr04RcRgG4rTpIYtn4XUzztG4houxbDrZ9MamfyMEeC4
SltfyQc0szq2AcbdLbFpz9jXYb723PabVx+m2j1eiFdqBaO8e9ujXeyVia97Zo2MJoRGC80fV8vbseJWTVob6HBxjH1XFsbfsKF+r/
    CjtMo9vP6VrB+XuYBqCYrrCUAAAAAAHYABwBAoztss8ZSQNEm9m6zvPddl8of+
    xGuLXzlwvHiR81XtGUtrEaXDBPdepfatI1BePC4e092q41lb+4OJ7ZEqdKoelZa7eqEw3U4P/7
    gGSLAAYdW1juaeAAhQg6Zs0wAA4lJ2G9kwAAwgLst54gAMkMfEtqvr6FatDrsqzaZqdIDlG3ZvXL2lms223zPTkttEt9syZGPOCOISzWva26UnOwsA2jACAAAC5
    +Lo5IYj4Kdii9oVGVW1scvY9ZbXdmW5nrypM9fum+rSZ5rPmxGft82HbM/zH+eJP9+b9726H1o72/Z98ZsSsiH5WB96sWEgJOKT45BUJ+
    SIyENM3gH2LiaKbV0BHkyfUjCpgHbkojVm/XQguLJqlBzAKoPAh0enLmS+f/
    Kb1QI24gQAAAFKcZI8mW6bxlEArhiTQZuCZK6z7Ok9lBfIVEVE803py6HHoa6rpK5KVspEcoXVQZq4Q1tHP9iVZNsFNLKJonOrO0ZihgzV6QAhQOLIhBV3zOusa
    //4vy//uCZDoCA8BJ1+spG/I15IrbPSU4DmEpWyyky4D8hCv095gQbZH3IQKwoyADgsA5j1QSfSRrr+5CmFQ+buCVxz5ZlRFZ115lIvqZfYncq0YB41pnkHi0uZ
ZtS1TAJaKc4J2ENCrB1IpJo+BsUNYKRRcKt+KshrJloHJ8XIjxinB9lQwq0S2Shla71Nz25SqeqmsXJGlKe+8ANaVkMiBAAKmOe0dU0oq+
    zpN41FM9gBOKInBy8QCfcsFUxXoULjYC7vMVu46ULiWu0ZzvGEVe0SRUBAgo0cK56GYxbUmNlsN17klxTNQwUsjtTKZLMmlmUDjQwckB
    //uCZC8AE2JA2HsMGkAvoSrMYekGDrklWawwccDCkit08YkgdSgaHCx4o5ll+fl4cdpBU28AFoIAhMHSBUu2Jclqmywo0WJFRE5dilW+dADlCZS0EGX2PgyHRYC
```

```
t+jlV0dNkKsyd73EiNa1GF7m1f/8MbLnICGTZssEAJ3tREDUvlEWQtjeaNuFAWIzoJDchp06g7Zlt2Nr5mUON/+
    YYeE6HkFqU8ti41JWcSJgUg1ZM4E2goUmjnin8Fl5eZVHYz17T4uUXl/905edv/PYa/qV8ie/sAQakrBAUZBBfKjEASh8qQkOT/+4
    JkOQATAknX6wwZ8DIEGqtgwjpMqP9np6BR0M0UqnGAiWDyQdAyiA4gZsVj/X+zLdERUpz9HSxhW4ME9Hf1Hjv1qsFO2NpJIlKSg+
    ROCDA8jCM8rkNNdfHqL88UjArZO5PtNQ03q0suaqOM0gxVqYpOhVyqSbr+9
    qJhlTs3nr7qQhRRbVUXG33Zs5ahCu9tXryuxDB8QIwA8RCV4AoQDW51tvXTNWSp8Am5BAQAm8F5s5JiwDBDeyp0Io/
    tOIxM4Qzb5xoE2xulN70QHLv7lMbd7D0+6
    DpO6sO2JgUpG2iiQCnKFYqJ0RRN01O6CoDozEounxPRaVnLxHdGep9nMcBGZ30Hoind11RxCfe81sq++sz9Vs0nm2u8pW/0vKzlRXapE/0
    XlFh5xUyVa6zobpiWkkswAE5XQCQL2GSjWyhpSIAtB0+Gyw6WkB/0v1eYh/79+66Sqi0Vj+lmwG5pgR+R6St0SPfQQnLm2mn/+4
    JkSIACx0lY6wwRZDMFmothghwK6KtprBhNENaVrHDwCpYSnLlgxCTNq8QgloDYZpq7+
    uTSrCwUCkYloSMpKKxuQKbZ2bgiF0O763ZK2VOF7ovpvQwVhSBr1yL4VDRjIrUecty3z/548g6dav/
    EKGTy4uNKgty6UQBJwbBzvCJWx3KMzIZqx1czkuTrE1bfQdxf
    ///elP3Auyp9+oCGOJOHSFizFaf6dRCiBb1ssaaIJTpcSxokaxlokjjcFgFihJKKmIQPKUDFPaYmvpg6CFNkupoFL4wYdSQreXH/Md4t7U2HxcJg8IqswCxQWaW
WnwQrQ/XemxLKFJag+GjrSQLd+1k4qAGYcCSK1GGEU6fC0dRuHZaGteIhRUsiLvaZQIT6NJtX
    //SzIA0dxwzGa6xCBEKEYpChs0KqI0iCU4AfiaHuWqRaiIaQwGM6V492UsQ8gS4QBgRleXQ1RMndNaEGCTMjMVqr/+4JkYQAC0yFbaeZDLDYFKyw9gjfKxINv55
M1TO3VxqbsXdPvbiARVyOKNchGAg8QMdunvuOvWSrLp7J7VrkK6+Q/
    UUOwvIJiwRSFxdErZxbaAVZHEBQVhDxxkKRnQpAwrHAwhYEspec3m7GIZHVfCJ/T0+19TspDykdNQ2eLe
    //7gmSNAALaMldrDBq0NMGbPD2GQ4t9MWunmKzQzBUqLZYIcR3cWwdBB61n0jSAGFwCCqNZQHhDQXjhG+PhItcz9eNDFluecYDVJXN5Lfg7epir9762DOLoiJlr
1BVbDsLCUZiHCmytHcCo2MBHLRUcyOFIYbt/mFOkNvcjaqQdg6rodzrBEG7U7G4ZmMdijbKDCJ
    //31zezN+lv/ou1fkWdDERlsCpHo/TevYkFNtf/7gGS6gAK9MtlrDBHUNgHrDT0iRIrpMWOnjFEQ0BTssYCZ/iAgAlgUcAhYAS44ioP8kzia7Za2aI1rfhxLk/r
YUqoEQRxxgyYqv/60D+TpRR6cr1F3/NLzt8R6r93/E9/PcXFTxvdxvG/ff9/P6X3SfqNcAn2xFAABwAJoGpyI1
    //uCZOqAA1ta1WsmK/JApTqaYYI4jJkvZ6wwaVD8FKv1hgkWyIaGGfr+dSBhYVQ163Ej846sGWzqeCNaZlOT7t/TlU79E6wjzrGMVGZx3uMLB0RgbySAQFJpCIF
0yl7LGamKyBAME1TLQB5R+moejkpx7giN6BCP33YzGLOlIVTpm0zyvNbqWFUIU
    //uAZJ+CA4Q12ldhIAAwQats5gwBDkz/ZmwwycC1CS1w9hhUcWoAJAAAAAXGfhkVDXnKAKiWPBLO4ba+27gwFAtWbn+tNmLBQH8YJAfKUsQRHmRScIx8nLCkHYr
ML2ehcfEJxEiMqnWhZh89lvrtWrvYzKY7nW43/+4Jkm4IDYT/b0eYbxi7lOxUwQ4oOlS9nR6RxwOkU7TT0iKijFp37dX6exoa3in7/2zO+
    jf4txQUrAAAABWxeEiivBeS0JQymvSORFXEceNP04HtRDc7LnsN7sl4QmZy61BN+
    aqhosqQshJTHCkpw4ClYEIfTR2FCEZCJeEz9kLfMKTEozuy3y0sfPI4n
    //S//P+q5buUW5qUIUFD5gOjH5HTKwE8AgBAWHOH5NN9MsfN/usX8hGdBghIKjNpVhWfcoTOufGGY0XWJ3A6OMPGlj35beuZqZAU4/40p8yqKH0IGaO5OMFYAgj
kqEg5QqR3dbv/7gmSmAAMUK9rR5hu0NOT61TzDOgw86XFHiNGQxpTscMCOMPnsTWGWdckFuZreW5KvyfyN/Da+TEObnAQYdpLI/RyDh60/
    yMQgFygAAAnBYfQLcvguTE4rJQbcYm7x8frrOtsaFSPXW7sexNYKWHXRBbmZXj2Eq/A3yBhVNWOXozEKehJDWePBMhfsd0Jbf/
    eoArkAAADAE4PED4UFEniHEjOzw5N3T06QHbpm8B52iSETbdXDC/jct9OwUirqWmZa7YJjJFKHsGuOAES9BrDBNx+3
    y4kQeSZbrEZkkKULw8E+LgU+ydvA6CYkIR/CVEHMmcXxfvXcBJD23U4L5Jy3ZNYZVdS08u4bBMZEqQ9gwPHI4dtuO1DTRdw/48MJ1/6
    qAqzAAAJKCIWMDOeJJoh7MqFSRU7FgMYL5htniNjsMwVqLmSMuucScY5NSOSEzGdyFSHsa2eOJjQiF0GDigwKByTvMFemtH/oHoVbSf/7
    gkS1gCIzK1pphhsASGVbKj2DHAkgq19HsGHBGxWsqPSM6ESEnLEL0PWyJeh1MpJojipWDnPEV/
    s32uNYdny1F9EZcTmhObT6IyGd1333VeOJNGmyiHG3dP6U1o2+upAXmFadOhFGpYDYoAuAvhPSKJaNd6hTGfasNoQh4ISZ0CmJ/
    CLIZu8ti2kKFanS/7RpnSspJgD8HNgsu3S+
    hux7VI6H0kMjUThliae0q9CAQoAAAJwHIZIeiShFsYfFcXCAjWiz2DE6ccOIoZlN0HNz1uS0mbrZd+1
    J9L7pgFKGZhZY2DI6DDJ2PqR73JIZHbs/
    aVoAGoshEgAN7gwjDAdxDRWotFmWFR2LhyFrli2FJiyU19QRA83QSPoe9v6kmTsc9DLlTyydcgIPUpkNfQXBRDegkVYH0pS1SNNKaUbNaRQ2JBS5NLcrQzyS0CC
    /cTKfQmBWb1aF6mtf/7gkTLAgJHKl↩
]
    dR5hqgSEVLajxihoi4p1tHpGkBEpTsKPYMsNZzmyNXGmjvaxTAQkPzOFXGqmOAs8BAAMm62YbH9IJUuqFujktqw2NMyee5DLKnvnL76SrLHsewjp8mGTjF7ekSp
    /Uh6NzJ7auw1Bp53361etn+nTVR7NRIEG3c7jJSBITWiqUfpeYRg4PdEEDLEC8chgawpCU9A5Eba8oo1POanf2wxZqmgpC8lLUjRVAp+
    fJLQWSUW3luXCtD8egBXWLdUV35XqSg2BiwStqPQEqCnMhlK9UKXieyaVq6/wsKFDwVnKmCMS1pWGVk7kE0c1mntIVspbRzBK5c30gFf/7
    gGTiAALCKthp7BnANOVauTwjmA2tI11MGLNBApUrdYYVUDa/DkAogrXGAFAwiiswiiAjjHvPklyySn09a6HhCrAQQCm5MYYJcW0P+
    RGw2fKBfJzalaW+ZYdjbCT1gbCMEJy86aY24kqiAbn8+nBs5KEKHD9uwSgU4/4
    Ge665upyErDx8UY1rtYinXd3VW5zOsYkOXRVfJGGDBXXd1zIeWa84bJx+uask/
    qnDf28qMZYZsIwTMsjppC8uxBbm8nTODO8BAwAYMcaYo1xKCVfiK7u6nIwcft9ZG1253VXczoUqAWlkggFNxVdVUAmLUqazauExnNb9g67WcRt2nnfyln3xiFiV
    +3pUrW
    //7pveLCAfFViiNZrbuS/tbDZ1r00BskAIAHITQOzwC4RgY0nGra1Npj+P9M5RWEOXbisERQnq2IAtU6//uCROiA4pMrVtHpGjBSBWqhYYZkinSrWUewY5↩
E4lWsNlg0gdyPZM3wbHJiQ1qqpfOjsH3YLNwlyw5tWDa1eTAq0LJJSbjSKBMlypBxiXEdCdYjpPQryZGOuWV8S8/
    hXYiIDuLapX659ajQPeLe89CHTmUSxJRQX9KqW6AmMxaNrIDAW
    //7q09jop7bt7p/MjpvatG6op19rVsn2yAA9gRxjUGe5IWutEkEQYOqDuF7i/VhUTGXsXi5kcb6dYBBa0p+VwXDVaAQMycqzJw76pVxXIdfs/2YPIYr8PqZoDf4
i/L5h363IcoHkDNEggcygC6ji67YNxYk4pShU9qSJTtynJGAglC3kxJGKw1OEXWaHh8SAYGEFkPZByDiC02KjFZiq5lu/Zvc77KYz/3BnHE3/
    ZVvb6yDxrTD3p0Gv75zflShtgFuCpGgBAGCAAHJY2OnCgmBIUwhUDpZgNThT1fTatMMxfRfyyI6C4Eam3K22rX8jpuJvQ0GyyqWgX8tkJFdrsWMzqAHJvfc4t2U
    /mPl6R5mCzLI0vrosKaLbpVlzdsD67jPPWM/VsEd/hVhlyiI+5696qvNTLQlXwtTYdPyEgv5TTWZIo2ipCRW7DHbu/
    n8PvrOmETZcGFrrxccNQ0va97SA+MOXocjZsQ96TtSyCagAMBxRF47INUTz5pK/FTC0mb/+4
    JE7QAC3ylT0wwp0FnFOqdl6DYJ1K1hR7BBkTqQa6mGFLIGQWbIwo5Lh1ObQdE0OBjF/NntcAdls/kHetwC8shp/hiWdrh/
    GyPDeYo3TdKI84ayXX0CD/k6F1AIq0QQC5LdC6RhQhExF83zl7axpT8NOfffq6/VgsOJJLsIIKMFRWIkgyPrIrYqGX7hiq2w1kyirb3M+
    V1t37KCtTzw8c+MRz7L/
    Z3PayaElcsoKCA5EknU93POFQiF1hAEbxCF2xWAhd5UXQJvTas2RiwyIjGIlXWb6lvYgxO6hb2IfIbOvicIt6nDkShdvbPAt4hyKPRyVa0CmViZy1CleyzO25Dm
    /YQqVOABB2tg+XSOaz0OwXxfB1RI7CuSSRXxjqt5I0PlX/+4
    Jk6IDzrDHTUy9KYEolWiJh5kwR8WtbVYSAGN6SaYKwUAGi531WBLo0VwHIPwsbFBftcV4gLt9MssEuB6IFXs↩
KH5b2p9FYWxs25qRU5ZzqMo5HppuVHG0K1oFs2mxFnxDjertcIQqosRV13I6Yrw4GY0k9ojmxvYdvaHTLx6o1h/vdKY+NQa619W19
    //OaVm3uv1n/+PH0zwbs/7+YSgrHJNS43ZIpDa43ggAITCImjfIWMUMPgoyBRCgj4oKoCtyKtVWaySUg4jaBrFgs8i6YUJfi7wVCc5w4rGJItp4yT6PqP7QJF5l
```

```
BAA9BShnp8MAgyhLqdL9LKtZfvoWYjI67a1q7Or/6GEjg4LPyYsv
 ↪ //1471gBKooAAvtjJsjRH7UL5GXLcuYlDtltEsLHp1J+e3e5u339xpeXzlzXZ5w5W4kdYbGH3SnjJtVNpzajIjZiHRhAwThguFBaRVA3awDkTBEjiovYIueuotE
jiW/UK48dW0BKGZCIgCVgBpNwYqiS8VzZMY0Q0W+AcyT6
 ↪ //9YUAjfP6ExXESpQfO0uTJSMx31JNyNpEEAAKUYMrlQ5eKlDE1LG5rxYOyWHn2hykisX2mUJBRRw9u0uWbMznYhESNqqjJSoSr2e86jiDWFaF2ZWZHKHkbOFAz
OsvS3Rns08kP73e6+miuhf/pOpTgL3qz/sn5kdd/+
 ↪ vx6xZbedDZ5p5cYfPtCg1LZGjrQEgDGYySAAwJAcJHylyEfDgN1VtSWVV2V320rdG6W0Wijabcuo JhNe0/
 ↪ GlBEIsj6dCPR6IltyNolEAl2jmMCwGROEIauH0lGgNycIR2YrRPM0ES1Xrm/
 ↪ vg1HTIMZG6qgtDgxWa5mqul2PTqTdNqh2O5oIzkSQtpenitQuERSXhLtAoueAjQAacb6lcFqRjlrjABAYoAABgDQNFBF5MV+
 ↪ qRn76uiU4aYpRdTKo35+12+1tcuiW0cQtRA+aakYGTDSX8q
 ↪ //uCZEcAMuYs2GmDFKQzJBp8PSIqC7z9V0wwp1C+j+lg9Iioc9S+9QAAAFOPE3RpqvkfkBQkdCESRBH8AodX47K0S4tCRer/891GKVGR5bHZbVL3dFQjqrBomOc
wfgQAiSANDzCOhZJhXofmfnBu+amkXtpFJ/ctFkS4wRS9t2lBzh2A0QhDLCdImegANjOqfKjiJIf8z/udh6GzeCCk+
 ↪ lyxI6ATFmSg20ooEyoWT3aratRwSQTYUjgDAUZIEerdRKxPj61/E0LEofu3c06yr5HjQjaMx1jdyfI4LCqHmr1e62qJ7/
 ↪ fRpohSbngWBqkD47Kg3CEPwJEMlvA6SAoGWrMMaWgz37WQ0t5dhus6OuD4nd2pzs7pKa06Q+rjHnPrCLiMq+
 ↪ z76hxDJuqZKjBUUFaKp29gudh9gfIGxTO2
 ↪ ///uAZHUAAvE3WGmGG8AwJVqcPCN+C9TlZaYYsQDHFKr0wIloEL/UwKKAECAE1sAmA25ZbYxOexn8przR/PK/+n///VwBGB5CZIIBAbTy//KNAdd6nHRRnMKSfe
YEIYW
 ↪ //J5l7nnkZL5JEZLGkWzjd3qBX/////8l9Fl/0ssaIC0vAwiKDYApyBLR8MQlTDwQVrA+HRkipRCB1SiBttn20/REKkJw/8FSlT82N76Ebi8k+kfNniUA0lFh//
NtFQroK+
 ↪ mKbcS6UyRGuNK1IAEAQAE9EyLvOhTM9e9ti0ZVWrVe4swGThMvmv0zyZaIN8pLqfJGE5hBk8hf0rDn7JBAAA7uEmAVSWPEJek6VSpQo6WQv4
 ↪ /T6nZVSDuOMIKHntUgfFGFg0drcyjXMTSzodUKcXaUMyFDQHFSxjOIijMA4qcGVv/Uc
 ↪ ////TTao8W9Wfp2RGMzDZgb2sE0R15vvN+BjsJ5S0pGS0iC5a1UIstZkH6sY85z4cea///7gmTMAALWX9pp4xREM0U6rTAjagtZC2usBFNQxJJq9PCJ8C/aZqp/
7OpUedCDVk0N9VIv4kslsxP+xl/ONo0h6pd69YN9Qcbxnxc+tfo6FY1pRjkapqKu/77eLZ1B19brrdP
 ↪ //O7Xy5pNRqNwYIaMpT+sLv42rW9oU52lUP8b6AJKU2yAEAZhQsAaL6lwEsdFtEvnqd1dEWikYf2AohZIMIAdZkw40d4yRKQ3jZBhsJptUVPRt2Z3lFWuDIVcZS
FiIiAHAAL+KSUz5WpR/E1uzWMNCcRm3H6
 ↪ //3+p1S+gGwGtYHDF1VBrq6YA1tsgAAAAOmPeCskmXZb9s8HurBsOtheyGJS+7qtZpIsAOTLmNUnaijSULTJOgNEbT0DEVpWPmEL4ELEdRpuOCyWFDCSR9ksfgi
vY+UtSlck9ph3VHgMBG1lhzAnPCacWW5gYEHKzYzJqkZUlP9rA1CkFASAfgN0URehc8sNLmO/zHW6Cl/
 ↪ pZnbZdrMbz5VV7MUzOyxhIwbn1OKAJlNUDSLl2MxTJCLvG6IUOo8BqFodx/BoCbCeE7x0SUysZ0g9b0MyadE1FQRVAb1AhXhW8
 ↪ //uAZBiAAwQw2XnsGjAyIZtcMewSi/Sna6ewatDPBGu496QgKkeCoYt8+AiAWpXjIgYcIFJUYiAVTDNMgY29Jqqg1WHYVY0+58sSKnA697TzPTlTH9tZH/MAF4E
A0WEoYbJCZAFjgeJlVHdENvX38eZThFyAUa8Md+9wSkbSUtQcPhcmEwaBRyxvelbB6zyF/FrW/
 ↪ TrrAi1gBDSxO9QM01pxoGlUVQ1BZABGCiUuTmDH37n9t1t8pyszwwCOp8ha09fvQEjm6bDxBEXW5YGX/+4
 ↪ JEQIACQiRV6wkY0EilWqhnBgQJKHtpp5hJMRIPbXTzCOquXSpFBAWEp0EaTr1Oi38uFNZJI0SQCnDDK8VUYuQ8jHPdKrZNCBxSIIgYpNqq34M8qVItzcze9
 ↪ /
 ↪ XHYioBnCTFIQZZ9dRSDvpl55Qq3TPLLSwovZypjtfRyoBsbcjaJJdtEjJMa8ZacjjT40eegocyJkEJTpqTfQGeyJZXmTM3vf1xxRdRc4hhZCDP
 ↪ /xL03F3gyMEbcyg8xVNf6/J/
 ↪ SgXK5JE0kU5c76UQ8aLkz1xr2f4MBpIcQLEY0enajYuLsTjfhVmr2ksdIXnKSdilIrF4UCEqpq44Pcs1ah2DhJV/3vtlaor/++hd1
 ↪ //9/RP/2VN5POYmrNVQeKQAM2aVAAASCmAJBK2lAYSTAc2Q3u2dy/VPQas66eY+6IdHNegk8wtGU1QSBOz/ej6FgAnBIhmKBAEm45sMcVtZert52+kbrVLMiyjD
szcU5E9xbDPh8+Lglf/7wJImkigSC3AIStuhfGyyZc1grUQkKBwhgxdEIQfcgzJfcZ1cGCb9HcvUR4QlmVSH0SOuR2WwSQ07/9
 ↪ ZRdoAchxbsJQGEWh9nUj7kOZFIZXERH/+4
 ↪ BkbYAC2CpVa0NEQC7EinxkAoYLeRdhrBhHGMYUq3zwibjgAZHDAD3v5ObJsAaKWAUADaL4AA2HEG4rBcC5MyWYl28ndKqPxkv
 ↪ ///mkL2lj28f0Z/fgzbmfLOcfuSoBKNtIlIAJODhVkkT3jaA1EYGg8kMSheaIaEWu6Ft1CjzGsyu1VMp1HT1ChdDMxaFXQ8fubTVzoph/dLJf/7LTKVRzibicXc
JHeICVTCfzxTG9YsSVKcn8n8PLQvc7XPzUePwL46P/+7+ai729Uo50gB0Vu/76B1UNzyyxSNF27DtbUNL4PB8luJoc86jVCq2E5qlIZv/7
 ↪ gkSfgAJJJlTrCRFgR8VKzWRGbgkgU1+npGcRHRUr6YSMqqeX/6455PidyXuvlBGujb6tp73qy93YWpz/j1Zr6dAj/
 ↪ JLDZFIo2W0dNbHqBu71EgBuQQXBBVnyhZkWG69wrozM0J0Z7M0UuYO5X9iFa2FcvbzhSufEzedL0/UWw9BjYGPhc5/xXtt/
 ↪ QI3v8lDaTYo2rR01kAE25G0kSDBIBtF3eqQmiWPJpSk71RIlWK9mXD9/JZ7CPR8MIhylAsFzK507a71pzvo63diZaIP/nWDNxZX
 ↪ //HSP/9ILgGA2mk4l+wsGSQVc1O/JqMJAQsgmkCWDxQYk+/32VHZ5Y1lsysozpX8QHnE7F1dL7B+VO3+04Fx2fLzMoDovwxMTxo35kRdyAdVAADjMLPMERIgpah
zmQgfIhyF3OUK+nVkvYSB0f/+PUro5UoR/vI3dfuUmry6zPpfPST1Z1fVet2HTopkfYw9R2ZAMbaYA
 ↪ //uCRMkAAlEqUctPMLBJYuttMSNjiRSnUawkQ8EdlSuphggyKAEoAog0uQ3V2yx/GmQuVsTk07D8G0+e8hAoS/87UBlC5L7fY0MhV392//6A7ZsOmIyhwxohMhs
F1aZIuS+0rsjlY3eXtT
 ↪ //iD49lADIAC1ChBAADBCoGAcM3DSTwH49C//uCZI2AA29D21dhAAQuwTtO54AADXljc6wYr1CjjCz8sA4YMb8/pkwbb/WCXDvSOXap253/wo7zJioBStpEkAAA
mECCSHFOR484QJyEjZYTnVwIbtMSn/+4JkmoADnz9YawwscDajC848Q0OOPQ9frCUPwOUW7HT0iHqr9WheQQ+5FS
 ↪ //Mlf/6m2Z2o6ct3k1VRb6bv////r//00Fy2yRJIkFzQRaBmM4aDs23ziajEfg51UlWxKtaVYI7aiQhbPOWdH24ibjfnTNEWohVWXjPLOD7qqdpnHfbzadtMPbc
j4ijv/+4BkogAC4Sza6ekaNDPBGv89iRALpKttrBhPUNQJKzz0jGiXR38UQHrYxO/GigsbUSE1K51ORnYVOKh82F+
 ↪ aYqHx3st3aPuelQbtttsaJAKdgEQCI5DqrFlERbMy0yIixE4gEE8vKSlpdrM1/P7aTC+W8kDUsfOi46AXid6fKPECzYr1Vpo2GfQm
 ↪ //9adykmygc2tslSIBBc4PC2BYUlouHA/DkbnjsRZGCQgFXSSzto6qinfNyo61ojf/lSn5WsQ9bpOAB0T6iIN5c72J6EbU/dNiaLWr1/0ZYFNKNJokApylUAcGI
U6lGDCSEuc/2+
 ↪ nmG5w0hVqMYYIMFZjZTfkzEeQMclFfRAp4GDhAQdZoEyy2NpolO7HYOAh7AtOBrK2CVyLP5So5Sqw8ZLyvZ498aRAYi0iBm5tNycFTvswjPp
 ↪ //34dQ+FqS5hSoI5/n/+LMqfWJDJWnTexj0E4TshJB8OMheFWqQYtCPpCRAPsgM2Q239KjlZnnfnxn+vm1zFBG/zAEcuoBQAEwBCsVOnBTiU2EEXcUIURlEjDSZ
YUWPX/6Wav7v/03DEBjOKHhRsf
 ↪ ///1uuYpYicbeSAcl0bjSAUloJBKCiYRbg5yGNKhPBUF0OFoFqYGJGkh6UucgaYdbbTKMTd1aodApCLQVUS3ebhNyLdDuJ92GR//2Elme1DNM5nNvo6PRDjUNCO
]
 ↪ /P1RQ70gV9qiAE24KWUrTOqhDZcME6QjBCeiocizQ0YLlzlx1w74gKbY5C0YZMaxVT3tFtP3cD2uykUqIWLbMigeHYgihrGhrtuvdN9Sq1oQEb6lLWSI21W/IoJ
 ↪ +t1aCqySTPlLsJYAzuVYxpxL7r3X6B2uhARv1LWSI21W/Sgk594bvvUD/+0iApJQhRb1rrNEcpVTLBWFUwrD6E+
 ↪ Ym2XgtZGYcBM3sXYcLTlLI92Rv3fVuSShKODwouA2/+f3qjfp/VH+JuF9ak2MzX+u3pJg
 ↪ //0iQDHIOiS/Yt7NaQjEksXfGxadHw+YKxKLrMo5UnM+Y7y7rPUOyPMxCtrUz1Vu//uARQgAAmlPWGnsEkROy7s9PSIci3StVUwxA5FwFat1h6RYSShKOFshXUF
```

```
HrNLhR+7hDuqPRnjz3gwJ9d5Vy+v/bV/FjRJ8xd
 ↪  //+OwRGSt2ePcIAyhcE6BCAJASABCExQAGdoWAthpbZcMwTYiHiI+YUSTLGXLvZKABSCRlwiHGHDpdCksoyCwF3CPqVAsMSV4Xt6r9++r63p+/Y3O715IomN80w
6Pw71Qi0W8VEeUKJo/jBrvp53spAqYXDm0jDlc9Iyox7Kc5P+
 ↪  ZspGNm8tnx4rbjulNXeyux1jbFSiMTKmKoQ5SVd2Ta3eYmqTLHFh6kXxXNedBQJpQgASAwENQEkUJ0Ky1ryfW2NcMimiCvpjmK/
 ↪  p8mwTApUKhUsxxH0f/8Kzqb0CqQAAALuN4R0kRpBdnkZLsnbYt0WjSS4+
 ↪  eRCJRFZhWlWkSJVfBCogMzTVWtmWqURypibGp7smE0Wu3xYYURV9SgdS1cVCc6EuiFx9dz3Ny4ZeucUsQXMFlTjUxiAMg8xbJ4LtyUYB0tOs7yRAArAIXz2JSqF
 ↪  /+4BkSgITYTzbUekbcCyCGwQ9IigMePVtR5hRwKcGbGD0mKL1HixXWaW8ADvgFMlBczLQRfdLgvztApxgPdJTrhlT7a9J0XRzoqt4WseKuNw
 ↪  +KIfGNi0rN2tMeNSxR0wuvmeEJzWP2Jf2/nNh8t1/VS/T7l7DxCwEHBe503PA6UqBqsXoyijKl7LXNWEMzKIay4Mg6B4oumQufNcDhk9P+
 ↪  qBJKBVI5JgAiNwRE/GHrWuAp7p/9FUhTtkgAAAp7oocp1GmMg1ixMKkBJGLQTklSgHjaaN/6OGs+
 ↪  jXwmCF9rvRVGEEGJCYwr9I3RIEBYYkg4QiyJGO5x9ggMeg8/IogC+
 ↪  aLFxK5medLLxzVLsy5IenMj2Zb2yy9FKgBlmPR0qauoWvVpd77YDPoBDSRDxRg0kSE6Vh8iqHAlH5+jpUT3H/kwL/W1rwGB0gdwq6tCvZtT+
 ↪  ggx1tIAu9kLWpUhnHWGcbioFwIUeKvUBxomP/7
 ↪  gmRbAgNnTVtp7BnwLyGcHD2DG40I1WunmFNAuAZsbDYYKMvNjMlqJBC6QHfSQWbCZrrlF0YwsgkyEOVN/
 ↪  bW5HSZjUWUgq51IpL56rE0xdkPKrczzyIYaXC5tRKtgkhkONET1IewQg9dgOu↩
R78WWuugAFOoIAgOjocCCw3qsnnrxgdmgEMnznLQ1qAUNqKVS50/0eLjp1z9qIkemV/IJVQLbiQAAABd+g4R/
 ↪  jtPZAkJN5IotISlpCBGBheuUWxdb3kg8AqG8BsHaIUQkPq1FQ3cnCGWetVnzI4OAGpEDWAohxKeF2IlhwVKzRkqGiAyf9Aic1E7LU6Ot1rsoiFKJNEAggKUJ8Ay
 ↪  /u/qdxF
 ↪  ///tIfpERZzzg6UNNEpt6SSnIiSSQSnNcQtdkhSKBOx4aZinCeR+GkcQSND4BW5XSK0ZNxNiCWJ3cPlOZMOqWxZ9qOjedNi2ep7vNcbx+P/7gmRmgBLmJtrp7Bp
6
 ↪  f2VjQah4sK1k1lXjBrYFTWMW8AgbvfSXYVvNfoAG5ERIAbgpAOyjC7LihKGEkBAQlYMeOh8D194mKInYRc7yaHZggSeWckr2vaBXQD9xWEHhOhQBOgAwmiBGlqy
 ↪  /GUM8YBcUOjJYDCq8mHkRRoJFk5ZACU7NRpE+3x5nZ7I/
 ↪  Umd8LlJBBcelm8tT61mFqsYdFSLQuJg8GSQs80d1ulhZghElHj6NEqUU5W2yACzZ/DML+cdTQgLtjYyYCtgYxi/KltX6w9Blv9Sj3/6e
 ↪  ///+jFjzlusjIHOqTLKgCBAAAHIyQTEwgwSzF6//uCZJWCAs4o2FHsMUA2Ymq5PYI4CxihWywwyMDNBy1w9glWsugVSx7BslAw4ej4ObiQzRQuwm55V+264n9nX
6Boeb6tTRSm1bfy3YokBnlCcsLkOVjnLZf6SxeJUh1aSkZnKrP8JQYhL/
 ↪  BIsIsAgEGCniR9BiSZ35GtOg572LqbQZZiOqGpBUrkKYR1noLBKUCSGLPJzh5SS0kMConQjWRqPRbPOjKhFJyPmqdFVpkdzMzLcrdzBrUKgAwAAAAKcHDpGpaIg
 ↪  +bin3ShprSMdsdVuU6J6bwVaQskNWl6nGCyn+n+hH/2L1DFf+OBJcSQVSDBUcgP
 ↪  //uCZMAAgr0/WVHjFHQ4YipVYegcCsipW0wNEYDblOjBhhTohOSFqw1k8CAE1dC1JCYUzPsIdLyIKJPEweeDoTJiwJBHSwv1pIf/b/x8ZvPbUF+VJICbdL6N8Rh
jxxiABnyQUgZQF2SYp3wUHwZHQpHBBxUwTSeFR+5DzRio90/KVkrQUGZYa5n3z2hSp
 ↪  //xbf/9v8V+0VeZgZB+mRJ6QkABJwRUGwC9P0E+ZR+bIaZCpTx6lxTb0+IgxgfgjNI3ZoeyBaDrw8Y2XGsafMEee3t9tLxyYlqk05wh7ZwcYdv/92Wtqi16t/t/
PG92wqpTxKrK9/P+oxSAol1OsZrqZavLPavo2SRAmu8uuH11IRXaLoEDGxm2pdCSrTRBAALlotIAAE+N0vJDRongOIkZwoT15MN65ORkZHy6v/
 ↪  M9fqxLmeaq4rnex19+I419vGEOJXB0VbQj444fMXisNTcdg8VrzOEAkbVS0Vyg5f/289
 ↪  tIX9FlrKuiAGPaCsZ7aEL6DQWmx8GRlPbpoN3Lqygg4IQsBmIbnfYT+3X/vY+
 ↪  //X/ZD/+4Bk7IATZ0JU6eZUYEMk2ic9I0YMjSVfp4xS2O0VqEDxpiDu3dsMq+OAfSAJANwwLiOFKJR3cLA5aMkd2Ymn7PN9/FRI1RlYr12z+yrz/d9J3qL+uJbJ
vyCnEfLdFY4z4hD1Vqx31VnUBqm3/8R7OOoUO7EqbRX7c5Yzv2ZnCRUP4/ffbc1d78f/H/pAmkYAJNewJvqZOev/9VV4K3KAAAAPpwW8uA+
 ↪  QX5KUae5OcuCQNPSyjXkhQLKUoiP/sSckx1YJWypS3hiQiUDwJGhMaUImsjGSzaA6AAUkYiIU0MBT+
 ↪  kxsVCMFkCyF7eFgGfaBhpZaN0jyWsREMZb
 ↪  //7gmTUgAYbXllWYeAGhcb6z8y8AA807Wdc9IAAzwZtM55gAIb/4IoiUKxyXvQd30R57DkV9QbcLRWZunhVMx2tt8hAEcnBBxvGohx5KRDoUdJnJvca+SyN+5DJ
AuSCYVVzpgdpqUTNJ4MZJDhLTEqjpiTDPbRr6CcWHdgfo4fFfXHHaL1L7KSX1v/Of2Uet0
 ↪  //xLB7kaZRanfk901fVAoogDS0SQpJQJh7qtCiJVBQmicUEAcS0WIkAnE+tnDLsKXv7Kr2uIKJiilmU1gmFjIONiRwHriPk//////91IEqCAFRN4uODxCAZZBZS
7pmL9GM/xEy0XrO/yVQaZkkAAqVEJ4FQUEBZoNsnbONwVTU/wgA5SSUQlDVXarwn2Gw4iToE51PaDmvpa8e2uH/+
 ↪  zj5YpR1QrGaP5w7cNF69dNrLbP3/ZLzW+dYFUf43lPjkctN/M8xC/ueaR6VqAK8RoUgJmqeIXqH1WCgRSB
 ↪  //uCZDkAA4lEWNIPMPQ34hrZPShUDV0/X0wwS1DOCKrs85lAURqwIBCHYD5OUJU3i9ORMIebUtc+Gg2y0RMXTgI/a8KHYVcGVIIKODWD46GySa86ohIRV+S/6gG
by1llD3ZZUs1CB6mYz1fWtC1re+99NCoValLN1e/nl/9+j6HGllG/MEFBXGi6HLXo7p2+
 ↪  hGdWzCQEAEiBsIgHKYpCnxbWM5SwfAjrDxVyweJ4zFmEG1qgYllI1U4wMDiWtbUKFT0UV9v+00BD/+4JkSAASvj/
 ↪  Y0wwSZDgC6to9I0cLONlbTDxlgNcKahWHmKgBQIAIBd2oxEkXIYRBi2cnCWRLjtPtTkOuilsmuJ0U21lfM0VxZA0QFCkPqn0kq9sz8i7I4YWbBq5g1w9RAvECnJ
 ↪  /5Soyqjxz3RQRuToxOre/S+v/+TAEh6SxANWABhiFjGUVZLS/
 ↪  JRgF7pS6HleKicR4wJXCHZ7Lm8fMw5yQdoUtNhIlLHD0iLWhO9UACAAAAFds1bQoMvkjnEZA7kHvlCoNj1PDsueeteks9vupTjTRCJqG/
 ↪  TP0QIplUt0bfqzklod5kq5JGxoZE3YzN7l36kbo1bdbKV0ZjkuRUYTK9zjRcUE1IHy48DEnBkBrEhdH/
 ↪  iZ4CmLJBASARxOgkwhoVjYMh3FoZGpSOYC0iDzo0hrRSQebWbhzQuTEa4nHedYIrlKZppIq
 ↪  //ff3klJKJEAgkpx+YZ0lyL3DMg/yvoyIkmjd1wGAk7/+4JkXQADNUnXUwMs0DdCGro9iCoKpStnp4RTENoHqzD3mDAa4cPe0CiZvKRzVeNGZRtOS+iRHORHFmE
lwjASVCj/Z
 ↪  ///+cTpajWr6iiAUnDdVY+VGOraNbeinq4a2A/D7WrUXTTBMwQwQSYza8kYUTMRa1jYdKRd71QlrUazKX1OMJRuneiGEYeWfSOPPnUjErQvHpSf/+4JkbQACpSr
4RP8zn/8ZKRHhM5XR1n9HzEXf
 ↪  //p4zo6v//fCpImxFyJooEgASfhmFZNAyRx2H8cR+L4SBD+AB5XStPxq7R2sytcuZfMdJV//7gmSdAAKLKNbphiwwNKNaijEiSApxB1unhFMA3hSsdMCKMvLmZI
S4U5
 ↪  ///PZGp1t/7O7OrZsw5Ete/Kuznu9rpbPR+qaLMNJDPpekiG7WIEIBDEwaNZCZiCrcIv/7gETPgAJIKVTRgh2gSuUbDT0jSIjspUbHjZEBIBSrNJKmKJD09OZxS
kU4x6mxGXlZ5/4xyBwRrvkACYAIAAUABAwBlGBBU0q3Fuz
 ↪  //uCZIwAA01KYG89AA4vJarI54gADbFbZawwaoDHFKu08Ik4ouYyAYf/mOpqd/XXknYnUgdP3Xkg8Plhq7vf///15CsCSJZDMQIAACmAIpaMNAjIMIIkSUySEmg
EZXRiEloHAu2d0fD5CQffe
 ↪  //uAZJoAA2Q32esGG/I04SsdPSwCDeEzX0wZDcDQBW40xI2CutvpC6E9///I9OKZUe8avQoBS+StEggF3mODSyWgtL4fBYKjpSHJ6MiAT2E52pO6UQ6JvrCj3YM
```

```
JksQACtilY+ekaIDQByww8I2SLjKVjp4jPgN2UqvWHlBgsWJAYtLrZEkCUgViUJyIJzhAhCcR44ITZtKdH7T7cJJV+pHCmhS+i+
 /VzpfEjoYqzvW3TqQLW1ztmueFR99qsCx7iaLFpYsWAKRgDDN4EHIIN5GHoisxbn5TWg0ATUOMdKbheUwrj3lNu/jxCOE3W+
 dni8ei3c2zIiHA+nigAd319oc1DU400EcX/lv/6wW++SJAbkHwD5Hs0Ni8JSDROQlqGQRPRJjeWMLtgUnFXrpFJA8qo+HiwuVVteAGV/
 DChw4KKNyzOL8rerwK1zFaP/0IAtuNpIkgtyl8IwRh+o0w5nVFaV01w0dUcWdKsIVd5NNVRFOWmjovp/VHQ/XpyWsO3/6
 V2Qz1QFrd7Yq337i9V1CP7lFDigEAWpXooSALf2ACOhILMnig9H6JUUtNy1R5b6KLljO5bZ9URp/efB4v8T
 //qPD8lSwH/+4BExoACLinWawkQ4EWFO70wZYWIwKlRDAzLyQ2FrOj0mJJfQxEIW7/h2Ka3O6zgq1F9+4v3UI/90sADLJGiSAZdziIAJFEMeX7QBuHJHJhcFrwr
pwpVOsMGVBR5Us9PGWIlKzVEvW7pPoWHwgABh8kuNC2vpZZ3160KtIgv/6UiHJKUg+
 SZB6Lip0EznCsKrbahp90er6vo4zUgKFWol9TP7JsUHQ5TKYqa1P7ozL9/h+Vk1wMIBgAYXC4heRahREc785DHr9n/hse/
 tMgn9cUQFJK1EgeqkVjgGVlE/A6GBKjORJGTUKdfztMYJp7mbHXOfdmR2Y1Zzm1dutT/2ZbpuvgXFBQ+6NRvPU84i8jHliI77EHIYjL1
 //86PfbaZqAKkaTSRJUlpJgtCIESP5dmsqRynOqzkbUKYrMG/ee8WzVueJVEF/Al6sM50/QvuGK+p2fc5zjO+YLEO+t//n1/s7vve0cheXohB7Z63MdNr/qDtMQ
esaZAIXDhRzngexRj36lEpn/2I+uwAC+puFoAu7jhQENN0rSA+IoMEgOicPJcgOYjHhROMHRjUDJb3MzgcZ2LTROakPPnb3noyHH8ipZ2bZ/5
 //jjtDXVEtBMVYH3ANarb1JBEYxlX0U/JgJy1yNxIqS49QU6JI8k6lDwchmG9ASJyNytkb0jXDA6BLZv49vlbsFqs7P/SW6NSpa0JIZHQ/a1yvNd//FOjaoxVE6
O3C9ZJ5Q/Uo2AA0oSCkAnuEhzHNEFbLVtN2X+0t93/fmP9rR6A5hyqGWWeMoR0VyMhXdQkyalpImLaH+pZ547z1MvovUCgbHl/16TOhe3/
 MPQxDFoOES7HB0uywmGkmQBpVJIALkZWf65BLIVOlHn/ZXBcRZy2O3TN2A0CQGgrKky2nY2Um28it9PjJRIOiW+KPOI86kvodaxQXIf8j/7
 n2a9JcUcHS7LGhpKd6XuWHShyqgT6WJJAcgWHIgEj0ISA6LV4p0crUogdzI53aM/jynNHQQHMKQk6Ntlyvq9x+bKyQ3ZqNv1UGPb/+i6BDyHq8HWIHCwus6GlIj
 9+19tX9KOtB2AJUgkgF7UvSDqwZONyA5CWpsvM0xiTrRh7KWhrP1gQUGRtJBOIrI7p42x4/mGzSb7uk1/5j0/8i774hWWt3vbL5BJaLf
 /////9tfTnF3+owlQnNuj1KhuLQ5yUql20bYAU5HLG2QU3ZRRnaYh0QCnVSwjjyViAZV+EzaDg4FFRcw4eQiFJi7I6WZKOKmAdXO4uP+nUTfsa7r3WHBZ2//0dX
48WeJDtE3
 ///////////////ltQ928gMjuJSG/ZyS4ZZoXBaZBGJDmRQBAOJbwlhV5pVAxdinAeUNbl+VPM153WYOocGEDcL5qutSymee5omPAuEoOwcZTYvqY0JQ8ZuMCcM
OZ2ms97mzpxYi/AjlUWuhEeFVAVPnM2mQFotfgofGDjqzo0dVdoWYBdYAjvAIAOIHMVI25UwKWx/N4UIFMCQ0MVmIjMBKtncjIFT+2vSm4M5I+3
 VcGVntYA5QymQgAQAVeeWAQCqizWetwYpJYhBD/O4RgiBgqITaqq7dsuUnkmSBzJzbVCIElc0RCAuHsZ+AYemYQhE3atSPYfv047K2iEIz/
 BcrRMVz29n/mlT8+wVFEx5R1e+/v36l39H9//2KMsuqsiyRFWsJRi+E2HqEuiAExCU0o8NTFIwYATY+9N83CEZa7O5vYRV3ML/w+
 zrn8AzusaIAABJo1UDBTsb1jC1GauQ38fc2nikFwVHb/+4JkpgATKjdY6wYb8DBkStw8AkIM+K1h7CRqyMaQrrzxiX/
 FKwWiElET9PxWvBSMpTM3h9Pu8n31oN33DfWScBzkhQwCGFqmEnKCarOV7LhG1HGBgHFmXnnggLSD2hkIKeve13d/
 rGlyZytv8AQfHMJyhS5UYqjrJwWDHv5YjbH7uiP4VjCo0R0XMtNMOJnecfBYNf
 //+li35cF1pxIkgAAzHkEIB6I1BidG6c8VHPtzp6Rpbn6BKCVIw53a1wclc7C4uezT53Wbaffzju2GhFJEwk4DFkT6nFIxer5odLy6CDQrnTD2SpNlQmPLyOWGO
IBMUiKRJBKdQAEBFgIHhUkfOPNMl0VvylsDzNItO2LCBZ/Wlme/9
 GSosu6BZNnmpaUzyql0BjnP1eyt9oqGgolB0HbXlb2SjYpwPmlIYxWoTWV51XKi2xd0BgJQVxQggAjBnZBVMttDVKkpD3C9bkhslnOCyNQMVkiCavEYCtGVSxRZ
 +/+3//bX6aAEOodWNG0Qn+CQAvRgJhWx8uCqIiXoMIES2LbTjD1ocEqE+
 Ra0MVV2Syevhv07ecU4lO5CLRdKJXqakkcmgWGpTSw8xtKwEHo6lEAArgmgairIVqJKEMzvlBW3DxxdBSB+
 kRh60OCoMYhP/7gmTEAALBJ9hp5hsgNML7DzzCOAs4rWOsGG7Q24VqtYekSFkB1oZlXpZPXw27sQ12a+cUDI3U6xbbp12IY8ZSVDl+
 pSl72gJNxJIlAEpwaengyhcbTodalDhRYHIwPQfPfR0Ys3dqCuBU6lZU1qYhsoshrMiFf7JiOAWSuVFw8pBVreEzv5nO0I/Ijuv/s/
 WoEfqAADATubtIICLcBR9T8YJxwsgiC7Lx9hSk7KGLEcqtf3PtUerM3VRHf8uaQ48HXqjRUSP3ArXzNTJ7mGQn4lEv6Hniu5zNVQpNfbY0iEW7GVStgKDrbVTUF
 +FQa84FAbLKF+
 RLpCJD8dUlGXIZd3KqtMLsO0Qzp9kaM5USr32kAEIqOMOQPDUNVVFzoy1Ucz1Xt0nIVLqOKk2KnRGDUkBg8iVrMKinYgBEaZgAoBam0riRAJDj4No
 /SOKokBtRTeVxxJxGK5GMqci1k8XDFdbawURowf/7gkTaAAIwKdh56RBwR+Uq/WGCHAk8hVusMKdRIpVq5YSg4Hn/2ea7VbCy9ugUCva31PN
 /b//DEVd09P8PTZtJYByWy1ttFu7EtBaNxFjYF1RwNkzzjLkYLs7k+pHzk4qPc+np2tBOSkJVNghYJBZWCqLhursXO9LBO1/f2VAMZRLO6Z↩
Fq6P6Kr0p+rEIVjqVxgQNW24ceKwqlRQRt2sfHpaxSgMFLdtqBIy3INnOuSsM8mRqjCRSAOldR1LAUanWRJJMT/+
 sacimUqCmRCe9nPcIdvb3SKqK2fyxBa+x+HDqKJa1dCgE7LLG0kW5cIWIW9I+IdpeBsG6Y6RQ9s5+Ix27ZTqQ030fPK7CApGUggje1/+
 Xv7a3OI3wgD3PnfWdpiRbgIWU+z85mG7FhLMe6h8AhJm9nuB+CseG9ky3MP5/+tuv+
 CM9YNTYyhOT5OUNIpOCg3EcFjQmRLx3nQ2GoWMlwQx6ZBQDCYl4oif/7
 gGTugANeQFtrCStMRMUrDTwji4x9AWenjLLRBBTtdPCVzlPfStIHcGATFnHohDgCCqvp1h5XpyjlKWcL1Ld5f6QY7rY220U5aZ4/
 EWNcpRbRqG/YcRBl21E9TyCME5pmSFHaC7sbUW0yOzxuinVuYFxu0QtyoP7oe1E1VJNVBIDcI+
 m31mUtDId7CtH3IqQTUQKc7XYlqvdnHnYDMORFalOlLLn2oAZvqdKkANACqB96vgeDaZEp+
 IPuy06SLhZVggns7yNqZSv6JdyvI4IwJiwSFPSs6oVXFGkG/rrDQYSdWYoAYodWZDaAATvHomtaW5MZf8eVa9g1C0eBoN/Mzcuk+
 pwyeBlHQ2OLAE47CRiO6hUBd+nM7arUjYjQUWjBXjygyECHOnf/56VH8qWfaIPE8yf/eBYzuLss++fsPPRr6Ja/juf9p9b/
 VeiADF3WsARAF4WcerLADxXllzV54FUaEQmA+KHLYcbW
 //uCZOgAAykr2envGPZAoPstPewTjKEva6egURDzC+o1hJUgbuTrYrMn/9Dg1SCRl7F0UEG2Af///sEph7NuqYDCTjgHLdY4kgDJeXzARFg1vS14laW5cbV+ZdA
VT58XIFQqKGChwctq1eXewPXvHBRN1iU6EIaBDyeChRh8oNhutWgpZfTfjxRDjEDmQY35UYzNDG5d3OOl+zkl
 //zmBBKz////uaM9dlQFO061EQBJMwYwlTtZtTs8Vyz9qrOLj0Qp+cJLSdmaKvOjOJtbY2BqHratu0OTZV3K2qbL+9tZ7+GAa1SDxHFHoXgaLBhXLX+2VNZFRUI
p0QAACCV3wWuOhLjSDiGK44NLJvUUWvsuxff2R2P7dX691RL8lSdnVzhMo2qCYJgoUKf2P/v
 //5b//+qoBmNyNpEAtyg2bBUMAoC82aPQ/MlCk/Px8Do0ZVM1VHEggkTOyN27v7KecOwv79OlmREF9UAjn8XFr//lmN6UBOIwg8Mkx5VHLJpNh7CJ02TFJZZG2C
XBigAYORhEgucLt7081b+3375nXd+
 //otYljfv2VNuzK0bbjSIdkGiX0UAGK1RkgAe7rDHrIFAidBmrVVQNdbdfrYa7DqjL4aa05gTFUogiUtrGio1IiI4aGU2u1g2cqUX/eaTZ1qZD2BdlsK9ish6AC
V7n4ira7u7RZIePnLXFT8eP9QVWdCCsx4pPZc2vK/non0PZ8U1ZxR8+3zNnLa3sVcbrF1Mxs7/feKyJV/DcXONHiRpn8tN+bwtx/vWY87zxM+
 Pnce9/4DVLrUXe47zEsb97fOtztPiIBAEmogjAUEgECE0lkG9iRDl1k7V0u6VgsopFtNnXIwycZS3k6+ggsAZg7wbEaF00N3IkTpkYv/7
 gmR9AAYZYdxuZeAGhyhq7czEAI5RO21dgwAIvYZt95ggBKK5ERzg+
 AXAgmZsmcRYY8PXFxkeMqtJJSRIFwmi8m6CCGaKtnVL1K7amTa2zHFspI0kwRQqLfSNGoo5xAxVu9yYNC4gmTZPjjqPva6xyfr6QYuYAABL2dozhVgf9gGbD4Ee
 +4yWHpZ26Tu6sfs
 //n5M/5/2+Tvby97//97W/v/vfyOv7KB7Y/MEpgN9fcvVvr3947jvmEigfzCYIEAEQ4A3GgE1LTw6oezdmy1amYBCXQO7ggLC59bPQxfb//4HIgcO3upG1QiAAE
qKW2R7VAEn9FxPyTuMkHMJj5ELVoTNVT/GX/qCwIGeuhA6u8o+
 UDra4fbiJVQKEpsAAAABzTC0l6M4tg3CcrzeYCFsSMNw9AIAuQ8J3IkTcmVLWb8i/
 MoC0S9smYximQkMloKyQocnGwo3SKKdzgQWMxl0SVFmHq6qa71kSsf1uB/9/ieXN6MppkOUI
 //7gmQvggOxWdrp6RtyM0J7GzDDOAzow2tHsMOAuIXsYLeYGIc4vo2ZnzPsJV+w///6STxce1Z4yRj+E5hJJBGF5HFicPy2SoWFjC0nnUOEhC5JKd6WU2ewJHDh
```

```
yfhb12RHDSg03UMpGi3aWTy7RX3Izpc6WzO0VqL4HVlUN8ZfxoUg1rtv9LfPsmgINGlUYBc7vKFIsEbyT44e6SiFV73/8
    xJ4AMo8py6o1SKieajayNSl1crjZ0VCnfTSGXs3Pny6k0QjcqOPAkmBxIMahlwdFtF6AV
    //uCZEeAMt8sWmnpMxAxQ/rlPSMoC8SrZ6ekyoDLEmtg8Zl4abAIABTuJ4OdMmETZhL6bhekLTDMiMEojLBM4iU1krW8NXWanaeTOmHFcpnqseKP83ON+4VbJvW
r5B/TBeb/c/UtG5pUSSU4G8oDJY02zHonV98EAQaKCQhJySJV0Vyo3XeripdVpNr6sXZqO9CF01shVQmRK/dfIj0Vei/fbXUQQVjGfS+
    eFRYTC4QAcgl8AOFgakn2fx2S40JCgRYaJ
    //uARHSAIh4pWOnsEGBHRUsaPSM4CSz7Z0eYSZETlSvZhIzwQYfMtFScjlKKbsb1DV6/NtKX9JaRThjwIivsyhyKKoJMggLDwVQPuBCxsV+uj//F1RCEFGFCoTI
P3uTK3Z4xvcnJZ1zOEV/ZiPOZ359L7M5FMvnghYLkn/oWqz/7qV0ah7nDvt9D/
    gBUFtaiUCm5U4WQ6TREo9TE4jiKEDBp3K1MuFXPIgy0jEvkudk7/+4JEoYICPCrVEwkyQEhFWyo8YnoIzJFZp4RxASCT7Cj2DHiU+
    IDJAUoJf
    //2LU0Ci0hc1zBnj7qNwh/9Nam/15zq/rHRXQA0FiaJQBklQwHwEpLaXDKOTwuCIkge+GxzYnYfMJorcSexbqmrJoiyIihm/6NXU6fXV7//7KhX7fr7JW/WCbuf
GKgQ2YexXXKX6ewLjQBQAYEdJdDsAUgTSuuDk32oOUMmiycHZRBm+8hv9eqHHmXtNiYfS81FtWxRX6UPbuv7bmO13/+4
    BEzIACSinTMwkyIEfFOqZlJkYI5KdVR6RlQR6U7GjzCZJz4+I3gYqAJsk9ipTxL+
    qSACQIS7S1FBZgAUPQbZ2WbLMcsUDRGcIxUDEGDknpWol1HSUUD1bKcMgYgNMsuOuyeYcnRKL/H8mXW0ncELJflj5wMRX5VZpp+
    y3UkwMxyKWEV65pjIarrGAABb/ESCQkkyODqMeflqY8Bi4sQi3ap8TKOJGo7ahNdSuMMEKn28zKkChKabOpsWThycoDF3h+MKXC2rXBDkN+
    WfDEV/vFz9nTSYGViY0lLDmtc0zdXXSqW6rpIkEpxDyBBpDkMSEvFsQDG0ZW9cYA7Ig+
    F5y5zMeSJKaxZZe5npjdLxm7Oq4jopV5NFQEOKHT8rsexfhZD0pTXXlSjFNiEMqSKuallZXUcOVgot0ykVLcxj2CuGAJoyOk4lJRax/
    MHBuZKF4T8k8zIXhKRDqzL1X7I7KRcVb0VcCGWisq5P/7
    gkThAoIcK1KLDzFERYVapmGGTAr4p01MJGmBYZVqnYYNMNKDB5h6fldjxRfhZEukmlcbV7XkCOx1J7ApfDAQhcvEoAAEt5ebiChBxTQqhqFCITDVsfrKVpWorWw
    /PioDsbZVEpnYiE7s+lCnqQJMexVAcklompTjynB8Ph8QPf/z5+XKOEDcQAm8oc
    //0450eCC7JJISbrWHcF4EAIcTRxORYTkVvR1miGinBzeY9y1+8ZoTrHBnJUx0hk9j/KrqwSZ+U5PrnHlLYPh8PiB7/+PPy5RxRuUBN5Q5/5m6RpLOjy1UAsAAA
zxNRCRmGFPHdUi1UrkqkyWYTkq9Q+QzDzsaiv9XauYWVOV9Qbs2JT7pVkaXYjeQv/7
    gkTpAAKbKtjR5hNcU8VbKj0lTIo4p07sMKWBSRTr6PQWGteVWvkLlnnrYyY+
    UAEGAAbxr4quEsF9TGMZe09ZbkMzX46Evxh3co6FmkHzAHPfXrAYzCjzN7wzOVm+q2qmXr7+/yp7ETrZ+
    hCSPws4SYg9e9b1LAS/kACk2zEWkyKQiEA0IjY0dE8m7KYFxVMQuCXIso3nVIIAoaxZj7QIu94sBJgjhwEsXbCu0MNMZjkSkNwN0sZ8KFv2jMNTChrCeShkH2wE
    +99fyqxjnrA3/n+Je8be75+
    //8bvekTb/eoFNf409GL/UCxO22uw6Kx5CIUjsdMADn5GC/q+ysEZEBi8ElmgcMKsNTT2bjTwq9DJ9nSb5p35P/7gGToAASMXVRVZSACS4TKWayYABiNgWW5h4A
2WAKZYId
    //IUKBq3987/zDuGWhiYJIGP6Zov0O//uCZHkAA+Y7We89IAArAWtM55gADW0/dawwaREQrSwo8Io6CxtC+/DiCISq9KezEMtnzYsobUVmc6WJc5PeehsYBtDVg
qqR2o1HozEaWWR
    //uCZGGCA5ZLWOsGQ+QyYLrsPWwADlU/Y6wkcdCxkG6w8AnWeKNECBS4ycutDH5nh6XlVGllIlPmTzzhDTRA8Hj9pV5zIs1icuksp0VQ6VjdT/800GJm0CU/Isv
bZ6ewx1DPA2v8lKRIKrKlhp7EI0NGErTTDJQ610A0NfYrZS7dNAF82LTWz2Q2hqS6ViAkgACvgGNoyESFAkh8SCAuGpgKoZHd/Ukh+
    sceYSYhMRIOb
    ///////Xarc4VHLUtSopqVtJIkAFyIEdISZPiasaMaiZlIWF8qWiyhW2UOSPLYhSclWpwg83Gd7+7j75CPcTtdK6gnBAV9K9EUhWc7vqv6dPIyUtb3o+yqO8FkM
xR0MsUaNWGDFF4p0EovShBVOECMzq8JBrC7mR6EJc6IDB2o9+q5qcl58/JyzqdHsxAk9hD2TIlrbLy/
    V9CMgOZHoCjum1tHpRBkajdKAhDJqrmSJ70RrLCZlNu4EHZoWouGgNwTHj1Pp/PNC4RIMjlO
    //+Thh9kichhtOR1pEkhJSHUGuTs4jLVSPqpHJDV9GG+fiGwS/2YGSAIJ6qtSztKBky0KohTo+rSruqM/iKM9PYoEAFczVRiXQu0+pcr9F/PVu07f2R1odgyBsc
]
    ↩
Hn0RPX7WSMgCXc1CEm0gTUjEthq00CQLktw72cQEAgVFKRZjEMy6iCacwvO2G71acazOdUUxk6lTiXEnEA+VgoUKghCQRExEff/5H6f
    ///Vbafem66jKDzsdJZjMVs62z6IR9qtsNUtFY5rrta2yDtg0CQSIR3mjLBGBtIhECBNeaKE4zm31JzyoNqpPbagCQfyPhl3sM6fbhSAc/S7//p/5RzPWoimEEA
h1HljMiPvmTN8f41SlP/TNL3vjxNTzf9/NIyKu7+
    JuIzspQxmYmBqwu4Slkan4gMCCFoRKgkoDFGUMhQaKp1jrGh6BYjbzw7npuBICA5qyQRamnhejgzEQLZKRhOnvPkjsRsRnMPxFk5hAp5QRJM
    +jq7GNMYFwely7x4KwFQ+ZOPLMTkhkqx6U5MYxJv389
    ////nGMrmOx5J/////MdTyTBoMAElMsiZKRAAAAABRMlMJuAQwzHsv0ftcNGzRjEeUJE2DgCijqtldBYq5WfpYtUwTgl7WEnNwEuznQi8UR1QrkOMlRk/P9+p0w
cn5CXtknIgUVHA60IRUqSDi72ip8IihLIn2BlyAyvpfMYQLMlbcb/WKgUdS0R0dEyuJIAy2/
    RMhB01MwjPrqloEEhMeilEjRLVTdP2H9xZUkjlM5nqEJTl068oWkwESsq0MXITnpaFECk3iocOgSpLAAIEodDgCguPicPZSSpyF2ACMgNlfv3
    +OVSTZ3WI33EO2ZqqCOihQAAAACoLBLUqZwGtKAMScYDdMRXhGHI1K9ESRISFkii58lRYqU0GuRZRDtUpHcxZqRDt+
    SmbJkZOp0CAOkemcgdjnycn0ZZJpmZzedCzrLRRuEfksbjRuwi/T1K8jIvx3PLJ9FId767ezhnP/Iu7Wyv/+4
    JkOgIDgj1X6yww0jMEO5wwIonPhTNf7KRzwPOQrDD0iJgZ6gBOMgOMSG6CgGYeF9Bddr9J9ay0AJ/zYBn/0caEj7Fd6Gh+
    YZwji8yJv3bo10CCERDAgADxwpgo1JNuMsg5p6lL4M4gGCYeb1+
    r9mPWspLlcppVYzx2eTbbJeofOMzjO6ioftptRZB2GV4FZVoXdCZ1fDMjDAh2MLQyQu1hHD5AvkNjT6uL9OtTydJzyntbn7ps3yBt7vbGW9Q4
    /qkCqxR5cUfa9oA2vsMhz7dqBpGgE0HEMxuWz7ecOicNERKHStFKoLKV/2a+
    iJQm6tkRT7pS0I9hZIGwZDpkE9SkNJDKsp1u0q6qKg1a6kQAAAC6AyiALhqtZI1BkcGOjCYW+
    jyy6BZfJojiCiuENVKGvzNIk5M8aTIZS8Ne1KiJhrVcttCtAxFOcidmK6IT4BJ4kqltl22PDEQSJCKYl3dVBAxdp2L4NLqXw37/+4
    JkMIIDtEtX6wkb8jZiK50xIhuOsN1brD0gwP0Qa7T2CDBmk6QI3zz55xSocDXI1vNw/3bs8r/BJBdVIwiACo0uAwUQLPpb+
    ZRtkJph61eG6tc0KI/mJNx6yqGWEiwHkqvf//WzXinYRYV/2WLcA7CkEAAqiG↩
YCm6fQzIpOlAtmOfDIOhB4QPqHFM66aKDeNIm5zgaaamKBdVJDOA+QsdxKk8pJqlOWek0mFl2U1GhM0UC/
    IWlcTfCM26Sguxq9M6dPbJP4nCTtk3A82EHPIODjnJIQC0IRKI1nQzyPKvcDUqq1QqstqrAABSwLsF8HoGh/iSv0WOGR0g0P+1RayIUuyt/
    dt9jPdy1tWrWQKliAGGD0gTDAXUtjyNm9H
    //rqFZEnQa9G2iSQAU6/auWzQ4GUKJnYfFcDqAXWyKeqSouWl44MKgIeqGO7M6gihguWhj3S7sPSkdMGzuW4OGK+jMRv1Rt2aw7QVXarLr/+4BkJIETIVfaawwP
C76dpiulkoOlEn52xYALw7maFEAA3wXLB0VISwTn1RZoTqpigTuKLaVnUFQzj6sNwviRm/MytIrPzGQqaEag+lcQ0etzlI9egrvp+
    un5RLZ2xdBdttcbRQAKhQiVHhEDrji0GQcEBhS5qqhfdoQpijF6RUt6/i1JqR3nZU2zMTv/
    w33sx5QEDpwkXPKqgjeB4dO0GxqHkaFSsyVYpgtcVcJv06zpFVA7AX4vsqAUfseEQAAGC2gdpOOnWpwfDe0aHQ6XSw/
    BuKMGeKGh5UgsSbTSyguIDTJG4SNQOZv7
    ////1A///7gmRIgALQKdtp5hvsNWGKvWEiKAtAe19MJKlQ0BJqcYYIWPcSACXICHh7UmKqx5YvyddW84ECwg4VHCHDE4rmEqpjI46jkzxExHkFlLux0/DSEKqCQ
```

```
sM8xtFsBzI8YKELTrYaNmcj6mdO0mgk4lzzD6Ges1tFCRVjfKM5r6yWmCvIOUJUmHtHuou3RcHhwXDKCbyAClDYcaRKYuV4lyLalyj004MC6CUYCeoTC4IhKm1t
/rf
//+u+U/+0yuOAKAXvlRIBSkQFgNhMSG2cSZ55irAVmEvPLKKgp5GJD4a0musOqbpqculCU3nuzrpmfbkFVDBjHpdmUcw+zf7svz870nf9a6+tXXfShrs7K6xcct
0mQSAkpKi8pdGZFpW1QXlWeIvnS2sbfFVCE3eSeRjpxTM7ZW4/Fp5nZBqmc55L1ot/EsZ0oQY7y/8tbmSa736v7dpNM1Wv/lq4ppAKo3/0
nAAlEaACjASUAIAYee5oHOCyaDpfrOW2rQ2267azItHP+Shd6/zRE3mf2VTuooWqg69HQmb+
hQE422k0UCxK3IIQT0wpJwsPhaufOrllczQzeP4NMpHuu6oHZujnlReukrnciujkRt+w3lgg1/Si
//uCZKeAAsZM1usMEPQwZUpVZGIui2ktcaeYUTDIlSnxgI1w9bf//23Lp/7UIBCVUYDk11tljZVG5eAYJIqAqIxYVDASj8nJwYlYDYLSTaacxe96dgcMKlyQRas
BBE
↩
M7VayW8+Eof6jZ3r5FFU1TvE8q3I74LqdZGGMdgeHQTe0SDB3vcCPbdYaan03wqaVcMhZ9Oeol6WzDivtVzT+n1uDr7
//9rUiUljv40Sd5T+14U9BmtO96SYAEAADAAAAAECIAAtyz8TNLRHWQ1HsUlEbobFFX5N1/hf6kevuRVce6Cx+TER2uPZIjLsa4lqzA3SEw1Urr4IL/+4BkgIAC
KV8li4JdK0wk7PdLpmZVKfRAMFo8W5Rmf9upBlvAJhOr6ItHutuVyKbSPTi49cm3W6AUmrLAAAAE5C5MUrLDLJYClLGXnjUlfjJ23YOkMkRS7G0uW5m73uVRM
/M0ycxtnDlshkW0aauZzTdWnzEViQd0qb5o1qw7aUFRpJAWPk/QhVzm8V0P/7gmQqgAO/P9frDEtwNsWq/
TwiXg6Vf2WsIFHZAiesNPCJsvvjU4w8C+N5clVaypwQzz
//25c+HVbki7xWwQGrvXLZYjtQBEeJI0AAU8BOUwKGEcR42xjUaXmIbNoMU0mTsX/xETo/+Yh0+i6b1qrEZA3VvkXWVJI9+V+4YCbbGkUAAAnSEC2R50pb2D2SL
xDOA9ZLWrDPZ96PvUiCLJ+rfdgdBBaneq4AMHMRPue1
//t/////iRhl/jV1QnHY0ySSAXKFMPsfxhBknggi2l9ilISZCD+vLCVLTGvsyYX/DVPudKzBxo5ETkNdaCcHYiiCMshVzs5B5Cu9rfq355nVX+hrv/7gGQkAALZ
A4YBRFT9Gc2kr7m+stv7+v/9tdv6r/9SUVm+urgjIBFsUSfgGAgyAFMXjIetL1KRAdPsz6Ll2iqXtanyHXzWK/
tb2QrgtAmAokNlRUFpHpKooBLcjaSJAKcg6QDPK2OsrF
//uCZE0AErpf3OnjFEwzBBqsYYIaC1T/X6wYT1DFFKoRgwyhPtpVa9C4hDbyUU7Wk9IQJnkJLjp5snYMKjX/7/zA6ZlmfIfTcSqd1s85RJWcxAhgz2o1OjRPrR7
BN89tdjTULQ3ZqqX5n8+8k6wIgJVHU5qUPIsya3+UZwFFxet47aZRneiyOAYJqR7dNoukCVuOKYBPqsrIwSrcG8LgEs3KBlkVdnTdLnO7/+4
JkfYAC5SrW6wMz4C6hW1w9JjWLLMljrCRLEMwR63TwjfA77OmMku+d6O6
//pa8nCuFYMMSrBb8se9230//XQXHG20SACk40Ym0NpXiqq3ZQKAjUlIIuwK04+loiKUQgklx8vGaa1HsdDKVDEKxWpZN5hTzld6iL4mgtOCWNYcXUWUvbt3Hk3
THGkkkQC25HxZytMgahOgSBaxYXYLk5pqaQ4/A0sem/86zFDtqS0soS/9nfBKEYLzoQPrMgL9b+Xd9nS
//+4JkqwACzitYawwZ9DKjas08AnCLFSlrrDBHkM4FarWGGFj0Xf/39y9IB10bbbQKkuXYaQYuFB0ggfJQiLRIFgMKXZS1sxThGTPvqDkXKCpJAhO7Kpc7OuesM
3879PSJc/wYNDuT+jTLMcilef/zXy7SY6iV6WMG7ZJHIUS5MC7AwQUQv/7
gkTbgAJOKtlp5ipURyVquWUPbgjkkVusMGGRHJVr9YSMaBpJCIKpDE8HRFUEx5xpcJ0WFxELKJs4vW6GORzfoxyvKws5j9HM1GLdUR2IgTykUIiijju
//v/y///97et66DlzECt3qS0m/1I6HM9HNZloIho2rRaMZalBnHlqCyY1+JFphHQYgaXFSVeWuinJ0CFkBnRXjSwibic7hxKUKwLojpfdYrKajsUcbd+bn7Mojj
f866Gjfd2sL+/sXxbc5+Yj5tfyVLygWf/7gGSCgANpSVxvYQAAL4ILBeeMAEyEgXNHpGtYrYawcPANjplWFicGSwTf5GiX3bUswZYwCLFg+xI3+9
R3/G5xbgNav2Ef9pNr9dUESdykAgAqTL5KCEOZXqlDRbUeii7ptQHIqnGCkzTDCXL8JEygZBfejNwy6IJLY3yw8Uus6o/
VHr7zQyXKESJ6cazggk9ChHytrafvSrwzOMPwLXC4mCbjNM1v01d1Gb2PJAK7UhIEgh4Bh8d3Q2PiYEhY6o4nC5n5FmwJjDuCVy9X2aOg3aXanp
/d91waWeNWCiRjbhQIJJV3YRGi5XLEdI9L83D8Wk0YkJXJVDoYSfRROCiWUVXYrUJiXLnG7MKsychjELsvhvwQUwXC7mfJw45PuXT+F5GnT/
pZvsn4cFWdKUkVH7pfpKhfdTt2G2oQi9nJa8+RUQANQBAJAgGB6F64PR2jXJh6o8SrGlLbSJTouXYrrX7rZMbnnL9X
//kF//uCZJCAAw8+3OnmHDAwQdtdMSMmDJ07c6eYb0CmBWz0waSYAIcSAABABeyRMUeyGlvPoyYJosCKUZ3sSeQ0yHQWECSwAT1RbRa7PQLCiJEjekzyzMPk1Ep
GsoIYs
//uCZKUAE49KWunmG/AtwWscMEwgDRU1baeYcwC2Cmy4kI3ATpbVZ3EyPAZMJUkGrw9qV53eUlCUQoy+WQYNWSBaZR5VQQyEwYnddYxylVlVZ9Q7n8Y8Z446bEC
QzAZsdPSJECrzDaaekbJDYhSu0x6QgSJVNBLm400cdKvfx1EjEyBLnVowXcLetmAKYkUesVoQj
///5p75GdFP////F8eAXHGkikSk5qFsnSwxXoxDvBvANhaHuI0ryYKMuW8KVi+mh6SEZdwFDZQoaX2cmHVlwaCUedJC6AodeXMFnhR54Fjq3NljgDsk99A5BrRf
mEjZiEpCMxAzloZi2De7kZ/ayk0uX87adOYMEtM5v/
oAEJZAABAJd7rg0gRsZMuN3AGA4Oo1rRMEcciybCSwwX1FNDne0uzGGn0ijkTqL5MzebtpRLo5SsxVZdRMvXbEI/
yzgsEQYAwo4FDwCc3HWPXGDgkOQYF8lVSeLnrVRV/wdch5jSoBvAy44BcpfUOIh1YI1t73/
XUjw2zGXcjNPYgbPOYo84JCmYn1ppWM6xNNmPPqE7QzpmY6nr+t/gwA5VuofJqgLiWQJKbleEEHUTAy6C5E+
OddngdjTuMlnLdG7MPc7bVw5PizejClLO+Tqlpg/1KyE7yPmY7KkTqcUnA8RW9tLlzkYxz1NSayKYdzUsZm3tRTrp/
c5p7TlRlVHOT5q3fe7vp2rnKpNiboAIAAiAIlUACdhflzEXHoSaGpNyottaB8QIUpgdQiv/+4
Bk6AADLzrZUwwbRDzkykFhg2QMuKdZrDDFQPqVKeWBiigQoIM6hl3u9k3IY9IzvRb2VU2Zrz0XcgK1/o6tTaQAIySQAAA7e8cYLTqGCx3PZ/
Ayo1FnVm3u3OgAjQ9EkFKsVVYk9Mddq4+FblvlWnmHGvG/5kDyqi8ZxN0iCKoumcaGjP/bmKWmjIlqq3ZjEYn0OHHnh8ySbkLbooN1wki48+
KjX1ADAAABABii3gJAPsKwcJCC2tAQEQlAxjjOSgyoYMXb2K+lkmYDIKIanmk1JM1nS8/EX0O8L7U0/kGJ/
yMBqlSSAG3HhZIVlky3nmYFSuc8bfumyl/IjOw/EgADIyhpqulbBP3Ik47TotfKkdkANVZV1Vu0hVsSOUSMSokQNIjf/1dbPlO5kM+W/
o9jLgpLsmy1S+uy6GLpMrf0M21Uyv2UcVcAJVQSCZQE5mBE/D1aa1dYGxUtcvPQ6+tt/8hPUQBS/6n8G//7
gmTogANKX9hR41TWPIU6bT0FZgy9CVWsGK3A8RTopPYNIP/9NbxFuU446///u26Vt9↩
vVrlNUGPVNcZ0ASRxtC68ciNJuXEUpnA0x+ljSiDT5fIMijSCrZYsZmdQEDD2YYwglJI7hXd3QAaChjT/2nkQaIDWKIe6ElyHg0QIv/+
cQO3Zo8UZk9eayGPpyp+r/Raru9DU1S9FY4761oqP/4tyAuQgBAH0BN+Vsqb5WGlHkRtqzLnNoCfqvCbn7mQ4rQWnVQ+T/
kxcthdnP5V3l0Mpnkqp91BTizf/lP95Rrj9n6kAonGkiAC5KIsEnF+
X04rqFCzkKFmVZMWdTClLykkUUCL6zwdavDcHHUIIcySBPVkbSz85qprTsZzPmBMn
//C09yTAc5mf3mRrIf5hvaEXn9ht9K/VI5JXq7f9VS+lq/kvCCCR1SACJUYAKKu4HCwMnS4S9cBtc9Q1vG0xHos00YZ+Yx4pzPEe3t//7gmTpggMwYFZTBivGPA
/t/qLJUkjbRJIMkpjhstRIRvoMQsXMtCdFOeijNIfx1Qk8i1HtP1gLbCZCmOz0nZVd2JGuK1RBY2Q4kJoGZGzEV9T3Sint
//HWSvMV2FqKQ29HyT35iNy/X/uT/6pZ95aX/F+DWTEdmIFJEEwCaqTIAACoAJifxJzsocimbjIbHMmmPxICFcIASl75RTq2P6xCET5d6ws4moasELvpGX0LBAi
NtdywItZaODy81LRqd7GevrxJYulbBtRxiwocjU9bJ1dApR/JuCxOLd6YvEg3fePFzq16Rb/H9poOMV1/
NjOP8Q4UKms1ve8LRAEgDmACAgJkEAAACABqmQcBEKOGTIIjWCjLxzjEFXtAQEOgGFQzcxMmCIyM0YHRmuYh4ucNxUTTZmEsIAuQzEJZCIcXt1JAJeztiFRY0Im
/h+LaAwBCCMKJAAAAAIDQ9pqEjcXDEIVjoRolnMqy4QnNK3Lye59nqdg+2Ur06aC6LYrxHy6o9HIeh6bV5fG
//uCZMkABi1lVn5h4ACQxkoOzTwAVw0xZbmHgAoHpev3HvAARwRKnajwOftiPJ2xFgONEDoIhGK6Uux6mSnH7iyptkVkr8/SHx1Sq0fBOeBM3O2ZhJelX+y9KF4
```

```
SDTWEwsMUPDBMXK41iucFKioIzIe2x
 //+qmGAo38sp9ivrX+lT/QAqQABfB3jSEZAJRSEydhPTiIIl4Rq4YTJgMRmeDDlnHbP9G5BFQRWK52cz9HXIkMoLarTpdAaONMMQQmtjknpV7B4vXq1TrKGtuRJ
koABJF/JEgx1WKKpUJIvhQOzxJoYHmu7OfZAOD9wMDiBaXyfoQa3iRP/
 h9F3vFeZlEGVrKJJJbibKJACcyRGgH8PQKFTIFlIQM4iXYRTekTxKXaILYjmSKhQ95f2KBhlWeBXOR47VsVRzk
 //DeMZ/nP9tttZ9rjF98TjFXgVwxbTyPrmOgt1UgozD8wrCXkgoQqWAkKsLJ/QHyiyOQBhSr/flIkAQQCRkxOcinqbfHWUpjVMW0JVGNpvR5qxo4GODVnDOHOeE
JPJoB1xQO6mnA5RtUAHhgD3g5BoieWqCF0jNpA/a7REDsnLcGJ5dQeIDp5ESXtnHekFoRlb3v1jiAf9vjS/bX0giUE7ma+1
 rtH0yiw2pZIyHaWYiMVbH5hFNnQ95J/N9X+VXYCCZSG+A8CSpPh1EMUiJWyWNvwu4KxoHWDDZfRMOH0nBJHDrZsCxo8eCgyLWL+OE1T2+
 HGud/Ja9rlAAATgeNDkZbA5wZ+FquY+s9uAYmJMscEYGrejIZvrXEJ4aqim4OHGSiNvq1EYfXURz7y2ghWxRxHVWsW1Vixqv/7
 gmQgAAK2NVYbDEJAOGIayj2GOArk/2dHpE7Q6oxpxYSN2PMcuQTfP6W+9TexrWyijalpF63E3M+
 gBLRikELgLDBAZRDQ0HpWn0HhDDdMH69eiMqSeoHVZeFVRP4LDcHRISFybl/bWmxcLv/+
 ePb6HcohWrKSBSlrENAyyuG2zjXJuGWL4sKhMTrbSa2KHWMXou3e1PyDWMfFwSyoOJATbMn7We66164dShSmNpr4Jyt6o7/6
 lsZHoijtKuXYKpsgYflGWbke07oIhfIWaAyCTSlreKEuIlwni/
 VtPPUCQhWni4NG21KKMVlpex1vPsyv9VhStPtfSXcLMIqScEACImLf1gHAAIAAV3fZpQHUNHBSn/
 jA6EwE0EhDap6gD5d1HFbLGDxXUrxZWZOmriUelCN54lIMjsWdiMxDkUjKHA3/+j/q1
 ///49TEnHxoXJsFt7HyH/jK3gKPIESDaBBKxTkNOAAQqBYL8fpYE//7gmQ0hAKfP9dTDBFgOSG6qmHpGgpdG10njLGA6RTqJYYJIEvnFtmVQNdWHwy+M8C7qcQU
VNQvgDiwVda0imKDGJBGGrDcTr4fKbL2TRuzqyBnbT+de3Wzl28lCb1/QYQ/Qk0k++1
 NWltxJJJEgpyELFsFGOMk5DUMNMXiXsX4r41gslFEg1xYx0dLSq5kDtdTGc1uVFaxqNVTtrP1RPY9v+2t2fvucrY/3Gn+
 pGW6tZQW0XDYMPAEPInjf66dpgVCMRNrou1IhqzjNl4K6JTkMOhYJrqiRf47VLNf7gmQ4NOHWJSTqEAGGFAooXTAgfAqmU2d1lDBuZbUAICbmsTP
 /7gGRMAAK+
 Sdrp5hMkNqJ6hWGDKAsYq2GnpGqA3gvqZPYZCN2PWW1ZL6S4JgW1UJkTVMOmWcJU0BqBEw7lOFnofDy4CiP2KfPz4ZctQzOYEGyXgVBLmFcwJw2FVPA4DJhNRwE
 /QQAiTUNAYBnPkKToaYOYjF4WrXUEvQUdfdHZfNbBK8jnoQ+
 Sm1mmawZvQta2XQm2EgMO82sE83HHGQEruvMcG5bO1ez919HVXowDokYLjjevWakx4qmLSORDNIY9KOxHyo/6
 o6zayDkq5RvorBtv10ZRb35tN36u6CLvMCpEMyJqpo5580SLNcW5XNtikCsoAAAAQKzqeEmKUDsgKhrCoDcZSP7GUP8Ug16uprXM/OqdUR/X
 +kQkuk46wEzu99H/1f+qBBwAjSRRJAATu5J0EWMaqWTB4DM8lDD108H0wRNMu8KoZSLf4TatyzKPpnFPPPKBzupGz
 //uCZGEAAsU+2GsJKkA3hIp6YYIaCnypWaewZUDsBap0tiQYigogRICh2RjGd7kE2RiHVufNOPKS6ws09JNIqJ4mV+MHkQEXUiyAgVLgLxIFI8AHCHyuAgvgQHg
KVT
 ↩
iQw+jpsedYTNOTSUOrAPdsSqGmQc68qm7Fbq1WxUmzVvXzgixv58lvf/4H7sps+
 GPd9ElCpq24kgABdwAwPgmRAqMi6TjEsHy48RasIOQHzK6OOsaSnnPatGraWbRit7LbBdlIub9udyXBxhcb
 ///4/uymc9YYT3dM1JIRAAFAPoXAS5ECfIJQm4hxMDic0uuXF4upsOTZjdIvkXfr/7V8izLjdRnm//uARKYAAj8pV+mCLMRIJWqJMGiGCSSlWaeYrtEeFKs0wpp
HeJIJGy9+HfBqElX4kUXa/
 bcOIrlGacbpphOacK4CcYYbbuHdUSRNulXBp38ulYhioZ2Njeq19CxGr6M7gxx833SPm1tYtn6fyq9ndb8dugmilGZmxSG+nxVnL+
 ZaUhMhl3vt6+e2ktfGfNjX3vuEigQuPiJDjsjzM0HFcZ1T617f
 ///6ve7x5SUORgjIhI2Vy1s3/+4JkUgAFIVNfbmHgBFzlS0/HrAANLStlvZKAAMAF7vOYYAYwA5o61LTBGRhEoBSiBABwCoDUHyzk+YCERFbJK6WEI8U/hdA2Ow
SpyQMlIFAaQECITajj6OR0LFWeXR3VaLEwAM7PRKHNYVcRCxxt6H3/YFdRUXZJEkCsLTH/6uzAmc62kO/
 Vya1VL9Lhcfau91xKAS3jIAAAACoJ6JTZjDjCog5ktZAu1ibOmUs1ftxaSjCUFIiDxTbyjtbaSH6LkKbc/pMx9qesff/7
 gmQbggOpOdhrBlvgSYlLXT0lVo2VLWOsGQ3A6BSsvPSI2PSmFq2gHt2gB3PIWQxx73yTiIQBq1N6SEVB2Ip8bkV9M+
 h7OJilB6xAFDR9SjbnPiEXWDJsFijN9/vDdXYLByTORtMkpCBkWCqLMR06xno5rPsyB6QsfwX2BNr5F1I8hBP1KzYqcouq/Xtt3ydQ9bK/
 Qo0JmdGWln3s6pmIv91
 ////+aMvxX0AKNVJIAzDClLF8vxBbwV3ouPZEX0guNxwRHtAkHUBZR5hnnESmmU0AOFb4mUtdm+FbOFu8q0IvyMAbdjzHaFIWBIOtafSplZd7WUmJm544a9q6/u
S5Mv/RgzcaTIRWMDL3O5/mcSaje9Vx8vCo5a0A2kQA6yEmJkYwhBMyIaWJNIerS+x51D6v
 //Fh2qaG9CTpdfmfHUeDdYadehn////8r//5KhBUh2dVIpAAvgRRG8bCwvhudC40iZGBUmIoRYbVWf/7gkQPAAI5C9l5jEiQRuQ7TTAilgkoSV+nmGcBIhSstPC
BclOSf4wglailFBXLdpbKPU6So/8xt9lWputrAPL/d9UYVUkagwd9DiRNIrpM9o0YkX1mHP/SgJmBCaAAsZSQgEA2xyexkaM+0
 ec59ngZcBUsHuSFGyr5VV9P9
 //uCZDwAAr4x2OsJKrQ0RTqcYMVGCv19X6wkqRDbFKs9gAoQ5k2Ruv12LtuMR0K+Mj4uEL+jT5YAitpskkApyPuSJQ+Y6zCkfKINzRTIASHBUSEqestJDqqbTK7
pjm0dIO0UBH1SiQC5AOIDuJ0Nce0hu2mFqYxKjzkRA+JHMaIPe6kMJdg5uzWuz4ZTtq0nIj2zsS9GuYzJf7/a/IrnGV3Ywl6HD
 //uAZGkAAtErU8tGS3A1RTpAaGZeC2EXY0ewQ1jbhes1h6QAhyMQ4RKDAZjBwQsAKOYgmeqenf7CDf3/YBM1OrYID2AipCwU01RMXCBAQjxdYjVP9PFEWYmI3VP
ZEeLJqzyT6a1UhndzMvsVat1pLZ8i9e12RppC4MKDzvmA0WLmBEyinv6v0SX
 //7EUA+rkSSW7/+4JEk4ACOhNZ4ewY3Efm6v1hIhwJPDVTrDBjQR6U6zWWCHhF6igGBkx5DjB1C/kQAZJiwZSQERZNyGRNeic7TNuu+/dfRF3RiaMM1Y0roqJ4P
8D/mTfdhlulTIsHgdW/PU/8ef8KMn33f13mz0GGqu+R8qzCar/+4
 BEvIACTCrS0ykpwEDB250lhhmJCKlCzRlvwR4VKWWjLfgJOKN6NdOAAXWWICAdgEszpgxAmQEpBLZfR02sXGm0E7C72UhixhahDs4G2yi0DET
 /mmWZhBq+rsDOIiJVqaAKcd9ylyn
 //oBCT8LZ9FYRBIZDYjVZRXjhCq08YyJiFfymJJXmUOKigRItkXLChFK6jT6RTaJQxTp0ERZKAbBBr2lN1+nmWuS6lommJpo/vNjLsaO//bFOux0Vrwlrc9KpX9
Q9JSnq4RZFOUe1XRzIunq9/f/7gmR/ghNbLthXaQAALiIrnOSMAY6ZLV+sGQ2AuY1uMMYM5k3ce8PnAGwIqBsy+
 K4n5NKybSSfwUiwYFFJEGJieuEshGA/BRzNljr8i9XNorChB1tzhYR1tUo1Bill4gE1Ac0qQAAAAL5vkLVO0eB/m+
 byROI1VGrEOPNFrK3wcmVBd0bsW2VrnQizxf1zAb979PTZ/hz1l5RPMVZrUSvP3fctLySump39qwLkB4qsMEQZawKho+
 RMuaPtVh8GRj1zl77vrUeEo1gCUOAgQgEAICQgaRvEOgSMrAXVhjGanBMUN9db/S+OA/Epdz872Q6hmKgVCbNJCd6f
 ///6QFM4mEAHjGhL9UaUL+MBCYWAVSDyWYS2XiWftnJZD5qifcQcv1apIzmNbJ5CR7bmxsygxeviUvdEITAqA6O26rsbDeySXRrWS8zaOHPNY31/XW4ImXIadPD
uCZIwAAylJ2HsmG7I0JHu9PSJFjHjJXawxBsDOEGz88Ymg/Fmky3rbWUFdk06vs57f9R1bv/77f
 //+pSa5ZGkSQSU6LcJ2ZxDSBE+B+mwym+cR/HIT7qBiYwGJIARZ87uycXKhFx6il5FH9NYf/ywfhkswYVLlFpPlRQouYmiTnJaS4MDziFmHd+s7Sa/8V3VpADGI
I2S7SlbaeYTLDNFKt08ApIecZnTYFEyXI0MkTEpKS7jxeyGeYaCqejkhiPX+sL+
 q2kAEDE7MAEBAFgRUDC6S3ALcU1njadnymJbBDUfRCB1P5cllHoVPt/9rA2MFG+MX2k0OGXOaFbFcIwYS2D+
 JwVSEFwVilJVkGWshKzjaocrKHqkY9GKtgskW6AxrjzAAs/ql0dysBke1a53vWtVJBoBrW9SCCfQnXzMnoNi6ig7w/633L+waAgQM/2
 AShABYwijtTTinyYjPZY+84Bkc81I9jPCpOWTTDiDY/Iop/gRpFWwkrq
 ////RSctBNs00VHeXwc53GuX0ow0j2Ow8TApIIC3hBlw8bFCOJ2DK8F1r+N29MZJnHhQlTZk9TFdp9qplO6xYKBOL0+pWeoQ5jKikCg6dGkZhA8Xb0O6Bpnjvc9
```

```
afmZkImQBRMzPa0UpNk3T40M5nne2zFnDVAESmLUlkO7dpqNlMp/Bsqsw1lYVTltmB0VoOks+
 ↪  cZn3Wb6g4QF7GGnXxJ0V1dB7jjkc7UKmz24Pcyz2hf1VFwxt52nL+TNaViof4VFLnqX6S+
 ↪  KA0oIJWPKEy71mK9geXoFE9LJoPpWZAQLGPxSIAgEcEboWinTDCB620RY08MLk/iTsH7BRoPE2DMnhWWRJRme6o9a778Qp2mq5P
 ↪  ///QSiehAYaGGqLAVl4tQf+SKjC6gTQCeIuXs5yR0uAgHAjghc3yvKhuo1Ckrwv01FJd92dSs9olyOoR8bP+3eY+sInu942VbBwYMZX9NPEs9a50+R3t7KmoRpS
DDXdDv
 ↪  //SBoBEZpKNtBa7sLYYh+x5/Xef8f4qq3xCE0NnRJH2FOa/UBhHJCGUjMc+uDuyu4iilw4Cdipj3dzEEoVYdSPVqmKwQUlPqcQ3svS9qbM66U+kCMZWGFuqm9NK
6IAChufUjCi+lAaaRBXFY4oXISECM1NzRWyohw5BRls+1YcxRkZS6uSxwjIX9twxyHYAFKix7o4gxThkr977aE9aP/T/8hnR1Ih/
 ↪  Buyqymdk9lZrblZD3zkR7apM4M5QMoT6SRYUGCKKAtXzCM0msWTworXptMqaBEt+ZO75TaG4vXwk9W9LKzvIJP93
 ↪  ///P6knJgeAASEe0bQWv4jhEVQwOBWCw9lsdxCILR6LiWuYs6vTHz25CQO3wZUYSL3UzLNm1GM/JLomqOt1dlyBZi//7Pv8rXR1/tt6oylu+zOhGmEjs7uXoler
uP
 ↪  ↩
Nw50qlgIFAIjigJEoomUEZXUWSBRNA3o81qpOu0XRKn0B3U0kaikW4GYIgiAo9VaJghlFKK3z1KRodNMNlGHYnNWhQ9eY5xzpnz77Of
 ↪  //58w4kRwIGqJ17nfZ/rD1TGLd7/29pKO62XMRvOm69xDjXng/DE/O0iua/edSZ2/Uc4GwUov8J9CVzRy7/eZ+1HVhMFSAk6gAAAAVexC0C3D2NcxxknMX89Cd0
9uWOpwuEj4AYMDY6QSWEJY2tz0pjjpNQFMpUQdWi
 ↪  //YES4EADQjTgq4NrEwMlqDdvmaRvO339l7JnvNWaNBW5sbhkVW/uwgLm2DXOcLPBxLuS5CpFmRIABKcwWAl5Ul7bRN3jWZKVMdMIxZKaAyyri2v5RrUNLKFvBJ
OlsS3JHQ0qpWsD7mrYygtGh5Cjqp2+xHc50MaeQIR7q8pOQMGNwM5bj6Bksj0aHC3dnntyNaf/Z1kkKEooEEECSweIQCCgbhNxB/qgsZoA6vP1Cz
 ↪  +F5vGRgB/WBzrj/2OOqOn5O4ayKDv
 ↪  ////rdItoR/0IAKLEkKXsNTpSLYfCILHzx5/G3bR4oIU4f8yD5OoL698WhSHadI3qo85eGrZ/lTT7CLAfyJhJceVtQRBxOIqE2P2sVUeeVQVAY9G9bcKsqlf9g5
r/q7AAhWAABbvPIJOcB0CpegLp4gMEihsCj8RIvkhPjNyFcycM9UlfyOKak4fpRWM8/8x9nJ4gxF9KaZsHgb/J9EXR7P/wDJfoph2gBIIADcD/
 ↪  EpU+WWIQOwYGtqCR8kRgOR1OE7vm7ldsd3Jwz1VU80aKao4fpNYw+enmPsZPoMRZUZtHOA4C/yaMlEpT2f/
 ↪  jVSAAJAABd3iKZYKTL6JsKLk8zB14ahMfZPR4jrGIil1kijGVEZLa2D3p2Z+caae/z7zqmP/7
 ↪  gkSoAAInF1rR6TGkSOVbajzCeIkgrVtHpGdBFhWr3YYM6CYSh/
 ↪  wWsR3Btf9HsadrDPzjXoaihIUCoKWrfJYFcxIftpakqAMCKiVAOMUDS6rBtKmg51GbaYPFSviQ+iN2+8+
 ↪  y7EqFVGCgqhA0JrhhhhBONLGgDMlyAUR/+oQV2kSSnJdEbFwEkUId4hD8kCadKUnE7Ysoa8FDz8uQfpjCr0l5ujXsu1aPl9NsPR5EEIhT/
 ↪  CHX/p9Kp9qNL++j6KQglYebPLbQbPu3eoS0qLATsG1Aq4OkSvSLUGQ8S/WI7bFjYlATWsoDHr3HLUwqSrr/s2772g2e/
 ↪  VbupQjq06jvwT60ygABuaRKkliTd2ksoiKyxosshh+EYJgTm4PJvMIJNs23P6tV42SeNqvhPUufqynFnfotnEHPRhApjU0sEgtRyf7+
 ↪  dNp3lMh5szJt27KKkOcZnovpGcnJGDhtKqIslIAAALtQAgKkoD6IWf/7gGS+
 ↪  gDK1KtfTKRsgNqGqlWXmNAqlCWlHoE7QxBJqlYYJIQYYAPOUfAtIoBWrGGacTtxYlWXIFLr+uv
 ↪  //ojtZnKh0KWutwYK//orEABwAAGXd+BAdDoDERZWFAC1yBGoKZv4LtnDIkhpubmQOnrEoLWxawppLciv0Z2/2n+ZuVJ4GP0zDDT/9CzcBrPFpUkl+JV9lRS0Bz
zieSezFKdTWXuB1eXt9taXo9Bf5a199JW1sFcAO/1bBWpASukkSnJU8OcapKE2XYX6OOpEtqwm3JjQL+j1Y1Ozu/COYYrVjUJ5oZFu9V0/
 ↪  yP1K1tN7hFcN+jbGnTWs6tOleksfSQXn548P0YTxAl80YA6h4fQAVYkgFtx9guEeyXUU2YkklEp94aF5ZVm7cUh/
 ↪  VFYx21d1hCmgIa1Y1nsjKXe5iPUnfqUjleTe4Is3/9qV
 ↪  //uCZNkAgulC2dMJKnQ0JUrcPSI7CrSlXUwkaoDllOr1hIkw1spZhAMmRp1af5YH7F1H548P0YTxA/Qg3eTAAJSdRJALd7wOFYEMB5Oh6k8ZbS8L6hrcvQUQ/Y0
WQNRfmYeTyNVop/lIihMcT
 ↪  //3oH2UgL//9SgBgBLlIugyMTJK0xdHtgYIEgZpB3QQeGxQtiBf5XFHqNJDG94o7n6wZl2nuKDoqZeoi5EE23hUFEc+lkd5QYCOzqdBSDi3rPnycz7vk9d///7/
mxrQUmTiLGFf/6aZcZaHSlgulM84MAZ
 ↪  //oRbcbTJAAAUsGDR0saocqZV0y5it2Ygx5ne3GZDECgYlYc4AIu5bVnSfdzO/aMiz5xbO8+cx8KRQJoEVP23daN/iiTPEakdW22t/+7Vr/Vt+//x8+UdWbP7dk
jQhgs1FIfGkbLhXTUtoq7PP4eReqpncG2Rf/LvSFBERPeSJPXIxF0cuQUQBIAE+VEv2kE3zuFV4nSNOG2fSskG3o/hdkjLnShl/i0dITNq/
 ↪  Wd2zHds2p8f2hfHVF91zKG09NvurWaK3lv/tVpQK8vS/ZIzc6cUpWTla+j/+4
 ↪  Bkf4IDsj9YUwljYDZkGthhgzYOiP9frDDRwRgfrPWHiGL683nzbGjddkYnq81/57Zt+NcVOEAqefY8Uvbk/1
 ↪  K2MuqQkVDCoGkSnACqSLLthDGGwBnZ1IDV67PxDk89Ri2EgIVPmMp1f7MyMKexXT9sExvun1SpzIhjM37Vs2aa+22
 ↪  ts6ID40hkEKSBtIAABdJFuq019HNIFpsXV8r322Rm/8
 ↪  r7giVS6G4dhmJcooVHGhg8o5E6vGpOdJ6ivrCVTxmlYSm0io8LFj6iDilkfDxDECPfy0i1L4S2qfVZTtrBNbkHdvroVkZW/6f/9
 ↪  a0Xd25xGoYUAyJc+QLARYSxOku5ghhVyNizxNHzLwMF0RYVi5o5/zvRAohSue+h7FRpv/csMl/
 ↪  MtGsdQS7mS5mvsVr6HSQZATyesvKOlxODhRqFIvlV9KwpC+1jzc03li7fH2ETtaLQunyNernvkyjEcX637o3kgRRWGdr4mf/7
 ↪  gmRwggNRYNlTCBTkPSQK7D2COg1Fj2NMILHRBQprdPSJgO3rMq1dtM4Wbvjdu01ou9JZtF3ViOoQdl200syroR1TmftJ
 ↪  //VM46RSZ2MpkJaQSKCAQIoWgEVKEUhRO2l6FMBbccxeuSGS4F9LgYtuMnuAKtDXhiDnvEHTJsKBoXBCEUoQ6CrtneGFlvpcJOtN7K2/TXKWxFEyiSneouNMGou
+3Ts+HrWlKy5SAqa9Rx3r0pEv1RfGZJlGA6k8X5FpCGkrgELGk/Z/1qEMBJtEgFO6SrokKxuoeFBMGlpN4Gofbh/
 ↪  EXRDxfEcctv6UIFOvefwCVTUaj33nn/qbuXP85S6zCEWkevfhZX
 ↪  //7gmRsACMxY9vrBhQkMgJa3GHiNgxRNWmsMGkQvQUrIGwsSWuGQKIy/ZZm7NTNjlz+/9ufV5r+uHoTDB6MCXZ5E+IEJkywIAHwpMJJxaTi08jsJi4HMPutC9jy
S3KKXtjwg0qwXQXfWYqUi1NXmP/U6jfD8X0YyhXLlP/
 ↪  MCOiTCIjUdOK6g4L8GuLsRXVP9EIXxg5wNkEkgSQDOIBhNRJB1axuj0aF4VUz2VEAA0leSXsP/7
 ↪  gGSQgGLCH9bTLBqkNcE6iT8vEArAxVtMpK0A1YrplaeUmnh8K1LIXuxLc4PU6fp91I7h/QhFL5rYlc1fABBwoY7LVDCg5+
 ↪  z5kTMSIlCCqcCUgYQWARILGKCYsll7SlLkxYPrdJrF8Ig4Iz3UkANgChAAAK0QB8BPJMIMoLyoZFbhABU6bsjNg440fZS+q4+
 ↪  dejBXO53Y7kv+27+KhhWiKYWEYSOyBQxRJJEpOWDUJys48/NW9l9BIl1VXmlEkd6X1sp6VUmF2llI4ZMIgR4HT/
 ↪  Jb6I5Qvt3n1Fc86gm9gTAyMTddu9V9nsrdveQ6+iq88qSImzkLCrYhrTH2NPJ2HGiAWn1lw1GIoYoB6cF+pbD2E8ZaiR7T+38yctX6ix5+Z/
 ↪  vUPs5CSlY8JMJAp49T+6j99
 ↪  //SoyByZVQokAm9zDJdzW4LgNZrHtvKuVTaUTjuu7IoxdLwGIJtJTZHY+ohUNGlY+vha29emm+r//uCZKgAIowpVisvGcQ34WqtMwkSC80rbawMUVDIkynBhgyY
VVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVUABCSgAAAAACAL
 ↪  //uCZMAAAuZLWXsGE9AxpVqcYYI0CuD7Z7TygADNFOq2mCAAUgbp2wGkA2FMjLBCQKCdHAvgU1FsD8PEbkkvLBLRiDBq1+FSBahghPg53/h1AygACgBCASweutb
qqqq
```

```
const VOL = 0.5;
function initAudio(){
  if(actx){ if(actx.state==="suspended") actx.resume(); return; }
  try{
    actx = new (window.AudioContext||window.webkitAudioContext)();
    aGain = actx.createGain(); aGain.gain.value = muted?0:VOL;
    aGain.connect(actx.destination);
    const bin=atob(LOOP_B64), u=new Uint8Array(bin.length);
    for(let i=0;i<bin.length;i++) u[i]=bin.charCodeAt(i);
    actx.decodeAudioData(u.buffer, b=>{
      const src=actx.createBufferSource();
      src.buffer=b; src.loop=true; src.connect(aGain); src.start();
    }, ()=>{});
  }catch(err){ actx=null; }
}
function setMute(m){
  muted=m;
  try{ localStorage.setItem("cavalier-mute", m?"1":"0"); }catch(e){}
  elSnd.setAttribute("aria-pressed", String(m));
  elSnd.textContent = m ? "♪ coupé" : "♪";
  if(aGain&&actx) aGain.gain.setTargetAtTime(m?0:VOL, actx.currentTime, 0.05);
}
function duck(on){                       // la totalité fait baisser la musique
  if(!aGain||!actx||muted||ducked===on) return;
  ducked=on; aGain.gain.setTargetAtTime(on?VOL*0.42:VOL, actx.currentTime, 0.35);
}
// Plein écran : un bouton, et la touche F. On part dans la fenêtre ; le
// lanceur (osmo-dino-play --fullscreen) ouvre avec #full pour partir en
// grand. requestFullscreen peut être refusé (WebView sans permission) : la
// classe « full » suffit alors, le cadre prend toute la fenêtre.
const elFsb=document.getElementById("fsb");
// Deux hôtes possibles, et la page ne peut pas agrandir une fenêtre GTK
// toute seule : on le DEMANDE à l'hôte par un message WebKit (« osmo »).
//   - osmo-dino-play : met sa fenêtre en plein écran, ou l'en sort ;
//   - le widget du bureau (osmo-dino.py, 400×260) : ouvre le jeu en grand
//     dans une vraie fenêtre, et lui ne change pas d'aspect.
// Sans hôte (un navigateur), l'API Fullscreen fait l'affaire.
const HOST = window.OSMO_HOST || "";
const hote = msg => { try{ window.webkit.messageHandlers.osmo.postMessage(msg); return true; }catch(e){ return
 ↪  false; } };
function setFull(on){
  if(HOST==="widget"){ hote({full:true}); return; }      // le widget reste le widget
  document.body.classList.toggle("full", on);
  elFsb.setAttribute("aria-pressed", String(on));
  elFsb.title = on ? "revenir dans la fenêtre (F)" : "plein écran (F)";
  if(hote({full:on})) return;
  try{
    if(on && !document.fullscreenElement && document.documentElement.requestFullscreen)
      document.documentElement.requestFullscreen().catch(()=>{});
    else if(!on && document.fullscreenElement && document.exitFullscreen)
      document.exitFullscreen().catch(()=>{});
  }catch(e){}
}
elFsb.addEventListener("pointerdown", e=>e.stopPropagation());
elFsb.addEventListener("click", e=>{ e.stopPropagation(); setFull(!document.body.classList.contains("full"));
↪  });
document.addEventListener("fullscreenchange", ()=>{ if(!document.fullscreenElement &&
 ↪  document.body.classList.contains("full") && location.hash!=="#full" && !HOST) setFull(false); });
if(location.hash==="#full") setFull(true);
elSnd.addEventListener("pointerdown", e=>e.stopPropagation());
elSnd.addEventListener("click", e=>{ e.stopPropagation(); initAudio(); setMute(!muted); });
setMute(muted);

// ---------- entrées ----------
const keys={};
function engage(){ initAudio(); if(S.mode==="attract"){ start(); } }
```

```
function jump(){ if(S.mode==="play" && S.z<=0.001){ S.vz=8.3; S.z=0.001; } }
addEventListener("keydown", e=>{
  if(["ArrowLeft","ArrowRight","a","d","A","D"].includes(e.key)){ keys[e.key]=1; e.preventDefault(); engage(); }
  if(e.key===" "||e.key==="ArrowUp"||e.key==="w"||e.key==="W"){ e.preventDefault(); engage(); jump(); }
  if(e.key==="Enter" && S.mode==="over"){ initAudio(); start(); }
  if(e.key==="m"||e.key==="M"){ initAudio(); setMute(!muted); }
  if(e.key==="f"||e.key==="F"){ setFull(!document.body.classList.contains("full")); }
});
addEventListener("keyup", e=>{ keys[e.key]=0; });
const frame=document.getElementById("frame");
let pointing=false;
function pointX(e){
  const r=frame.getBoundingClientRect();
  const cxp=(e.touches?e.touches[0].clientX:e.clientX);
  return ((cxp-r.left)/r.width - 0.5) * 13.5;
}
frame.addEventListener("pointerdown", e=>{
  initAudio();
  if(S.mode==="over"){ start(); return; }
  pointing=true; S.tgt=clamp(pointX(e),-6.4,6.4); frame.setPointerCapture(e.pointerId); engage();
});
frame.addEventListener("pointermove", e=>{ if(pointing) S.tgt=clamp(pointX(e),-6.4,6.4); });
addEventListener("pointerup", ()=>{ pointing=false; });

// ---------- cycle de jeu ----------
const elStart=document.getElementById("start"), elOver=document.getElementById("over");
// Chaque partie s'ouvre sur un ciel TIRÉ AU SORT : l'heure (donc où en sont
// le Soleil et la Lune sur leur montée), le jour (donc le temps qu'il fait et
// la saison) — mais pas la phase : on saute par lunaisons entières, la Lune
// reste celle d'aujourd'hui.
function tirerCiel(){ S.skyT = MOON_PER*Math.floor(RND()*40) + RND(); }
tirerCiel();
function start(){
  S.mode="play"; S.dist=0; S.speed=9.5; S.lives=3; S.invuln=0; S.cacti.length=0;
  S.spawn=1.6; S.px=0; S.tgt=0; S.flash=0;
  tirerCiel();
  elStart.hidden=true; elOver.hidden=true; paintLives();
}
function gameOver(){
  S.mode="over";
  S.best=Math.max(S.best, Math.round(S.dist));
  try{ localStorage.setItem("cavalier-best", String(S.best)); }catch(e){}
  document.getElementById("finalDist").textContent=Math.round(S.dist);
  document.getElementById("bestDist").textContent=S.best;
  elOver.hidden=false;
}
const elCard=document.getElementById("card"), elCardLvl=document.getElementById("cardLvl"),
      elCardSub=document.getElementById("cardSub");
// [2026-09-07] PLUS DE NIVEAUX. L'éclipse ne ferme plus rien : elle reste un
// spectacle (le ciel ralentit, la couronne se déploie, les cactus s'arrêtent
// le temps de la totalité). Le ciel n'est plus recalé non plus : la
// conjonction revient toute seule tous les MOON_PER jours, et les phases ne
// sautent jamais. Il ne reste que les mètres.
const livesEl=document.getElementById("lives");
function paintLives(){
  [...livesEl.children].forEach((n,i)=> n.classList.toggle("off", i>=S.lives));
}

let lastGap=0;
function spawnRow(ramp){
  lastGap = clamp(lastGap + (RND()*2-1)*2.6, -4.4, 4.4);
  const p = 0.34 + 0.46*ramp, half = lerp(2.5, 1.35, ramp);
  for(let lane=-6; lane<=6; lane+=1.5){
    if(Math.abs(lane-lastGap) < half) continue;
    if(RND()>p) continue;
    S.cacti.push({x:lane+(RND()*0.5-0.25), y:96, s:0.72+RND()*0.43, a:1+((RND()*3)|0), hit:0});
```

```
  }
}

function update(dt){
  S.t+=dt;
  S.sky = computeSky();
  S.wx  = weatherAt(S.sky.dayT);
  // Le jour de la conjonction, le ciel ralentit quand les deux disques se
  // recouvrent : c'est un jeu, la grande éclipse doit durer.
  const dEcl0 = Math.abs((((S.sky.dayT-ECL_DAY)%MOON_PER)+MOON_PER)%MOON_PER);
  const near = dEcl0<0.5 || dEcl0>MOON_PER-0.5;
  const rate = (near && S.sky.cover>0.9)
             ? lerp(1, 0.25, clamp((S.sky.cover-0.9)/0.09,0,1)) : 1;
  S.skyT += dt*rate/DAY;
  // orage : un éclair de temps à autre, qui illumine toute la scène
  if(S.wx.storm>0.5 && S.wx.rain>0.45){
    S.boltT -= dt;
    if(S.boltT<=0){
      S.boltT = 1.1 + RND()*3.4;
      const bx = 90+RND()*(W-180); const seg=[[bx,0]];
      let y=0, x=bx;
      while(y<HORIZON-6){ y += 14+RND()*22; x += (RND()*2-1)*26; seg.push([x,y]); }
      S.bolt = {seg, t:0.34};
      S.glare = 1;
    }
  }
  if(S.bolt){ S.bolt.t-=dt; if(S.bolt.t<=0) S.bolt=null; }
  if(S.glare>0) S.glare=Math.max(0, S.glare-dt*3.4);
  // --- étoiles filantes : par nuit claire, une de temps en temps ; la
  // dernière nuit de l'an c'est une averse.
  const nightU = clamp(-S.sky.altS/0.20, 0, 1);
  const yT = yearT(S.sky.dayT), frac = S.sky.dayT - Math.floor(S.sky.dayT);
  const lastNight = Math.floor(yT) === SEASON-1;
  const metRate = nightU*(1-S.wx.cloud)*(lastNight ? 1.8 : 0.30);
  if(RND() < metRate*dt){
    const dir = RND()<0.5 ? 1 : -1, sp = 520+RND()*420, tilt = 0.35+RND()*0.6;   // en biais, vers le bas
    S.meteors.push({x:RND()*W, y:RND()*HORIZON*0.55, vx:dir*Math.cos(tilt)*sp, vy:Math.sin(tilt)*sp*0.6,
                    t:0, life:0.45+RND()*0.45, len:70+RND()*90, w:1+RND()*1.2});
  }
  for(let i=S.meteors.length-1;i>=0;i--){ const m=S.meteors[i]; m.t+=dt; if(m.t>m.life) S.meteors.splice(i,1); }
  // --- le père Noël : la dernière nuit de l'an, une fois le Soleil couché,
  // il traverse le ciel de droite à gauche, dans la neige.
  const santaOn = lastNight && frac>=0.77 && frac<=0.93;
  S.santa = santaOn ? (frac-0.77)/0.16 : -1;
  if(santaOn && !S.santaSeen){
    S.santaSeen = true;
    S.cardTitle = "Joyeux Noël"; S.cardSub = `an ${Math.floor(S.sky.dayT/SEASON)+1} — la dernière nuit`;
    S.cardT = 3.6;
  }
  if(!santaOn) S.santaSeen = false;
  const ramp = S.mode==="play" ? clamp(S.dist/2600,0,1) : 0;

  // L'éclipse totale est un spectacle, pas une fin de niveau. Le décompte
  // tourne quel que soit le mode : le ciel ne se met pas en pause parce que
  // le cavalier est tombé.
  // Chaque midi le Soleil traverse la Lune ; la GRANDE éclipse est celle qui
  // tombe au jour prévu, quand les deux disques se couvrent au plus près.
  const dEcl = Math.abs((((S.sky.dayT-ECL_DAY)%MOON_PER)+MOON_PER)%MOON_PER);
  const atEcl = dEcl<0.35 || dEcl>MOON_PER-0.35;
  S.totT = (S.sky.total && atEcl) ? S.totT+dt : 0;
  if(!S.eclipsing && S.totT>0.12){
    S.eclipsing=true;
    if(S.mode==="play") S.invuln=Math.max(S.invuln,1.2);
  }
  if(S.eclipsing && S.sky.cover<0.96) S.eclipsing=false;
  if(S.cardT>0) S.cardT-=dt;
```

```
  if(S.mode==="play"){
    S.speed = lerp(9.5, 25, ramp);
    S.dist += S.speed*dt;
    let dir=0;
    if(keys.ArrowLeft||keys.a||keys.A) dir-=1;
    if(keys.ArrowRight||keys.d||keys.D) dir+=1;
    if(dir) S.tgt = clamp(S.tgt + dir*9*dt, -6.4, 6.4);
    S.px += (S.tgt-S.px) * Math.min(1, dt*7.5);
    S.spawn -= dt;
    if(S.spawn<=0 && !S.eclipsing){ spawnRow(ramp); S.spawn = lerp(1.35,0.42,ramp) * (0.8+RND()*0.45); }
    if(S.invuln>0) S.invuln-=dt;
  } else {
    S.dist += S.speed*dt;                       // le galop continue, à vide
    S.tgt += (0-S.tgt)*Math.min(1,dt*2);
    S.px += (S.tgt-S.px)*Math.min(1,dt*4);
  }

  // Saut : intégration verticale.  L'ombre suit d'elle-même — elle est calculée,
  // donc elle se décolle des pattes et rétrécit pendant le vol.
  if(S.z>0 || S.vz>0){ S.vz -= 19.0*dt; S.z += S.vz*dt; if(S.z<=0){ S.z=0; S.vz=0; } }
  S.lean += ((S.tgt-S.px)*0.7 - S.lean)*Math.min(1,dt*6);
  S.camX += (S.px*0.55 - S.camX)*Math.min(1,dt*5);
  S.gallop = (S.gallop + dt*S.speed*0.19) % 1;
  if(S.shake>0) S.shake=Math.max(0,S.shake-dt*2.4);
  if(S.flash>0) S.flash=Math.max(0,S.flash-dt*2);

  for(const c of S.cacti) c.y -= S.speed*dt;
  for(let i=S.cacti.length-1;i>=0;i--){
    const c=S.cacti[i];
    if(c.y < -12){ S.cacti.splice(i,1); continue; }
    const ctop = 1.9*c.s;                       // on passe au-dessus si on saute assez haut
    if(!c.hit && S.mode==="play" && S.invuln<=0 && Math.abs(c.y)<0.85
       && Math.abs(c.x-S.px) < 0.52+0.42*c.s && S.z < ctop-0.55){
      c.hit=1; S.lives--; S.invuln=1.6; S.shake=1; S.flash=1; paintLives();
      S.dist=Math.max(0,S.dist-8);
      if(S.lives<=0){ gameOver(); }
    }
  }
}

// ---------- rendu ----------
const elDist=document.getElementById("dist"), elPhase=document.getElementById("phase"),
      elDay=document.getElementById("day");

function draw(){
  // --- ciel : les astres ne dépendent que de l'heure, jamais du cavalier
  const SK = S.sky, lS=SK.lS, lM=SK.lM, altS=SK.altS, altM=SK.altM;
  const kMoon=SK.kMoon, cover=SK.cover, total=SK.total, elong=SK.elong, dayT=SK.dayT;

  const shakeX = S.shake? (RND()*2-1)*6*S.shake : 0;
  setCam(S.camX + shakeX*0.02, S.px*0.35);
  const hz = HORIZON;

  // Sources ponctuelles en orbite autour du dino
  const ORB=[S.px, 0, 1.20];
  const LposS = add(ORB, mul(lS, R_SUN_W)), LposM = add(ORB, mul(lM, R_MOON_W));

  // Même règle pour les deux : on ne les voit que dans leur quart MONTANT,
  // devant le dino, de l'horizon au bord haut du cadre. Le Soleil se lève et
  // sort par le haut ; la Lune fait pareil, 1,2 fois plus vite — mais elle
  // n'est visible que la nuit, et c'est un fondu qui la révèle, en marche.
  const pS = skyPt(lS,hz), pM = skyPt(lM,hz);
  const rS = R_SUN_PX, rM = R_MOON_PX;
  const eclUp = clamp(altS*5,0,1);                      // l'éclipse n'assombrit que de jour
  const ecl = (cover<=0.5 ? 0 : Math.pow((cover-0.5)/0.5, 1.3)) * eclUp;
```

```
const cl = 1 - 0.80*S.wx.cloud;              // le couvert éteint la lumière
let dayF = clamp((altS*180/Math.PI + 6)/14, 0, 1) * (1-0.95*ecl);
const sunI = altS>0 ? Math.pow(Math.sin(altS),0.7)*(1-cover)*cl : 0;
const warm = mix([255,176,52],[255,232,120], clamp((altS*180/Math.PI-2)/22,0,1));  // Soleil jaune
const Isun = altS>0 ? mul([warm[0]/255,warm[1]/255,warm[2]/255], 1.05*sunI) : [0,0,0];
const mi = altM>0 ? 0.34*kMoon*Math.pow(Math.sin(altM),0.5)*(1-dayF) : 0;   // clair de lune
const Imoon=[mi*0.86,mi*0.93,mi];
const ambL = (0.17 + 0.40*dayF*(0.45+0.55*cl) + 0.055*ecl) + 0.55*S.glare;            // la couronne éclaire
 ↪  un peu
const amb = [ambL*(0.92+0.10*dayF), ambL*0.95, ambL*(1.20-0.18*dayF)];

let zen, hor;
const e=altS*180/Math.PI;
if(e>12){ zen=[78,132,196]; hor=[176,202,222]; }
else if(e>-2){ const t=(e+2)/14; zen=mix([36,46,92],[78,132,196],t); hor=mix([228,150,72],[176,202,222],t); }
else if(e>-14){ const t=(e+14)/12; zen=mix([22,26,54],[36,46,92],t); hor=mix([72,48,78],[228,150,72],t); }
else { zen=[14,17,38]; hor=[34,32,58]; }
if(ecl>0){ zen=mix(zen,[15,17,44],ecl); hor=mix(hor,[104,64,72],ecl*0.92); }
if(S.wx.cloud>0.01){                         // couvert : le ciel se ferme
  const g=S.wx.cloud*0.85;
  zen=mix(zen, mix([54,56,70],[146,148,158],dayF), g);
  hor=mix(hor, mix([44,46,58],[176,174,178],dayF), g);
}
const g=ctx.createLinearGradient(0,0,0,Math.max(1,hz));
g.addColorStop(0,rgb(zen)); g.addColorStop(1,rgb(hor));
ctx.fillStyle=g; ctx.fillRect(0,0,W,Math.max(0,hz));

if(dayF<0.72){
  const a=Math.pow(1-dayF/0.72, 1.4);
  ctx.fillStyle="#fffbe8";
  for(const st of STARS){
    const y=st.y*hz; if(y>hz-2) continue;
    const tw=0.78+0.22*Math.sin(S.t*1.7+st.tw);
    ctx.globalAlpha=a*st.m*tw*(1-S.wx.cloud);
    ctx.fillRect((st.x+S.camX*6+W)%W, y, st.s, st.s);
  }
  ctx.globalAlpha=1;
}

// --- aurores : des rideaux qui ondulent, verts en bas, violets en haut.
// Par nuit claire seulement ; le jour et les nuages les éteignent.
function drawAurora(){
  const a = S.wx.aurora * clamp(1-dayF/0.45,0,1) * (1-S.wx.cloud);
  if(a<0.02) return;
  ctx.save(); ctx.globalCompositeOperation="lighter";
  for(let k=0;k<4;k++){
    const base = hz*(0.10+0.09*k), amp = 14+6*k, hgt = hz*(0.30+0.06*k);
    const g=ctx.createLinearGradient(0,base-hgt,0,base+amp);
    g.addColorStop(0,`rgba(150,80,220,0)`);
    g.addColorStop(0.45,`rgba(110,120,230,${(0.10*a).toFixed(3)})`);
    g.addColorStop(1,`rgba(80,255,160,${(0.22*a).toFixed(3)})`);
    ctx.fillStyle=g; ctx.beginPath();
    for(let x=-20;x<=W+20;x+=16){
      const y = base + amp*Math.sin(x/(70+9*k) + S.t*(0.55+0.15*k) + k*1.7)
                     + 6*Math.sin(x/23 - S.t*1.3 + k);
      x===-20 ? ctx.moveTo(x,y) : ctx.lineTo(x,y);
    }
    for(let x=W+20;x>=-20;x-=16){
      const y = base - hgt + amp*0.6*Math.sin(x/(90+9*k) + S.t*0.4 + k*2.1);
      ctx.lineTo(x,y);
    }
    ctx.closePath(); ctx.fill();
  }
  ctx.restore();
}
```

```
  // --- disques nus : ni cercle d'orbite, ni rayons.  Soleil jaune, Lune blanche.
  function drawSun(){
    if(!pS) return;
    ctx.globalAlpha = 1;                                          // c'est le CADRE qui le coupe, en haut
    ctx.fillStyle=rgb(mix([255,150,44],[255,226,74], clamp((e+4)/22,0,1)));
    ctx.beginPath(); ctx.arc(pS[0],pS[1],rS,0,7); ctx.fill();
    ctx.globalAlpha=1;
  }
  function drawCorona(){
    if(!pM || cover<0.62) return;
    const t=clamp((cover-0.62)/0.38,0,1)*(0.22+0.78*SK.slow);
    ctx.save(); ctx.globalCompositeOperation="lighter";
    const g1=ctx.createRadialGradient(pS[0],pS[1],rS*0.97,pS[0],pS[1],rS*3.1);
    g1.addColorStop(0,`rgba(255,252,242,${(0.60*t).toFixed(3)})`);
    g1.addColorStop(0.16,`rgba(224,232,255,${(0.22*t).toFixed(3)})`);
    g1.addColorStop(1,"rgba(170,196,255,0)");
    ctx.fillStyle=g1; ctx.beginPath(); ctx.arc(pS[0],pS[1],rS*3.1,0,7); ctx.fill();
    ctx.translate(pS[0],pS[1]);
    for(let i=0;i<CORONA.length;i++){
      const c=CORONA[i], a=c.a+Math.sin(S.t*0.22+i)*0.014;
      const L=rS*(1.2+c.l*2.6*(0.7+0.3*t)), w=c.w;
      ctx.beginPath();
      ctx.moveTo(Math.cos(a-w)*rS*0.98, Math.sin(a-w)*rS*0.98);
      ctx.quadraticCurveTo(Math.cos(a)*L*0.55, Math.sin(a)*L*0.55, Math.cos(a+c.b)*L, Math.sin(a+c.b)*L);
      ctx.quadraticCurveTo(Math.cos(a)*L*0.55, Math.sin(a)*L*0.55, Math.cos(a+w)*rS*0.98,
↪ Math.sin(a+w)*rS*0.98);
      ctx.closePath();
      ctx.fillStyle=`rgba(236,243,255,${(0.115*t*c.o).toFixed(3)})`;
      ctx.fill();
    }
    if(total){
      ctx.beginPath(); ctx.arc(0,0,rS*1.04,0,7);
      ctx.lineWidth=Math.max(1.2,rS*0.06); ctx.strokeStyle="rgba(255,96,86,.55)"; ctx.stroke();
    }
    ctx.restore();
  }
  // De jour, on n'affiche pas la Lune : seulement la part commune aux deux
  // disques, en noir.  Découpe au disque solaire, puis remplissage du disque
  // lunaire — il ne reste que la morsure.
  function drawBite(){
    if(!pS||!pM) return;
    ctx.save();
    ctx.beginPath(); ctx.arc(pS[0],pS[1],rS,0,7); ctx.clip();
    ctx.fillStyle="#000";
    ctx.beginPath(); ctx.arc(pM[0],pM[1],rM,0,7); ctx.fill();
    ctx.restore();
  }
  function drawDiamond(){
    if(!pS||!pM||total||cover<0.965) return;
    const t=(cover-0.965)/0.035, dx=pS[0]-pM[0], dy=pS[1]-pM[1], d=Math.hypot(dx,dy)||1;
    const x=pM[0]+dx/d*rM*0.99, y=pM[1]+dy/d*rM*0.99, r=rM*0.85*t;
    ctx.save(); ctx.globalCompositeOperation="lighter";
    const gd=ctx.createRadialGradient(x,y,0,x,y,r);
    gd.addColorStop(0,"rgba(255,255,250,.95)"); gd.addColorStop(.25,"rgba(255,246,222,.55)");
    gd.addColorStop(1,"rgba(255,230,180,0)");
    ctx.fillStyle=gd; ctx.beginPath(); ctx.arc(x,y,r,0,7); ctx.fill(); ctx.restore();
  }
  function drawMoon(veil){
    if(!pM) return;
    ctx.globalAlpha = veil;
    // Le disque entier doit se lire, sinon un croissant qui monte ressemble à
    // une lueur qui grossit au lieu d'un astre qui se lève : la part sombre
    // garde sa lumière cendrée et un liseré.
    // Extinction atmosphérique : rousse et sourde à l'horizon, franche au zénith.
    // C'est ce qui rend le lever lisible : en lune croissante le limbe éclairé
```

```
  // pointe vers le bas — vers le Soleil couché — donc la première chose qui
  // sort de l'horizon est la part sombre du disque.
  const low = clamp(1 - altM/0.22, 0, 1);
  ctx.fillStyle = rgb(mix([59,63,86],[96,58,44],low));
  ctx.beginPath(); ctx.arc(pM[0],pM[1],rM,0,7); ctx.fill();
  ctx.lineWidth=1.3; ctx.strokeStyle=rgb(mix([86,91,120],[142,88,62],low)); ctx.stroke();
  if(kMoon>0.02){
    const pa = SK.pa, ct = 1-2*kMoon;   // +1 : rien d'éclairé (nouvelle) ; −1 : tout (pleine)
    ctx.save(); ctx.translate(pM[0],pM[1]); ctx.rotate(pa);
    ctx.beginPath();
    for(let i=0;i<=40;i++){ const a=-Math.PI/2+Math.PI*i/40; const x=Math.cos(a)*rM,y=Math.sin(a)*rM;
     ↪  i?ctx.lineTo(x,y):ctx.moveTo(x,y); }
    for(let i=0;i<=40;i++){ const a=Math.PI/2-Math.PI*i/40; ctx.lineTo(Math.cos(a)*rM*ct, Math.sin(a)*rM); }
    ctx.closePath(); ctx.fillStyle=rgb(mix([253,253,246],[246,172,104],low)); ctx.fill(); ctx.restore();
  }
  ctx.globalAlpha = 1;
}
// Le Soleil de jour, la Lune de nuit — jamais les deux.  La seule exception
// est l'éclipse : la Lune n'est alors qu'une silhouette devant le Soleil.
function drawClouds(){
  const wc=S.wx.cloud; if(wc<0.02) return;
  const lit  = mix(mix([44,46,60],[238,240,246],dayF), [78,80,92], S.wx.rain*0.8);
  const dark = mix(mix([24,26,38],[152,154,170],dayF), [50,52,62], S.wx.rain*0.7);
  const n = Math.round(CLOUDS.length*clamp(wc*1.15,0,1));
  ctx.globalAlpha = clamp(wc*1.25,0,1)*0.92;
  for(let i=0;i<n;i++){
    const c=CLOUDS[i];
    const x=((((c.x + S.t*c.sp)%1)+1)%1)*(W+320)-160, y=hz - c.alt*K_SKY;
    if(y>hz-4) continue;
    for(const q of c.puffs){
      ctx.fillStyle = rgb(q.dy<0 ? lit : dark);
      ctx.beginPath();
      ctx.ellipse(x+q.dx*c.sc, y+q.dy*c.sc, q.r*c.sc, q.r*c.sc*0.62, 0, 0, 7);
      ctx.fill();
    }
  }
  ctx.globalAlpha=1;
}
// L'arc-en-ciel se forme à l'opposé du Soleil : il n'apparaît donc que
// l'après-midi, quand le Soleil est passé derrière le dino — pile quand on
// ne le voit pas.  Centre au point antisolaire, rayon 42°.
function drawRainbow(){
  const r=S.wx.rain;
  // Il suffit d'un peu de pluie et d'un Soleil dans le dos ; seuls les
  // déluges d'orage l'effacent.
  const st = clamp(Math.min(r/0.15,(0.82-r)/0.20),0,1) * clamp(altS/0.12,0,1) * (lS[1]<0?1:0);
  if(st<=0.02) return;
  const vx=W/2 + FOC*RGT[1]/(FWD[1]||1e-6), cy=hz + altS*K_SKY, R=0.733*K_SKY;
  const cols=[[216,70,58],[226,134,52],[228,208,78],[96,180,96],[68,140,208],[88,84,182],[134,72,170]];
  ctx.save(); ctx.globalCompositeOperation="lighter"; ctx.lineWidth=4.4;
  cols.forEach((c,i)=>{
    ctx.strokeStyle=`rgba(${c[0]},${c[1]},${c[2]},${(0.48*st).toFixed(3)})`;
    ctx.beginPath(); ctx.arc(vx, cy, R-i*4.4, Math.PI, 2*Math.PI); ctx.stroke();
    ctx.strokeStyle=`rgba(${c[0]},${c[1]},${c[2]},${(0.18*st).toFixed(3)})`;
    ctx.beginPath(); ctx.arc(vx, cy, R*1.28+i*4.4, Math.PI, 2*Math.PI); ctx.stroke();  // arc secondaire
  });
  ctx.restore();
}
function drawMeteors(){
  if(!S.meteors.length) return;
  ctx.save(); ctx.globalCompositeOperation="lighter"; ctx.lineCap="round";
  for(const m of S.meteors){
    const k = 1 - m.t/m.life, hx = m.x + m.vx*m.t, hy = m.y + m.vy*m.t;
    const d = Math.hypot(m.vx,m.vy)||1, tx = hx - m.vx/d*m.len, ty = hy - m.vy/d*m.len;
    const g = ctx.createLinearGradient(tx,ty,hx,hy);
    g.addColorStop(0,"rgba(255,250,230,0)"); g.addColorStop(1,`rgba(255,250,230,${(0.9*k).toFixed(3)})`);
```

```
      ctx.strokeStyle=g; ctx.lineWidth=m.w;
      ctx.beginPath(); ctx.moveTo(tx,ty); ctx.lineTo(hx,hy); ctx.stroke();
    }
    ctx.restore();
  }
  // Le père Noël : quatre rennes au galop, le traîneau, le sac, et une
  // traînée d'or derrière. Rudolphe ouvre la marche, le nez allumé.
  function drawSanta(u){
    if(u<0) return;
    const x0 = W+180 - u*(W+420), y0 = hz*0.26 + hz*0.12*u + 10*Math.sin(u*11);
    const gal = S.t*9;
    ctx.save(); ctx.translate(x0,y0);
    ctx.globalCompositeOperation="lighter";
    for(let i=0;i<16;i++){
      const t=i/16, tx=42+t*170, ty=4+Math.sin(u*11-t*4)*9;
      ctx.fillStyle=`rgba(255,214,110,${(0.5*(1-t)).toFixed(2)})`;
      ctx.beginPath(); ctx.arc(tx,ty,2+t*3.5,0,7); ctx.fill();
    }
    ctx.globalCompositeOperation="source-over";
    const N=4, brun="#6b4a2f";
    ctx.strokeStyle="#8a6a48"; ctx.lineWidth=1.1;
    ctx.beginPath(); ctx.moveTo(-8,-4); ctx.lineTo(-40-(N-1)*40, -4-(N-1)*3); ctx.stroke();  // les rênes
    for(let i=0;i<N;i++){
      const rx=-40-i*40, ry=-i*3 + Math.sin(gal+i*1.3)*3;
      ctx.fillStyle=brun; ctx.beginPath(); ctx.ellipse(rx,ry,13,6,0,0,7); ctx.fill();          // corps
      ctx.beginPath(); ctx.ellipse(rx-15,ry-6,5,3.6,-0.3,0,7); ctx.fill();                     // tête
      ctx.lineWidth=2.4; ctx.strokeStyle=brun; ctx.beginPath(); ctx.moveTo(rx-10,ry-2); ctx.lineTo(rx-14,ry-6);
↪ ctx.stroke();  // cou
      ctx.lineWidth=1.2; ctx.strokeStyle="#c9a77a";                                            // bois
      ctx.beginPath(); ctx.moveTo(rx-16,ry-9); ctx.lineTo(rx-20,ry-17); ctx.moveTo(rx-16,ry-9);
↪ ctx.lineTo(rx-12,ry-17);
      ctx.moveTo(rx-18,ry-13); ctx.lineTo(rx-22,ry-15); ctx.moveTo(rx-14,ry-13); ctx.lineTo(rx-10,ry-15);
↪ ctx.stroke();
      ctx.lineWidth=2; ctx.strokeStyle=brun;                                                   // pattes au galop
      for(let k=0;k<4;k++){
        const ph=gal+k*1.57+i, lx=rx-8+k*5.5, ang=Math.sin(ph)*0.75;
        ctx.beginPath(); ctx.moveTo(lx,ry+3); ctx.lineTo(lx+Math.sin(ang)*8, ry+3+Math.cos(ang)*8);
↪ ctx.stroke();
      }
      if(i===N-1){                                                                             // Rudolphe
        const nz=0.65+0.35*Math.sin(S.t*8);
        ctx.fillStyle=`rgba(255,60,50,${nz.toFixed(2)})`; ctx.beginPath(); ctx.arc(rx-20,ry-6,2.4,0,7);
↪ ctx.fill();
        ctx.globalCompositeOperation="lighter"; ctx.fillStyle=`rgba(255,60,50,${(0.28*nz).toFixed(2)})`;
        ctx.beginPath(); ctx.arc(rx-20,ry-6,8,0,7); ctx.fill(); ctx.globalCompositeOperation="source-over";
      }
    }
    ctx.fillStyle="#b2302e"; ctx.beginPath();                                                  // le traîneau
    ctx.moveTo(-10,-8); ctx.lineTo(28,-8); ctx.lineTo(30,6); ctx.lineTo(-6,6);
↪ ctx.quadraticCurveTo(-18,6,-14,-14); ctx.closePath(); ctx.fill();
    ctx.strokeStyle="#e0b25a"; ctx.lineWidth=2;                                                // le patin
    ctx.beginPath(); ctx.moveTo(-16,10); ctx.lineTo(34,10); ctx.quadraticCurveTo(40,10,38,4); ctx.stroke();
    ctx.fillStyle="#d9c9a5"; ctx.beginPath(); ctx.arc(20,-13,8,0,7); ctx.fill();               // le sac
    ctx.fillStyle="#c8322f"; ctx.beginPath(); ctx.arc(6,-14,7,0,7); ctx.fill();                // le bonhomme
    ctx.fillStyle="#f2d5b8"; ctx.beginPath(); ctx.arc(6,-24,4.2,0,7); ctx.fill();
    ctx.fillStyle="#f6f1e8"; ctx.beginPath(); ctx.arc(6,-21,4.6,0.15,Math.PI-0.15); ctx.fill();  // la barbe
    ctx.fillStyle="#c8322f"; ctx.beginPath(); ctx.moveTo(1,-26); ctx.lineTo(11,-26); ctx.lineTo(14,-33);
↪ ctx.closePath(); ctx.fill();
    ctx.fillStyle="#fff"; ctx.beginPath(); ctx.arc(14,-33,1.8,0,7); ctx.fill();
    ctx.restore();
  }
  ctx.save();                                  // rien ne déborde sous l'horizon
  ctx.beginPath(); ctx.rect(0,0,W,Math.max(0,hz)); ctx.clip();
  // Le Soleil se lève devant et sort par le haut du cadre. La Lune, en
  // marche elle aussi, n'existe que la nuit : elle se révèle en fondu quand le
  // Soleil passe sous l'horizon (voile), et s'efface en fondu à l'aube.
```

```
drawMeteors();
drawAurora();
if(altS>-0.05) drawSun();
if(cover>0.02){ drawCorona(); drawBite(); drawDiamond(); }   // de jour : l'intersection, en noir
const veil = clamp(-altS/0.30, 0, 1);        // 0 au coucher, 1 par nuit noire
if(veil>0.01) drawMoon(veil);
drawClouds(); drawRainbow();
drawSanta(S.santa);
if(S.bolt){                                  // l'éclair, dans le ciel
  ctx.strokeStyle="rgba(255,252,238,.92)"; ctx.lineWidth=2.4;
  ctx.beginPath(); ctx.moveTo(S.bolt.seg[0][0],S.bolt.seg[0][1]);
  for(const q of S.bolt.seg) ctx.lineTo(q[0],q[1]);
  ctx.stroke();
  ctx.strokeStyle="rgba(190,214,255,.30)"; ctx.lineWidth=7; ctx.stroke();
}
ctx.restore();

// --- sol
let near=mix([88,70,52],[196,154,106],dayF), far=mix([54,47,44],[172,138,100],dayF);
if(S.wx.snow>0.02){ const w=S.wx.snow*0.75; near=mix(near,mix([120,124,140],[232,236,244],dayF),w);
 ↪  far=mix(far,mix([96,100,116],[214,220,232],dayF),w); }
const g2=ctx.createLinearGradient(0,hz,0,H);
g2.addColorStop(0,rgb(far)); g2.addColorStop(1,rgb(near));
ctx.fillStyle=g2; ctx.fillRect(0,Math.max(0,hz),W,H-hz);

// bandes de vitesse
const BL=7, off=(S.dist%BL);
ctx.fillStyle=`rgba(0,0,0,${0.06+0.05*dayF})`;
for(let k=0;k<26;k++){
  const y0=k*BL*2 - off, y1=y0+BL;
  if(y1<0.5) continue;
  const q=[proj([-46,Math.max(y0,0.5),0]),proj([46,Math.max(y0,0.5),0]),proj([46,y1,0]),proj([-46,y1,0])];
  if(q.some(v=>!v)) continue;
  ctx.beginPath(); ctx.moveTo(q[0][0],q[0][1]); for(let i=1;i<4;i++) ctx.lineTo(q[i][0],q[i][1]); ctx.fill();
}
// piste
ctx.fillStyle=`rgba(${mix([120,105,88],[224,206,178],dayF).map(v=>v|0).join(",")},.5)`;
for(const sgn of [-1,1]) for(let k=0;k<34;k++){
  const y0=k*4.2-(S.dist%4.2), y1=y0+2.4;
  if(y1<0.4) continue;
  const q=[proj([sgn*2.6-0.08,Math.max(y0,0.4),0.01]),proj([sgn*2.6+0.08,Math.max(y0,0.4),0.01]),
           proj([sgn*2.6+0.08,y1,0.01]),proj([sgn*2.6-0.08,y1,0.01])];
  if(q.some(v=>!v)) continue;
  ctx.beginPath(); ctx.moveTo(q[0][0],q[0][1]); for(let i=1;i<4;i++) ctx.lineTo(q[i][0],q[i][1]); ctx.fill();
}
// cailloux
ctx.fillStyle=`rgba(${mix([58,47,38],[150,118,84],dayF).map(v=>v|0).join(",")},.85)`;
for(const r of ROCKS){
  const y=((r.y0-S.dist)%140+140)%140;
  const p=proj([r.x,y,0]); if(!p||p[1]<hz) continue;
  const s=FOC*r.r/p[2];
  ctx.beginPath(); ctx.ellipse(p[0],p[1],s,s*0.4,0,0,7); ctx.fill();
}

// --- les objets du monde
const boxes = dinoBoxes(S.gallop, S.px, S.lean, S.z);
const nearCacti = S.cacti.filter(c=>c.y>-6 && c.y<62);
for(const c of nearCacti) boxes.push(...cactusBoxes(c));

// --- OMBRES CALCULÉES : S = Q − l·(Qz/lz), union en un seul tracé
const shC = 1 - 0.92*S.wx.cloud;             // sous les nuages, plus d'ombre portée
for(const [Lp,alt,base,tint] of [[LposS,altS,0.52*(1-cover)*shC,"14,10,20"],
                                 [LposM,altM,0.34*kMoon*(1-dayF)*shC,"10,16,38"]]){
  if(alt<=0.04 || base<0.02) continue;
  const fade=Math.min(1, Math.sin(alt)*3.0), a=base*fade;
  if(a<0.02) continue;
```

```
    ctx.beginPath();
    for(const b of boxes){
      const gp=[];
      let ok=true;
      for(const q of verts(b)){
        const qz=Math.max(q[2],0);
        if(Lp[2] <= qz+0.25){ ok=false; break; }      // source sous le sommet : rien au sol
        const t=qz/(Lp[2]-qz);                        // intersection de (L,Q) avec z=0
        const gx=q[0]+(q[0]-Lp[0])*t, gy=q[1]+(q[1]-Lp[1])*t;
        if(Math.hypot(gx-q[0], gy-q[1]) > 70){ ok=false; break; }   // ombre rasante
        const p=proj([gx,gy,0]);
        if(!p){ ok=false; break; }
        gp.push([p[0],p[1]]);
      }
      if(!ok||gp.length<3) continue;
      const hl=hull(gp);
      if(hl.length<3) continue;
      ctx.moveTo(hl[0][0],hl[0][1]);
      for(let i=1;i<hl.length;i++) ctx.lineTo(hl[i][0],hl[i][1]);
      ctx.closePath();
    }
    ctx.fillStyle=`rgba(${tint},${a.toFixed(3)})`;
    ctx.fill("nonzero");
  }

  // --- faces, algorithme du peintre
  const F=[];
  for(const b of boxes) faces(b,F);
  const drawn=[];
  for(const f of F){
    const cen=mul(add(add(f.q[0],f.q[1]),add(f.q[2],f.q[3])),0.25);
    const d=sub(cen,EYE);
    if(dot(f.n,d)>=0) continue;
    const p=[proj(f.q[0]),proj(f.q[1]),proj(f.q[2]),proj(f.q[3])];
    if(p.some(v=>!v)) continue;
    const dS=norm(sub(LposS,cen)), dM=norm(sub(LposM,cen));   // direction vers l'astre
    const nd1=Math.max(0,dot(f.n,dS)), nd2=Math.max(0,dot(f.n,dM));
    const col=[
      f.col[0]*(amb[0]+Isun[0]*nd1+Imoon[0]*nd2),
      f.col[1]*(amb[1]+Isun[1]*nd1+Imoon[1]*nd2),
      f.col[2]*(amb[2]+Isun[2]*nd1+Imoon[2]*nd2)];
    drawn.push({z:dot(d,FWD), p, c:rgb([Math.min(255,col[0]),Math.min(255,col[1]),Math.min(255,col[2])])});
  }
  drawn.sort((a,b)=>b.z-a.z);
  const blink = S.invuln>0 && (Math.floor(S.t*12)%2===0);
  for(const f of drawn){
    ctx.globalAlpha = blink ? 0.55 : 1;
    ctx.fillStyle=f.c;
    ctx.beginPath(); ctx.moveTo(f.p[0][0],f.p[0][1]);
    for(let i=1;i<4;i++) ctx.lineTo(f.p[i][0],f.p[i][1]);
    ctx.fill();
  }
  ctx.globalAlpha=1;

  // --- poussière
  for(let j=0;j<7 && S.z<0.05;j++){
    const t=((S.gallop*3 + j*0.7)%3);
    const p=proj([S.px+Math.sin(j*2.1)*0.9, -1.3-t*1.1, 0.22+t*0.16]);
    if(!p) continue;
    const s=FOC*(0.12+t*0.13)/p[2];
    ctx.fillStyle=`rgba(${mix([70,60,52],[190,164,132],dayF).map(v=>v|0).join(",")},${(0.42-0.13*t).toFixed(2)
↪ })`;
    ctx.beginPath(); ctx.ellipse(p[0],p[1],s,s*0.7,0,0,7); ctx.fill();
  }

  // --- neige
```

```
    if(S.wx.snow>0.03){
      const sn=S.wx.snow, n=Math.round(FLAKES.length*sn);
      ctx.fillStyle=`rgba(236,242,252,${(0.45+0.45*sn).toFixed(2)})`;
      for(let i=0;i<n;i++){
        const f=FLAKES[i];
        const y=((f.y + S.t*f.v)%(H+20))-10;
        const x=((f.x + f.sw*Math.sin(S.t*0.9+f.ph) + y*0.06)%W+W)%W;
        ctx.beginPath(); ctx.arc(x,y,f.r,0,7); ctx.fill();
      }
    }
    // --- pluie
    if(S.wx.rain>0.03){
      const r=S.wx.rain, n=Math.round(DROPS.length*r);
      ctx.strokeStyle=`rgba(196,212,232,${(0.16+0.30*r).toFixed(2)})`;
      ctx.lineWidth=1.1; ctx.beginPath();
      for(let i=0;i<n;i++){
        const d=DROPS[i];
        const y=((d.y + S.t*d.v)%(H+40))-20, x=((d.x - y*0.24)%W+W)%W;
        ctx.moveTo(x,y); ctx.lineTo(x+d.l*0.24, y+d.l);
      }
      ctx.stroke();
    }
    if(S.glare>0){ ctx.fillStyle=`rgba(226,234,255,${(0.42*S.glare).toFixed(3)})`; ctx.fillRect(0,0,W,H); }

    // --- vignette + flash de choc
    const vg=ctx.createRadialGradient(W/2,H/2,H*0.42,W/2,H/2,H*0.95);
    vg.addColorStop(0,"rgba(0,0,0,0)"); vg.addColorStop(1,"rgba(0,0,0,.42)");
    ctx.fillStyle=vg; ctx.fillRect(0,0,W,H);
    if(S.flash>0){ ctx.fillStyle=`rgba(190,70,50,${(S.flash*0.30).toFixed(3)})`; ctx.fillRect(0,0,W,H); }

    // --- HUD
    duck(total);
    elDist.textContent = Math.round(S.dist);
    const hrs=(dayT%1)*24;
    const edeg = elong*180/Math.PI;
    const nm = kMoon<0.03 ? "nouvelle lune" : kMoon<0.28 ? "croissant"
             : kMoon<0.58 ? "quartier" : kMoon<0.92 ? "gibbeuse" : "pleine lune";
    elPhase.textContent = total ? "éclipse totale"
      : cover>0.02 ? `éclipse · ${(cover*100).toFixed(0)} %`
      : `${edeg.toFixed(0)}° — ${nm}`;
    elPhase.style.color = cover>0.02 ? "#e0653a" : "";
    const rem = ((ECL_DAY - dayT) % MOON_PER + MOON_PER) % MOON_PER;
    const an = Math.floor(dayT/SEASON)+1, doy = Math.floor(yearT(dayT))+1;
    elDay.textContent = `an ${an} · jour ${doy}/${SEASON} · ${String(Math.floor(hrs)).padStart(2,"0")} h
↪   ${String(Math.floor(hrs%1*60)).padStart(2,"0")} · éclipse dans ${rem.toFixed(1)} j`;
    if(S.cardT>0){
      elCard.hidden=false;
      elCard.style.opacity = clamp(Math.min(S.cardT, 3.6-S.cardT)*2.2, 0, 1).toFixed(2);
      elCardLvl.textContent = S.cardTitle;
      elCardSub.textContent = S.cardSub;
    } else if(!elCard.hidden) elCard.hidden=true;

  }

  let last=performance.now();
  function loop(now){
    const dt=Math.min(0.05,(now-last)/1000); last=now;
    update(dt); draw();
    requestAnimationFrame(loop);
  }
  paintLives();
  requestAnimationFrame(loop);
})();
</script>

</body></html>
```

3.71 /opt/GSM/osmo-operator/configs/conky/labgsm.html

11354 octets, 193 lignes → 193 lignes

```
<!doctype html><html><head><meta charset="utf-8">
<title>LAB GSM</title>
<style>
  /* [2026-10-01] LISIBLE, EN UBUNTU, ET VIVANTE.
     - Les tailles sont en vw (largeur de la zone, 1110 px de reference), plus en
       vh : la fenetre est rabattue sur la hauteur du contenu (osmo-topzone.py,
       INJECT_CSS), et des polices en vh de 500 px donnaient 7 a 10 px de texte.
     - Tailles calees pour que la carte (TOP_BOX y=60) S ARRETE AU-DESSUS de
       l encart (y=220) : sans titre, elle tient en ~160 px de reference.
     - Police Ubuntu partout (fonts-ubuntu est dans l image, 80-chroot.sh).
     - Une rangee « banc en temps reel » sous les timeslots : coeur, radio,
       abonnes, appels, depuis quand - poussee par osmo-ts-probe.py dans
       timeslots.json (core / radio / phy / subs / calls / up). */
  :root{
    --ink:#131018; --ink2:#1c1822; --card:rgba(13,11,20,0);
    --bone:#f4eddc; --dim:#c9bfab; --violet:#9d8cf0; --violet2:#6f5fc0;
    --teal:#4fe0b6; --moon:#b4c2ea; --orange:#ff9a3c; --pink:#ff4d6d;
    --ok:#3fb950; --ko:#f85149;
    --line:rgba(244,237,220,.18);
  }
  *{box-sizing:border-box}
  html,body{margin:0;height:100%;background:transparent;color:var(--bone);text-shadow:0 1px 2px rgba(0,0,0,.9),0
↪  0 8px rgba(0,0,0,.7);
    font-family:"Ubuntu","Ubuntu Sans",sans-serif;overflow:hidden}
  #card{position:absolute;inset:0;border:1px solid var(--line);border-radius:18px;
    background:var(--card);backdrop-filter:blur(3px);padding:1vw 1.8vw .9vw;
    display:flex;flex-direction:column;gap:.45vw;transition:opacity .6s ease}
  #card.faded{opacity:0;pointer-events:none}
  /* [2026-10-12] AU REPOS, PAS DE BANNIERE : LE TITRE. Tant qu aucun banc ne
     tourne (osmo-panel.py ecrit 0 dans $XDG_RUNTIME_DIR/osmo-banc.etat, que
     osmo-topzone.py relaie par osmoBanc(false)), la carte est invisible et la
     zone porte seulement « Telecom Operator », en grand. Au lancement de la 2G
     (ou de la 4G), le titre s efface en fondu pendant que la carte apparait -
     les deux a la fois, meme duree. body.instant : sans animation (premier
     etat pousse au chargement de la page, pour ne pas voir le titre clignoter
     devant un banc deja debout). */
  /* [2026-10-23] PLUS DE TITRE au repos : le fond (gsm-lab-ville.jpg) porte le sien,
     peint, « TELECOM OPERATOR » ; le notre s y superposait au logo ubuntu. La zone
     est vide au repos, la carte arrive seule en fondu. */
  body.repos #card{opacity:0;pointer-events:none;transition:opacity 1.4s ease}
  body.instant #card{transition:none!important}
  .eyebrow{display:flex;justify-content:space-between;align-items:center;
    color:var(--dim);letter-spacing:.16em;font-size:.9vw;font-weight:500}
  .eyebrow .dot{color:var(--teal)}
  .pill{border:1px solid rgba(79,224,182,.55);border-radius:.8vw;padding:.3vw .9vw;color:var(--teal);
    font-family:"Ubuntu Mono",monospace;letter-spacing:.06em;font-size:.95vw}
  h1{margin:0;font-family:"Ubuntu","Ubuntu Sans",sans-serif;font-weight:700;
    font-size:5vw;line-height:1;letter-spacing:.01em;
    background:linear-gradient(92deg,var(--violet),#c4b6ff 60%,var(--moon));
    -webkit-background-clip:text;background-clip:text;color:transparent}
  .lede{color:#e6ddcb;font-size:1.35vw;line-height:1.4;max-width:78%}
  .lede b{color:#fff;font-weight:500}
  .meta{color:var(--dim);letter-spacing:.12em;font-size:.85vw;font-weight:500}
  /* la rangee des timeslots : huit colonnes de meme largeur, hautes */
  #ts{display:grid;grid-template-columns:repeat(8,1fr);gap:.7vw;min-height:4.2vw}
  .ts{border:1px solid rgba(180,194,234,.3);border-radius:.8vw;padding:.5vw .4vw;
    display:flex;flex-direction:column;align-items:center;justify-content:center;gap:.5vw;
    text-align:center;background:rgba(28,24,34,.3);position:relative;gap:.3vw;
    transition:box-shadow .18s,border-color .18s,background .18s}
  .ts .n{font-weight:700;font-size:1.5vw;line-height:1}
  .ts .k{color:var(--dim);font-size:.9vw;letter-spacing:.04em;font-family:"Ubuntu Mono",monospace}
  .ts.bcch{border-color:var(--violet);box-shadow:0 0 0 1px var(--violet) inset}
```

```
  /* clignotement orange quand le timeslot porte des bursts */
  .ts.on{border-color:var(--orange);
    background:rgba(255,154,60,.16);
    box-shadow:0 0 14px rgba(255,154,60,.55),0 0 0 1px var(--orange) inset;
    animation:blink .9s steps(2,end) infinite}
  @keyframes blink{50%{box-shadow:0 0 4px rgba(255,154,60,.2),0 0 0 1px rgba(255,154,60,.4) inset;
    background:rgba(255,154,60,.05)}}
  /* ── LE BANC EN TEMPS REEL ──────────────────────────────────────── */
  #live{display:flex;gap:.6vw;flex-wrap:wrap;align-items:center;
    border-top:1px solid var(--line);padding-top:.45vw}
  .grp{display:flex;align-items:center;gap:.35vw;border:1px solid var(--line);
    border-radius:.8vw;padding:.2vw .6vw;background:rgba(28,24,34,.25);font-size:.95vw}
  .grp .lab{color:var(--dim);letter-spacing:.08em;font-size:.9vw;margin-right:.2vw;font-weight:500}
  .grp .d{font-family:"Ubuntu Mono",monospace;font-size:.95vw;color:var(--dim)}
  .grp .d.ok{color:var(--ok)} .grp .d.ko{color:var(--ko)}
  .grp b{font-weight:500;color:var(--bone)}
  .grp.warn{border-color:rgba(255,154,60,.6);color:var(--orange)}
  .grp.call{border-color:rgba(79,224,182,.7);color:var(--teal)}
  .tags{display:flex;gap:.5vw;flex-wrap:wrap;align-items:center}
  .tag{border:1px solid var(--line);border-radius:.8vw;padding:.2vw .7vw;color:var(--dim);font-size:.9vw}
  .tag b{color:var(--bone);font-weight:500}
  .foot{margin-left:auto;color:var(--dim);font-size:.85vw}
  #noconf{color:var(--orange);opacity:.9}
</style></head>
<body class="repos">
<div id="card">
  <div class="eyebrow">
    <div><span class="dot">●</span> SOFTWARE DEFINED RADIO · UM INTERFACE · GR-GSM</div>
    <div class="pill" id="arfcn">ARFCN 514 · DCS 1800</div>
  </div>
  <!-- [2026-10-01] Plus de titre « LAB GSM » ni de paragraphe de presentation :
       la banniere ne porte que le vivant - timeslots, banc, tags. -->
  <div class="meta">TRAME TDMA · 8 TIMESLOTS · 4,615 MS</div>
  <div id="ts"></div>
  <div id="live"></div>
  <div class="tags" id="tags">
    <div class="tag">IMSI <b id="imsi">001-01</b></div>
    <div class="tag" id="a5tag">A5/1 <b>✓</b></div>
    <div class="tag">COMP128v1 <b>✓</b></div>
    <div class="tag">TCH/F · GAPK fr</div>
    <div class="tag">gr-gsm · trxcon · osmo-bts-trx</div>
    <div id="noconf" class="tag" hidden>osmo-bsc non détecté — animation de démonstration</div>
    <div class="foot">bbaranoff.github.io</div>
  </div>
</div>

<script>
// Libelles courts par type de canal physique (phys_chan_config d osmo-bsc).
const LABEL = {
  "CCCH+SDCCH4":"FCCH·SCH·BCCH", "CCCH":"FCCH·SCH·BCCH", "BCCH":"BCCH",
  "SDCCH8":"SDCCH", "SDCCH8+CBCH":"SDCCH", "SDCCH":"SDCCH",
  "TCH/F":"TCH/F", "TCH/H":"TCH/H", "TCH/F_PDCH":"TCH/F·PDCH",
  "PDCH":"PDCH", "TCH/F_TCH/H_PDCH":"TCH", "NONE":"—"
};
let TS = [
  {n:0,type:"CCCH+SDCCH4"},{n:1,type:"SDCCH8"},{n:2,type:"TCH/F"},{n:3,type:"TCH/F"},
  {n:4,type:"TCH/F"},{n:5,type:"TCH/F"},{n:6,type:"TCH/F"},{n:7,type:"TCH/F"}
];
function drawTS(){
  const box=document.getElementById('ts'); box.innerHTML='';
  TS.forEach(t=>{
    const d=document.createElement('div'); d.className='ts'; d.dataset.n=t.n;
    if((t.type||'').indexOf('CCCH')>=0||(t.type||'').indexOf('BCCH')>=0) d.classList.add('bcch');
    d.innerHTML='<div class="n">TS'+t.n+'</div><div class="k">'+(LABEL[t.type]||t.type||'—')+'</div>';
    box.appendChild(d);
  });
```

```
}
// ── LA RANGEE « BANC » ──────────────────────────────────────────────
// Ordre fixe : on lit toujours les pastilles au meme endroit.
const CORE=["HLR","MSC","BSC","STP","MGW","BTS","SIP","SMSC","PBX"];
const RADIO=["FAKETRX","PONT","MOBILE","TRXCON","QEMU","GAPK"];
function esc(s){return String(s).replace(/[&<>]/g,c=>({'&':'&amp;','<':'&lt;','>':'&gt;'}[c]));}
function dots(names,etat){
  return names.map(n=>{const v=etat&&etat[n];
    return '<span class="d '+(v===undefined?'':(v?'ok':'ko'))+'">'+(v===undefined?'○':'●')+'</span>'+esc(n);
  }).join(' ');
}
function duree(s){
  s=Math.max(0,s|0); if(s<60) return s+' s';
  const m=Math.floor(s/60)%60, h=Math.floor(s/3600);
  return h?h+' h '+String(m).padStart(2,'0'):m+' min';
}
function drawLive(state){
  const box=document.getElementById('live'); if(!box) return;
  if(!state || !state.bsc){
    box.innerHTML='<div class="grp warn"><span class="lab">BANC</span> à l\'arrêt — <b>./start-native.sh</b> ou le
↪  bouton <b>run standalone</b></div>';
    return;
  }
  const s=state.subs||{}; const calls=state.calls|0;
  let h='';
  h+='<div class="grp"><span class="lab">CŒUR</span>'+dots(CORE,state.core)+'</div>';
  h+='<div class="grp"><span class="lab">RADIO</span>'+dots(RADIO,state.radio)+(state.phy?' <b>·
↪  '+esc(state.phy)+'</b>':'')+'</div>';
  h+='<div class="grp"><span class="lab">ABONNÉS</span><b>'+(s.total===undefined?'–':s.total)+'</b>'
     +(s.attached!==undefined?' · <span class="d ok">●</span> '+s.attached+'
      ↪  rattaché'+(s.attached>1?'s':''):'')+'</div>';
  h+='<div class="grp'+(calls?' call':'')+'"><span class="lab">APPELS</span><b>'+calls+'</b>'+(calls?' TCH en
↪  voix':'')+'</div>';
  if(state.up) h+='<div class="grp"><span class="lab">DEBOUT</span><b>'+duree(state.up)+'</b></div>';
  box.innerHTML=h;
}
// osmoUpdate({ts:[{n,type,active,voice}], bsc:true, arfcn:"...", core:{}, radio:{}, subs:{}, calls:n, up:s})
// - appele par osmo-topzone.py
window.osmoUpdate=function(state){
  try{
    if(state.ts && state.ts.length){ TS=state.ts; drawTS(); }
    document.getElementById('noconf').hidden = !!state.bsc;
    if(state.arfcn) document.getElementById('arfcn').textContent=state.arfcn;
    if(state.imsi) document.getElementById('imsi').textContent=state.imsi;
    if(state.a5) document.getElementById('a5tag').innerHTML='A5/'+state.a5+' <b>✓</b>';
    (state.ts||[]).forEach(t=>{
      const el=document.querySelector('.ts[data-n="'+t.n+'"]'); if(!el) return;
      // .on = ce timeslot porte des bursts (voix comprise) : c est le SEUL
      // clignotement de la carte depuis que le chronogramme est parti.
      el.classList.toggle('on', !!t.active || !!t.voice);
    });
    drawLive(state);
  }catch(e){}
};
// fondu : osmo-topzone.py bascule .faded pour laisser place a une appli
window.osmoFade=function(on){document.getElementById('card').classList.toggle('faded',!!on);};
// banc debout (on=true) : la carte ; a l arret : rien (le fond porte le titre).
// instant=true : sans fondu (etat initial, au chargement).
window.osmoBanc=function(on,instant){const b=document.body;
  if(instant){b.classList.add('instant');void b.offsetWidth;}
  b.classList.toggle('repos',!on);
  if(instant){void b.offsetWidth;b.classList.remove('instant');}};
drawTS(); drawLive(null);
// demo si personne ne pousse d etat (osmo-topzone absent) : un timeslot au hasard
let demo=setInterval(()=>{ if(window.__osmoLive) return clearInterval(demo);
  TS.forEach(t=>{const el=document.querySelector('.ts[data-n="'+t.n+'"]');
```

```
    if(el) el.classList.toggle('on', (t.type||'').indexOf('TCH')>=0 && Math.random()<0.25);});
},700);
</script>
</body></html>
```

3.72 /opt/GSM/osmo-operator/configs/conky/osmo-conky.conf

6860 octets, 103 lignes → 103 lignes

```
-- ============================================================================
--  osmo-conky.conf - le tableau de bord Conky du banc (live ET systeme installe)
-- ============================================================================
--  Lance par /etc/xdg/autostart/osmo-conky.desktop (pose par build-iso.sh dans
--  le bloc desktop) pour toute session GNOME, donc pour root sur la cle et
--  pour l utilisateur cree par Calamares sur le disque. Les lignes du banc
--  LE SELECTEUR (◄ ... ►) : le bloc central suit l arret choisi - op1, les
--  operateurs docker qui tournent, le hub inter-STP, la 4G. Conky ne recoit
--  pas les clics : on pousse les fleches avec Ctrl+AltGr+Gauche/Droite, celles
--  de l encart (tools/osmo-panel.py) ou « osmo-op --next ». La liste des
--  arrets est celle de « osmo-op --list-vue ».
--  Les lignes du banc
--  viennent de tools/conky-osmo-status.sh (execpi : la sortie est du texte
--  conky, avec ses couleurs). Chemin fixe /opt/GSM/osmo-operator : c est la
--  que vit le depot sur les deux.
--      conky -c /opt/GSM/osmo-operator/configs/conky/osmo-conky.conf
--
--  [2026-10-01] LE VISUEL. Police Ubuntu (titres, texte) et Ubuntu Mono pour ce
--  qui s aligne en colonnes (reseau, pastilles) ; plus de cadre blanc autour
--  du graphe CPU (draw_graph_borders), un degrade vert -> bleu a la place ;
--  des libelles en gris doux (color5) devant les valeurs en clair ; des
--  barres fines et longues ; un peu d air entre les sections (voffset).
--  Les blocs du script d etat (reseau, vue, services) restent en DejaVu Sans
--  Mono : ils portent ◄ ► ● ○ ↓ ↑, qu Ubuntu Mono n a pas (carres a l ecran,
--  verifie) - et Conky ne replie pas sur une autre police.
--
--  [2026-10-12] REFONTE : UBUNTU PARTOUT, UN PANNEAU, DES COULEURS.
--  - Le fond d ecran est devenu une ville sous un ciel bleu clair : le texte
--    pose a meme le ciel ne se lisait plus. osmo-conky.lua (Cairo) peint une
--    plaque sombre translucide aux coins arrondis, un filet, un trait d accent.
--    Plus d ombres portees : la plaque suffit.
--  - Police Ubuntu pour TOUT le texte (Ubuntu Mono pour ce qui s aligne en
--    colonnes). Les pastilles ● ○ et les fleches ↓ ↑ ◄ ►, qu Ubuntu n a pas,
--    sont rendues glyphe par glyphe en DejaVu Sans 9 par le script d etat
--    (tools/conky-osmo-status.sh : OK/KO/WARN, FS/FU) - l ancien bloc en
--    DejaVu Sans Mono rendait des carres des que le script remettait la police
--    par defaut (${font}) devant une pastille.
--  - Des couleurs PAR NATURE d information, pas une seule teinte : titres et
--    CPU en bleu (color1), ce qui va bien en vert (color2), ce qui manque en
--    rouge (color3), l attention et le debit montant en orange (color4), les
--    libelles en gris (color5), le nom de la machine et le disque en violet
--    (color6), le temps et les valeurs mises en avant en jaune (color7).
--  - Conky n est a l ecran que si un banc tourne : osmo-panel.py le fait
--    apparaitre et disparaitre en fondu (opacite de la fenetre) avec l encart.
conky.config = {
    alignment = 'top_right',
    gap_x = 20, gap_y = 44,
    minimum_width = 400, maximum_width = 400,
    update_interval = 2.0,
    total_run_times = 0,
    background = false,
    double_buffer = true,
    no_buffers = true,
    cpu_avg_samples = 2, net_avg_samples = 2,
    own_window = true,
```

```
    own_window_type = 'desktop',
    own_window_transparent = false,
    own_window_argb_visual = true,
    -- la plaque est peinte par osmo-conky.lua (coins arrondis) : la fenetre
    -- elle-meme reste entierement transparente
    own_window_argb_value = 0,
    own_window_colour = '0b1020',
    own_window_hints = 'undecorated,below,sticky,skip_taskbar,skip_pager',
    border_inner_margin = 18, border_outer_margin = 0,
    draw_borders = false, draw_shades = false, draw_outline = false, draw_graph_borders = false,
    lua_load = '/opt/GSM/osmo-operator/configs/conky/osmo-conky.lua',
    lua_draw_hook_pre = 'panneau',
    use_xft = true, font = 'Ubuntu:size=9', xftalpha = 1,
    override_utf8_locale = true,
    default_color = 'e8eef6',
    color1 = '5ac8fa',   -- bleu    : titres, CPU, valeurs
    color2 = '3fe08a',   -- vert    : OK, RAM, debit descendant
    color3 = 'ff5c6c',   -- rouge   : absent, en echec
    color4 = 'ffb347',   -- orange  : attention, debit montant, charge
    color5 = '9aa7b8',   -- gris    : libelles, filets
    color6 = 'b48cff',   -- violet  : la machine, le disque, l operateur
    color7 = 'ffd166',   -- jaune   : le temps, les valeurs en avant
    default_bar_width = 180, default_bar_height = 5,
    default_graph_width = 364, default_graph_height = 26,
    text_buffer_size = 4096,
};

conky.text = [[
${font Ubuntu:bold:size=16}${color1}OSMO${color5}-${color6}OPERATOR${font}${color}${alignr}${font
 ↪  Ubuntu:size=8}${color5}${execi 60 . /etc/os-release 2>/dev/null; echo
 ↪  "${VARIANT:-$PRETTY_NAME}"}${font}${color}
${voffset -2}${color5}${hr 1}${color}
${voffset 2}${font Ubuntu Mono:size=9}${color5}hôte   ${color6}${nodename}${color}${alignr}${execpi 30
 ↪  /opt/GSM/osmo-operator/tools/conky-osmo-status.sh role}
${color5}noyau  ${color}${kernel}${alignr}${color5}up ${color7}${uptime_short} ${color5}· charge
 ↪  ${color4}${loadavg 1}${color}${font}
${voffset 8}${font Ubuntu:bold:size=10}${color1}Système${font}${color}${alignr}${font
 ↪  Ubuntu:size=8}${color5}${freq_g} GHz · ${color4}${execi 10 t=$(cat /sys/class/thermal/thermal_zone0/temp
 ↪  2>/dev/null); [ -n "$t" ] && echo "$((t/1000))°C"}${font}${color}
${voffset 3}${color5}CPU${color}     ${color1}${cpu cpu0}%${color}${alignr}${color1}${cpubar 5,220 cpu0}${color}
${voffset 2}${cpugraph cpu0 26,364 3fe08a 5ac8fa}
${voffset -2}${color5}RAM${color}     ${color2}${mem}${color5} / ${memmax}${color}${alignr}${color2}${membar
 ↪  5,220}${color}
${voffset 2}${color5}Disque${color}  ${color6}${fs_used /}${color5} / ${fs_size
 ↪  /}${color}${alignr}${color6}${fs_bar 5,220 /}${color}
${voffset 2}${color5}Procs${color}   ${processes} ${color5}· ${color2}${running_processes}${color5}
 ↪  actifs${color}${alignr}${color5}top ${color7}${top name 1} ${top cpu 1}%${color}
${voffset 8}${font Ubuntu:bold:size=10}${color1}Réseau${font}${color}
${voffset 3}${font Ubuntu Mono:size=9}${execpi 10 /opt/GSM/osmo-operator/tools/conky-osmo-status.sh net}${font}
${voffset 6}${color5}${hr 1}${color}
${voffset 3}${execpi 5 /opt/GSM/osmo-operator/tools/conky-osmo-status.sh vue}
${voffset 6}${color5}${hr 1}${color}
${voffset 3}${font Ubuntu:bold:size=10}${color1}Services${font}${color}
${voffset 3}${execpi 10 /opt/GSM/osmo-operator/tools/conky-osmo-status.sh services}
${voffset 6}${color5}${hr 1}${color}
${voffset 3}${alignc}${font Ubuntu:size=8}${color7}sudo -i · ./start-native.sh${color}   ${color5}web
 ↪  ${color1}:8443${color}   ${color5}VTY ${color1}nc 127.0.0.1 4239${color}${font}
]]
```

3.73 /opt/GSM/osmo-operator/configs/conky/osmo-conky.lua

2698 octets, 62 lignes → 62 lignes

```
-- osmo-conky.lua - le PANNEAU du tableau de bord Conky du banc (Cairo).
--
-- [2026-10-12] POURQUOI. Le fond d ecran est maintenant une ville sous un
-- ciel bleu clair (configs/gsm-lab-ville.jpg / gsm-lab-antenne.jpg) : du texte
-- pose a meme le ciel, meme avec une ombre, se lisait mal, et les barres
-- cyan se perdaient dans le bleu. Conky ne sait dessiner qu un rectangle
-- plein (own_window_colour + argb) : d ou ce petit Lua, charge par
-- osmo-conky.conf (lua_load / lua_draw_hook_pre), qui peint AVANT le texte :
--   - une plaque sombre translucide, aux coins arrondis, qui laisse voir le
--     ciel au travers ;
--   - un filet clair d un pixel tout autour ;
--   - un trait d accent (le bleu du banc) le long du bord haut.
-- Conky 1.19 (conky-all, Ubuntu 24.04) : les bindings Cairo sont dans
-- « cairo » et, pour la surface X11, « cairo_xlib » (absent des versions
-- anterieures - on le charge sans l exiger).
require 'cairo'
pcall(require, 'cairo_xlib')

-- Les reglages du panneau (fond, filet, accent) : rgba 0..1.
local FOND     = {0.03, 0.06, 0.12, 0.42}
local FILET    = {1.00, 1.00, 1.00, 0.16}
local ACCENT   = {0.35, 0.78, 0.98, 0.95}   -- #5ac8fa, le bleu du banc
local RAYON    = 16
local MARGE    = 1                          -- le filet ne doit pas raser le bord

local function arrondi(cr, x, y, w, h, r)
    cairo_new_path(cr)
    cairo_arc(cr, x + w - r, y + r, r, -math.pi / 2, 0)
    cairo_arc(cr, x + w - r, y + h - r, r, 0, math.pi / 2)
    cairo_arc(cr, x + r, y + h - r, r, math.pi / 2, math.pi)
    cairo_arc(cr, x + r, y + r, r, math.pi, 3 * math.pi / 2)
    cairo_close_path(cr)
end

function conky_panneau()
    if conky_window == nil then return end
    local w, h = conky_window.width, conky_window.height
    if w < 2 or h < 2 then return end
    local cs = cairo_xlib_surface_create(conky_window.display, conky_window.drawable,
                                         conky_window.visual, w, h)
    local cr = cairo_create(cs)

    -- la plaque
    arrondi(cr, MARGE + 0.5, MARGE + 0.5, w - 2 * MARGE - 1, h - 2 * MARGE - 1, RAYON)
    cairo_set_source_rgba(cr, FOND[1], FOND[2], FOND[3], FOND[4])
    cairo_fill_preserve(cr)
    -- le filet
    cairo_set_source_rgba(cr, FILET[1], FILET[2], FILET[3], FILET[4])
    cairo_set_line_width(cr, 1)
    cairo_stroke(cr)

    -- l accent : un trait sous le bord haut, entre les deux arrondis
    cairo_new_path(cr)
    cairo_move_to(cr, MARGE + RAYON, MARGE + 2)
    cairo_line_to(cr, w - MARGE - RAYON, MARGE + 2)
    cairo_set_source_rgba(cr, ACCENT[1], ACCENT[2], ACCENT[3], ACCENT[4])
    cairo_set_line_width(cr, 2)
    cairo_stroke(cr)

    cairo_destroy(cr)
    cairo_surface_destroy(cs)
end
```

3.74 /opt/GSM/osmo-operator/configs/crypthome/crypthome-fond

3129 octets, 78 lignes → 78 lignes

```
#!/usr/bin/python3
# crypthome-fond - peint une image sur le framebuffer pendant la question LUKS du demarrage.
#
#   crypthome-fond [IMAGE]     (defaut : CRYPTHOME_IMAGE de /etc/default/crypthome)
#
# [2026-10-24] unlock-home quitte plymouth et lit la phrase sur la console texte (le clavier de plymouth
# est mort sur certains portables, voir unlock-home). La console texte, c est fbcon : un framebuffer
# (/dev/fb0) sur lequel le noyau dessine les caracteres. On y peint l image, assombrie dans le bas ;
# l invite est ensuite posee sur la DERNIERE ligne, ou fbcon ne repeint que les cases ecrites. La saisie
# reste celle du tty : l image n est qu un decor, et si elle echoue (pas de /dev/fb0, pas de Pillow,
# format de pixel inconnu) on sort 0 sans rien dire de plus qu une ligne au journal.
import os
import sys


def lire(chemin, defaut=""):
    try:
        with open(chemin) as f:
            return f.read().strip()
    except OSError:
        return defaut


def image_conf():
    for ligne in lire("/etc/default/crypthome").splitlines():
        if ligne.startswith("CRYPTHOME_IMAGE="):
            return ligne.split("=", 1)[1].strip().strip('"')
    return "/usr/share/backgrounds/gsm-lab-antenne.jpg"


def main():
    chemin = sys.argv[1] if len(sys.argv) > 1 else image_conf()
    try:
        from PIL import Image, ImageDraw
    except ImportError:
        return "Pillow absent"
    sysfb = "/sys/class/graphics/fb0"
    try:
        w, h = (int(v) for v in lire(sysfb + "/virtual_size").split(","))
        bpp = int(lire(sysfb + "/bits_per_pixel"))
        stride = int(lire(sysfb + "/stride") or w * bpp // 8)
    except ValueError:
        return "pas de framebuffer"
    if bpp != 32:
        return "framebuffer %d bits par point, seul 32 est gere" % bpp
    try:
        img = Image.open(chemin).convert("RGB")
    except OSError as e:
        return "image illisible (%s)" % e
    # couvrir l ecran sans deformer : mise a l echelle puis recadrage au centre
    k = max(w / img.width, h / img.height)
    img = img.resize((round(img.width * k), round(img.height * k)), Image.LANCZOS)
    x0, y0 = (img.width - w) // 2, (img.height - h) // 2
    img = img.crop((x0, y0, x0 + w, y0 + h))
    # bande sombre en bas : l invite blanche de la console y reste lisible
    bande = Image.new("L", (w, h), 0)
    ImageDraw.Draw(bande).rectangle((0, h - h // 7, w, h), fill=200)
    img = Image.composite(Image.new("RGB", (w, h), (0, 0, 0)), img, bande)
    # XRGB8888 petit-boutiste (B, G, R, X), le format de fbcon sur simpledrm / nvidia-drm / i915
    brut = img.tobytes("raw", "BGRX")
    try:
        with open("/dev/fb0", "r+b") as fb:
```

```
            if stride == w * 4:
                fb.write(brut)
            else:
                for y in range(h):
                    fb.seek(y * stride)
                    fb.write(brut[y * w * 4:(y + 1) * w * 4])
    except OSError as e:
        return "ecriture /dev/fb0 impossible (%s)" % e
    return None


if __name__ == "__main__":
    err = main()
    if err:
        print("crypthome-fond: %s - question sans image." % err, file=sys.stderr)
    sys.exit(0)
```

3.75 /opt/GSM/osmo-operator/configs/crypthome/crypthome-postinstall

11139 octets, 210 lignes → 210 lignes

```
#!/bin/bash
# crypthome-postinstall <compte-de-session>
#
# Lance UNE FOIS, par Calamares (shellprocess@osmo), DANS la cible montee, juste
# apres que l installeur a ecrit /etc/fstab et /etc/crypttab. Il ne chiffre
# rien : le chiffrement est deja fait, par Calamares, sur la partition /home
# (partition.conf : la racine porte noEncrypt, /home non). Ce script ne fait que
# RACCORDER ce que l installeur a produit au dispositif crypthome.
#
# POURQUOI UN RACCORD, ET PAS LE CABLAGE DE CALAMARES TEL QUEL. Calamares ecrit
# une ligne crypttab ordinaire pour /home. Avec une racine EN CLAIR, le module
# fstab y met le mot de passe "none" (il ne cree de /crypto_keyfile.bin que pour
# une racine chiffree) : systemd demande donc la phrase de passe au demarrage et
# ATTEND. C est exactement ce que ce banc ne veut pas - il redemarre dix fois
# par jour, souvent sans personne devant l ecran.
#
# On repasse donc crypttab et fstab en "noauto" et on remplit
# /etc/default/crypthome. C est alors unlock-home.service qui demande la phrase
# de passe, avant le gestionnaire de connexion : la taper ouvre /home et la
# session du compte ; Entree a vide (ou le delai qui expire) demarre en root
# sans /home - et le demarrage se termine dans tous les cas.
#
# LE PROPRIETAIRE, C EST LE COMPTE DE SESSION. Sur la cle live, le home chiffre
# appartenait a osmocom pendant que la session tournait sous root. Sur le disque
# installe, shellprocess-osmo.conf SUPPRIME osmocom et la session s ouvre sous le
# compte choisi a l installation. Laisser CRYPTHOME_OWNER=osmocom y designerait
# un compte qui n existe plus : run-as-owner, lock-home et
# crypthome-session-setup viseraient dans le vide, sans une erreur.
set -u

OWNER="${1:-}"
CONF=/etc/default/crypthome
MOUNT=/home

log() { echo "crypthome-postinstall: $*"; }

[ -r "$CONF" ] || { log "$CONF absent - rien a raccorder."; exit 0; }

# -- CE QUE CALAMARES A MIS SOUS /home ---------------------------------------
# /etc/fstab fait foi : c est lui que le systeme installe lira. Le module fstab
# de Calamares ecrit "/dev/mapper/<nom>" quand la partition est chiffree, et un
# "UUID=..." quand elle ne l est pas. L absence de ligne /home (pas de partition
# separee) se lit de la meme facon : pas de volume chiffre.
```

```
SRC="$(awk '$1 !~ /^#/ && $2 == "'"$MOUNT"'" { print $1; exit }' /etc/fstab 2>/dev/null)"

MAPPER=""
UUID=""
case "$SRC" in
    /dev/mapper/*)
        MAPPER="${SRC#/dev/mapper/}"
        # L UUID du CONTENEUR LUKS, pas celui du systeme de fichiers interne :
        # c est lui que cryptsetup ouvre, et c est ce que porte crypttab.
        UUID="$(awk -v m="$MAPPER" '$1 !~ /^#/ && $1 == m { sub(/^UUID=/, "", $2); print $2; exit }' \
                /etc/crypttab 2>/dev/null)"
        ;;
esac

# ── LE COMPTE PROPRIETAIRE ──────────────────────────────────────────────
# Vide ou inconnu : on ne l ecrit pas, plutot que d ecrire un nom faux. Un
# CRYPTHOME_OWNER vide met simplement run-as-owner en retrait (l application
# demarre sous le compte courant), ce qui est le bon repli.
if [ -n "$OWNER" ] && ! id -u "$OWNER" >/dev/null 2>&1; then
    log "compte « $OWNER » inconnu dans la cible - CRYPTHOME_OWNER laisse tel quel."
    OWNER=""
fi

ecrire_conf() {
    # sed sur les trois cles, sans reecrire le fichier : les commentaires de
    # /etc/default/crypthome sont la documentation du dispositif sur la machine.
    local cle valeur
    for cle in CRYPTHOME_UUID CRYPTHOME_NAME CRYPTHOME_OWNER; do
        case "$cle" in
            CRYPTHOME_UUID)  valeur="$UUID" ;;
            CRYPTHOME_NAME)  valeur="$MAPPER" ;;
            CRYPTHOME_OWNER) valeur="$OWNER" ;;
        esac
        [ -n "$valeur" ] || continue
        if grep -q "^$cle=" "$CONF"; then
            sed -i "s|^$cle=.*|$cle=\"$valeur\"|" "$CONF"
        else
            printf '%s="%s"\n' "$cle" "$valeur" >> "$CONF"
        fi
    done
}

# ── LE TROUSSEAU DE ROOT ────────────────────────────────────────────────
# /root/.local/share/keyrings est un LIEN vers le disque chiffre, pose dans le
# rootfs par l etape 8i. Deux cas, et ils sont opposes :
#
#   /home chiffre  -> on cree la cible du lien sur le volume. Disque non
#                     dechiffre = lien mort = gnome-keyring n ecrit rien en
#                     clair, ce qui est precisement le but.
#   /home en clair -> le lien n a plus rien a proteger, et un lien mort casse
#                     le trousseau de root sans le dire. On remet un vrai
#                     repertoire.
raccorder_trousseau_root() {
    local chiffre="$1" lien cible
    for lien in ${CRYPTHOME_ROOT_LINKS:-.local/share/keyrings}; do
        # "root-" : le nom que posent deja l etape 8i et init-crypthome.sh
        # (/home/.crypthome/root-keyrings). Le prefixe dit a qui appartient le
        # repertoire, sur un volume qui porte aussi les homes des comptes.
        cible="$MOUNT/.crypthome/root-$(basename "$lien")"
        if [ "$chiffre" = 1 ]; then
            install -d -m 0700 "$MOUNT/.crypthome"
            install -d -m 0700 "$cible"
            chown -R root:root "$MOUNT/.crypthome"
            install -d -m 0700 "/root/$(dirname "$lien")"
            # UN VRAI REPERTOIRE A CET ENDROIT N EST PAS UN OBSTACLE A POUSSER.
            # Le lien pose par l etape 8i est mort tant que rien n est monte, et
```

```
            # c est le cas normal ici. Mais si un repertoire REEL s y trouve, il
            # contient un trousseau - « rm -rf » le perdrait sans un mot. On
            # deplace son contenu sur le volume, puis on remplace.
            if [ -d "/root/$lien" ] && [ ! -L "/root/$lien" ]; then
                ( shopt -s dotglob nullglob
                  _restes=("/root/$lien"/*)
                  [ "${#_restes[@]}" -gt 0 ] && mv -n "${_restes[@]}" "$cible"/ ) 2>/dev/null
                rmdir "/root/$lien" 2>/dev/null \
                    || { log "ATTENTION : /root/$lien non vide, laisse EN CLAIR."; continue; }
            else
                rm -f "/root/$lien"
            fi
            ln -sfn "$cible" "/root/$lien"
        elif [ -L "/root/$lien" ]; then
            rm -f "/root/$lien"
            install -d -m 0700 "/root/$lien"
        fi
    done
}

# shellcheck disable=SC1090
. "$CONF" 2>/dev/null || true

if [ -z "$MAPPER" ] || [ -z "$UUID" ]; then
    # ── PAS DE VOLUME CHIFFRE ──────────────────────────────────────
    # L operateur n a pas coche « chiffrer », ou le disque etait trop petit pour
    # que la partition /home de partition.conf tienne. On le DIT : le symptome
    # silencieux serait un banc qu on croit chiffre et qui ne l est pas.
    UUID=""
    MAPPER=""
    ecrire_conf          # n ecrit que CRYPTHOME_OWNER : les deux autres sont vides
    raccorder_trousseau_root 0
    log "ATTENTION : $MOUNT n est PAS chiffre sur ce disque."
    log "  Les profils, cookies et trousseaux y seront lisibles."
    log "  Pour chiffrer un SECOND disque apres coup : /root/init-crypthome.sh"
    exit 0
fi

# ── LE VOLUME DOIT ETRE MONTE LA, MAINTENANT ──────────────────────
# Calamares monte la cible avant d y deverser le squashfs : a cet instant,
# /home DOIT etre le volume dechiffre. S il ne l est pas, ce qu on s apprete a
# ecrire dedans (le trousseau de root) irait sur le repertoire /home de la
# RACINE, en clair, et disparaitrait sous le montage au premier demarrage.
# On le dit et on continue : crypttab, fstab et la configuration, eux, restent
# justes - c est seulement le trousseau qu on ne pose pas.
_monte=1
if command -v findmnt >/dev/null 2>&1; then
    _src_reel="$(findmnt -fno SOURCE "$MOUNT" 2>/dev/null)"
    if [ "$(readlink -f "${_src_reel:-/nonexistent}")" != "$(readlink -f "/dev/mapper/$MAPPER")" ]; then
        _monte=0
        log "ATTENTION : $MOUNT n est pas monte depuis /dev/mapper/$MAPPER"
        log "  (source vue : ${_src_reel:-aucune}) - trousseau de root non pose."
    fi
fi

# ── CRYPTTAB : "noauto" ────────────────────────────────────────────
# systemd ne doit RIEN demander de lui-meme : c est unlock-home qui pose la
# question, avec son delai et son repli. La ligne reste, parce qu elle documente
# le volume et sert a « cryptdisks_start » a la main.
if grep -q "^$MAPPER[[:space:]]" /etc/crypttab 2>/dev/null; then
    sed -i "\|^$MAPPER[[:space:]]|d" /etc/crypttab
fi
printf '%-22s %-40s %-10s %s\n' "$MAPPER" "UUID=$UUID" "none" "luks,noauto" >> /etc/crypttab

# ── FSTAB : "noauto,nofail" ────────────────────────────────────────
# noauto pour la meme raison, nofail en ceinture : une entree /home qui echoue
```

```
# au demarrage ne doit pas basculer la machine en mode secours. On CONSERVE les
# options ecrites par Calamares (noatime, discard sur SSD) au lieu de les
# remplacer par "defaults" - elles ont ete choisies d apres le disque reel.
awk -v m="$MOUNT" '
    $1 ~ /^#/ || $2 != m { print; next }
    {
        opts = $4
        if (opts !~ /(^|,)noauto(,|$)/) opts = opts ",noauto"
        if (opts !~ /(^|,)nofail(,|$)/) opts = opts ",nofail"
        # Le passe fsck a 2 : la racine seule vaut 1, et 0 desactiverait la
        # verification du volume chiffre.
        printf "%-42s %-14s %-7s %s %s %s\n", $1, $2, $3, opts, ($5 == "" ? "0" : $5), "2"
    }
' /etc/fstab > /etc/fstab.crypthome && mv /etc/fstab.crypthome /etc/fstab

ecrire_conf
[ "$_monte" = 1 ] && raccorder_trousseau_root 1

# ── LE SERVICE ──────────────────────────────────────────────────────
# Deja active dans l image (etape 8i) ; on le reactive ici pour le cas d un
# rootfs ou systemctl avait echoue au build. Sans lui, /home reste ferme et
# personne ne saurait pourquoi.
systemctl enable unlock-home.service >/dev/null 2>&1 || true

# Les lanceurs a secrets pointent vers le compte proprietaire, qui vient de
# changer de nom : la liste doit etre refaite avec le bon.
[ -x /usr/local/sbin/refresh-owner-apps ] && \
    /usr/local/sbin/refresh-owner-apps >/dev/null 2>&1 || true

log "$MOUNT chiffre (LUKS $UUID, mapping $MAPPER), proprietaire ${OWNER:-<aucun>}."
log "  Au demarrage : « decrypt $MOUNT -passphrase- or Enter » (sur la console, plymouth quitte avant)"
log "  phrase tapee : session ${OWNER:-du compte}, persistante sur $MOUNT chiffre ;"
log "  Entree a vide, ou pas de reponse : session root ephemere (ecritures dans /tmp),"
log "  sans $MOUNT - le demarrage ne bloque jamais."
exit 0
```

3.76 /opt/GSM/osmo-operator/configs/crypthome/crypthome-session-setup

1880 octets, 42 lignes → 42 lignes

```
#!/bin/bash
# crypthome-session-setup - lance au demarrage de la session (root) de GDM.
#
# Prepare la bascule vers le compte proprietaire AVANT qu'on en ait besoin :
# autorisation sur le serveur X et session utilisateur demarree. Sans cela, le
# premier lancement de Firefox paie l'attente de logind, plusieurs secondes,
# et donne l'impression que rien ne se passe.
#
# Quand /home n'est pas dechiffre, ce script ne fait rien : c'est le mode
# degrade normal, les applications s'ouvriront sous root, sans secrets.
set -u
CONF=/etc/default/crypthome
[ -r "$CONF" ] && . "$CONF"
OWNER="${CRYPTHOME_OWNER:-}"
MOUNT="${CRYPTHOME_MOUNT:-/home}"
MAPPER="${CRYPTHOME_NAME:-crypthome}"

# ── LE MONTAGE DOIT VENIR DU VOLUME CHIFFRE ─────────────────────────
# "monte" ne suffit pas. /home peut etre une partition ORDINAIRE - une machine
# installee avec un /home separe, ou init-crypthome.sh n a jamais tourne. Traiter
# ce montage-la comme le disque chiffre ecrirait les secrets EN CLAIR sous le
# compte proprietaire tout en annoncant le contraire : le pire des etats, parce
# qu il a l air correct. On exige donc que la source du montage soit le mapping
# dechiffre lui-meme, et pas seulement qu il y ait quelque chose sur /home.
```

```
crypthome_monte() {
    local _src
    mountpoint -q "$MOUNT" || return 1
    _src="$(findmnt -fno SOURCE "$MOUNT" 2>/dev/null)"
    [ -n "$_src" ] || return 1
    [ "$(readlink -f "$_src")" = "$(readlink -f "/dev/mapper/$MAPPER")" ]
}


[ -n "$OWNER" ] || exit 0
id -u "$OWNER" >/dev/null 2>&1 || exit 0
crypthome_monte || exit 0
[ -d "$(getent passwd "$OWNER" | cut -d: -f6)" ] || exit 0

[ -n "${DISPLAY:-}" ] && command -v xhost >/dev/null 2>&1 && \
    xhost "+SI:localuser:$OWNER" >/dev/null 2>&1 || true
loginctl enable-linger "$OWNER" >/dev/null 2>&1 || true
exit 0
```

3.77 /opt/GSM/osmo-operator/configs/crypthome/crypthome-session.desktop

302 octets, 8 lignes → 8 lignes

```
[Desktop Entry]
Type=Application
Name=Preparation du compte chiffre
Comment=Autorise le compte proprietaire sur l'affichage et demarre sa session utilisateur, quand /home est
↪  dechiffre
Exec=/usr/local/bin/crypthome-session-setup
NoDisplay=true
X-GNOME-Autostart-enabled=true
X-Crypthome-Generated=true
```

3.78 /opt/GSM/osmo-operator/configs/crypthome/crypthome.default

4622 octets, 84 lignes → 84 lignes

```
# /etc/default/crypthome - second disque chiffre porte /home
#
# LE MODELE. Le disque systeme n'est PAS chiffre : tout ce qui est systeme est
# lisible - l'OS, /etc, et /root qui est le compte de TRAVAIL du banc. Ce qui
# est chiffre, c'est le second disque, monte sur /home : les profils, les
# donnees personnelles, et TOUT CE QUI TIENT AU KEYRING.
#
# QUI REMPLIT CE FICHIER. Deux ecrivains, pour deux chemins :
#   - crypthome-postinstall, a l INSTALLATION, quand Calamares vient de chiffrer
#     la partition /home (le cas normal) ;
#   - init-crypthome.sh, apres coup, quand on chiffre un SECOND disque ajoute a
#     une machine deja installee.
# CRYPTHOME_UUID est vide tant qu aucun des deux n'est passe ; unlock-home ne
# fait alors rien et le boot n'en souffre pas.

# UUID du conteneur LUKS (= UUID vu par blkid sur la partition chiffree).
CRYPTHOME_UUID=""

# Nom du mapping dans /dev/mapper et point de montage.
#
# "crypthome" est le nom que choisit init-crypthome.sh. Une installation par
# Calamares, elle, herite du nom que kpmcore a forge - "luks-<uuid>" - et
# crypthome-postinstall ecrit CELUI-LA ici. Les deux conviennent : tout le
# dispositif lit ce reglage au lieu de supposer un nom.
CRYPTHOME_NAME="crypthome"
```

```
CRYPTHOME_MOUNT="/home"

# Le compte dont le home vit sur le disque chiffre : c'est LUI qui porte le
# keyring et fait tourner les applications a secrets (voir run-as-owner).
#
# Sur la cle live, c est osmocom, et la session tourne sous root. Sur un disque
# installe, osmocom n existe plus (shellprocess-osmo.conf le supprime) et
# crypthome-postinstall met ici le compte choisi a l installation - qui est
# aussi celui de la session. run-as-owner n a alors plus rien a basculer et
# lance l application telle quelle : c est le bon comportement, le home de ce
# compte EST deja sur le volume chiffre.
#
# DEUX SESSIONS, SELON LA REPONSE AU DEMARRAGE (unlock-home) :
#   oui -> le compte ci-dessus, PERSISTANT : son home est sur le volume chiffre ;
#   non -> ROOT, EPHEMERE : GDM s ouvre sur root pour ce demarrage, et tout ce
#          que la session ecrit dans /root - et dans /opt/GSM, voir
#          CRYPTHOME_RAM_DIRS - reste dans /tmp/crypthome-root (tmpfs, en RAM) :
#          rien ne survit a l arret.
CRYPTHOME_OWNER="__OWNER__"

# Boot : nombre d'essais de phrase de passe, et delai (s) au bout duquel une
# question sans reponse vaut "non". Le boot n'est JAMAIS bloque : au pire on
# demarre sans /home.
CRYPTHOME_TRIES=3
CRYPTHOME_TIMEOUT=60

# Ou la question « decrypt /home -passphrase- or Enter » est posee au boot :
#   console  (defaut) sur la console texte, lue par unlock-home lui-meme ;
#            plymouth, s'il tourne, est quitte juste avant. C'est le chemin
#            qui marche partout : il ne depend ni d'un agent systemd ni du
#            clavier de plymouth (muet sur certaines machines, portable
#            Intel + NVIDIA en tete).
#   plymouth dans le theme de demarrage (osmo-bts), via systemd-ask-password :
#            plus joli, a reserver a une machine ou l'on a VERIFIE que la
#            frappe y est prise.
CRYPTHOME_PROMPT="console"

# [2026-10-24] Image peinte derriere la question (console, /dev/fb0) par
# crypthome-fond. "none" : console texte nue.
CRYPTHOME_IMAGE="/usr/share/backgrounds/gsm-lab-antenne.jpg"

# Repertoires recouverts d un overlay en RAM pendant une session ROOT (/home
# ferme), EN PLUS de /root qui l est toujours. Le banc entier vit dans /opt/GSM
# (depot, configs, hlr.db, journaux) : en session root il est "live" - tout ce
# qui s y ecrit s evapore a l arret, et le disque garde l etat d avant. Le
# contenu reste lisible et executable ; seules les ecritures partent en RAM.
# Separes par des espaces ; un repertoire absent est ignore. Vide = /root seul.
CRYPTHOME_RAM_DIRS="/opt/GSM"

# Repertoires de root qui ne doivent pas exister en clair sur le disque
# systeme : init-crypthome.sh les remplace par des liens vers le disque
# chiffre. Disque non dechiffre = lien mort = rien ne s'ecrit en clair.
CRYPTHOME_ROOT_LINKS=".local/share/keyrings"

# Les applications qui portent des secrets : leurs lanceurs sont rediriges
# vers run-as-owner par refresh-owner-apps. Noms de fichiers .desktop, sans
# l'extension ; ceux qui ne sont pas installes sont ignores en silence.
# Relancer « refresh-owner-apps » apres en avoir ajoute.
CRYPTHOME_OWNER_APPS="firefox firefox-esr thunderbird chromium chromium-browser google-chrome keepassxc seahorse
 ↪  org.gnome.seahorse.Application signal-desktop element-desktop telegramdesktop org.gnome.Geary
 ↪  org.gnome.Evolution evolution nextcloud org.keepassxc.KeePassXC"
```

3.79 /opt/GSM/osmo-operator/configs/crypthome/init-crypthome.sh

14186 octets, 283 lignes → 283 lignes

```
#!/bin/bash
# init-crypthome.sh - initialise le second disque en LUKS2 et y deplace /home.
#
#   sudo /root/init-crypthome.sh [/dev/partition]
#
# CE N EST PLUS LE CHEMIN NORMAL. [2026-09-25] Sur une machine installee depuis
# l ISO, /home est chiffre PAR L INSTALLEUR : partition.conf decoupe le disque
# en racine + /home et ne chiffre que /home, crypthome-postinstall raccorde le
# tout. Rien a lancer apres coup, et pas de basculement etale sur deux
# demarrages.
#
# Ce script sert au cas que Calamares ne couvre pas : chiffrer un SECOND DISQUE
# ajoute a une machine deja installee. Il refuse d ailleurs toute partition du
# disque de la racine.
#
# Sans argument, la partition candidate est deduite - et il doit n'y en avoir
# qu'une, sinon le script s'arrete et les affiche.
#
# CE SCRIPT EFFACE LA PARTITION VISEE. Il refuse de tourner si elle contient
# autre chose que lost+found, si elle est montee, ou si elle porte deja un
# en-tete LUKS - et il demande une confirmation tapee en toutes lettres.
#
# Ce qu'il fait, dans l'ordre : chiffre la partition, y recopie le contenu
# actuel de /home, met l'ancien /home de cote, cable /etc/crypttab et
# /etc/fstab en "noauto" (c'est unlock-home qui monte, apres avoir demande),
# et active unlock-home.service.
set -euo pipefail

RED='\033[0;31m'; GREEN='\033[0;32m'; CYAN='\033[0;36m'
YELLOW='\033[1;33m'; NC='\033[0m'; BOLD='\033[1m'

CONF=/etc/default/crypthome
. "$CONF"
MAPPER="${CRYPTHOME_NAME:-crypthome}"
MOUNT="${CRYPTHOME_MOUNT:-/home}"
OWNER="${CRYPTHOME_OWNER:-}"
TMP=/mnt/crypthome-init

[ "$(id -u)" -eq 0 ] || { echo -e "${RED}A lancer en root.${NC}" >&2; exit 1; }

# ── OPTIONS ──────────────────────────────────────────────────────────────
#   --key-file=F  phrase de passe lue dans F au lieu d etre demandee. Pour les
#                 installations scriptees. Le fichier ne doit PAS survivre au
#                 script : c est a l appelant de l effacer.
#   --yes         pas de confirmation. A reserver a un appel non interactif ;
#                 le script EFFACE la partition qu il vise.
KEYFILE=""
SANS_QUESTION=0
_args=()
for _a in "$@"; do
    case "$_a" in
        --key-file=*) KEYFILE="${_a#--key-file=}" ;;
        --yes|-y)     SANS_QUESTION=1 ;;
        -*)           echo -e "${RED}Option inconnue : $_a${NC}" >&2; exit 2 ;;
        *)            _args+=("$_a") ;;
    esac
done
set -- ${_args[@]+"${_args[@]}"}
if [ -n "$KEYFILE" ] && [ ! -r "$KEYFILE" ]; then
    echo -e "${RED}--key-file : $KEYFILE illisible.${NC}" >&2; exit 1
fi
```

```
# ── QUELLE PARTITION ──────────────────────────────────────────────────
# Rien n'est devine en silence : ce script EFFACE ce qu'il vise. Sans argument
# il ne retient que les partitions qui remplissent TOUTES les conditions - pas
# sur le disque de la racine, non montee, sans en-tete LUKS, ni iso9660 ni swap
# ni EFI - et il faut qu'il n'en reste qu'une pour qu'il aille plus loin seul.
disque_racine="$(lsblk -no PKNAME "$(findmnt -no SOURCE /)" 2>/dev/null | head -1)"
candidates() {
    local p pk
    for p in $(lsblk -lnpo NAME,TYPE | awk '$2=="part"{print $1}'); do
        pk="$(lsblk -no PKNAME "$p" 2>/dev/null | head -1)"
        [ "$pk" = "$disque_racine" ] && continue
        findmnt -S "$p" >/dev/null 2>&1 && continue
        cryptsetup isLuks "$p" 2>/dev/null && continue
        case "$(blkid -o value -s TYPE "$p" 2>/dev/null)" in
            iso9660|swap|vfat) continue ;;
        esac
        echo "$p"
    done
}

if [ "$#" -ge 1 ]; then
    DEV="$1"
else
    mapfile -t _cand < <(candidates)
    if [ "${#_cand[@]}" -eq 1 ]; then
        DEV="${_cand[0]}"
        echo -e "${YELLOW}Partition candidate unique retenue : ${CYAN}$DEV${NC}"
    elif [ "${#_cand[@]}" -eq 0 ]; then
        echo -e "${RED}Aucune partition candidate.${NC}" >&2
        echo    "Il en faut une hors du disque de la racine (${disque_racine:-?})," >&2
        echo    "non montee et sans en-tete LUKS. Creez-la, puis relancez :" >&2
        echo    "  $0 /dev/<partition>" >&2
        exit 1
    else
        echo -e "${RED}Plusieurs partitions candidates - precisez laquelle :${NC}" >&2
        lsblk -po NAME,SIZE,FSTYPE,LABEL "${_cand[@]}" >&2
        echo    "  $0 /dev/<partition>" >&2
        exit 1
    fi
fi

# ── VERIFICATIONS ─────────────────────────────────────────────────────
[ -b "$DEV" ] || { echo -e "${RED}$DEV n'est pas une partition.${NC}" >&2; exit 1; }

if findmnt -S "$DEV" >/dev/null 2>&1; then
    echo -e "${RED}$DEV est monte. Demontez-le d'abord.${NC}" >&2; exit 1
fi

if cryptsetup isLuks "$DEV" 2>/dev/null; then
    echo -e "${RED}$DEV porte deja un en-tete LUKS.${NC}" >&2
    echo    "Si c'est un essai precedent a refaire, effacez-le sciemment :" >&2
    echo    "  cryptsetup erase $DEV && wipefs -a $DEV" >&2
    exit 1
fi

# Le disque doit etre vide. On regarde VRAIMENT, en montant en lecture seule :
# un blkid qui dit "ext4" ne dit pas s'il y a 900 Go de donnees dedans.
if blkid "$DEV" >/dev/null 2>&1; then
    mkdir -p "$TMP"
    if mount -o ro "$DEV" "$TMP" 2>/dev/null; then
        reste="$(find "$TMP" -mindepth 1 -maxdepth 1 ! -name lost+found -print -quit)"
        df -h "$TMP" | tail -1
        umount "$TMP"
        if [ -n "$reste" ]; then
            echo -e "${RED}$DEV contient des donnees (par ex. $(basename "$reste")).${NC}" >&2
            echo    "Ce script ne formate que des partitions vides. Sauvegardez et videz-la." >&2
```

```
            exit 1
        fi
    fi
    rmdir "$TMP" 2>/dev/null || true
fi

taille="$(lsblk -dno SIZE "$DEV" | tr -d ' ')"
echo
echo -e "${BOLD}Initialisation du disque chiffre${NC}"
echo -e "  partition      : ${CYAN}$DEV${NC} ($taille)"
echo -e "  deviendra      : ${CYAN}$MOUNT${NC} (LUKS2 + ext4, mappe sur /dev/mapper/$MAPPER)"
echo -e "  compte a home  : ${CYAN}$OWNER${NC}"
echo -e "  au demarrage   : ${CYAN}\"decrypt $MOUNT -passphrase- or Enter\"${NC}, Entree a vide n'empeche pas de
↪  demarrer"
echo -e "  ${YELLOW}le contenu actuel de $MOUNT sera recopie dessus ; l original reste en place${NC}"
echo -e "  ${YELLOW}et part dans /var/backups/ au redemarrage, avant le montage${NC}"
echo
echo -e "${RED}${BOLD}TOUT CE QUE CONTIENT $DEV SERA DETRUIT.${NC}"
if [ "$SANS_QUESTION" = 1 ]; then
    echo -e "${YELLOW}--yes : confirmation passee.${NC}"
else
    read -r -p "Tapez OUI en majuscules pour continuer : " accord
    [ "$accord" = "OUI" ] || { echo "Abandon."; exit 1; }
fi

# ── LA SESSION DU COMPTE PROPRIETAIRE DOIT ETRE FERMEE ──────────────────
# Recopier un home pendant que sa session tourne donne une copie incoherente :
# le trousseau, dconf et les profils de navigateur sont reecrits en continu.
# ── LA SESSION DU COMPTE PROPRIETAIRE ───────────────────────────────────
# Recopier un home pendant que sa session tourne donne une copie un peu
# flottante : dconf, l historique du navigateur et les fichiers de cache sont
# reecrits en continu. Ce n est pas grave ICI - l original n est pas efface, il
# est mis de cote intact au demarrage suivant, et c est le trousseau (statique)
# qui compte.
#
# ON NE TUE PAS LA SESSION. Ce script est souvent lance depuis un terminal qui
# EN FAIT PARTIE : la fermer, c est se couper le tapis sous les pieds, entre
# luksFormat et la recopie, et laisser un disque chiffre a moitie rempli. Le
# vrai basculement se fait au demarrage suivant, quand personne ne tient /home
# - c est unlock-home qui evacue alors l ancien contenu.
if [ -n "$OWNER" ] && loginctl list-sessions --no-legend 2>/dev/null | grep -qw "$OWNER"; then
    echo -e "${YELLOW}Session de $OWNER ouverte : la copie sera prise a chaud.${NC}"
    echo -e "${YELLOW}L original n est pas efface - il sera mis de cote au redemarrage.${NC}"
fi

# ── 1. CHIFFREMENT ──────────────────────────────────────────────────────
echo
echo -e "${GREEN}[1/7]${NC} Creation du conteneur LUKS2 sur $DEV."
echo -e "      ${BOLD}Choisissez une phrase de passe longue.${NC} Elle ne se recupere pas :"
echo -e "      perdue, le contenu du disque l'est aussi."
if [ -n "$KEYFILE" ]; then
    cryptsetup luksFormat --type luks2 --label crypthome --batch-mode \
              --key-file="$KEYFILE" "$DEV"
else
    cryptsetup luksFormat --type luks2 --label crypthome "$DEV"
fi

echo
if [ -n "$KEYFILE" ]; then
    echo -e "${GREEN}[2/7]${NC} Ouverture du volume."
else
    echo -e "${GREEN}[2/7]${NC} Ouverture du volume (retapez la phrase de passe)."
fi
if [ -n "$KEYFILE" ]; then
    cryptsetup open --key-file="$KEYFILE" "$DEV" "$MAPPER"
else
```

```
    cryptsetup open "$DEV" "$MAPPER"
fi

UUID="$(cryptsetup luksUUID "$DEV")"
echo -e "      UUID LUKS : ${CYAN}$UUID${NC}"

# ── 3. SYSTEME DE FICHIERS ──────────────────────────────────────────
echo -e "${GREEN}[3/7]${NC} Formatage ext4."
mkfs.ext4 -q -L crypthome "/dev/mapper/$MAPPER"

# ── 4. RECOPIE DE /home ─────────────────────────────────────────────
echo -e "${GREEN}[4/7]${NC} Recopie du contenu actuel de $MOUNT."
mkdir -p "$TMP"
mount "/dev/mapper/$MAPPER" "$TMP"
# -aHAX : liens durs, ACL et attributs etendus compris - un home sans ses ACL
# ni ses xattrs revient a des permissions subtilement fausses, qu'on ne
# decouvre qu'a l'usage.
#
# --sparse : SANS LUI, les fichiers creux sont recopies TROUS COMPRIS, zero par
# zero. Mesure le 2026-09-15 sur un home de 6,6 Go : 19 Go a l'arrivee, les
# 12 Go d'ecart venant d'une seule image QEMU de pmbootstrap (16 Go annonces,
# 3,9 Go reellement alloues). Les machines virtuelles et les images disque sont
# la regle dans un home d'operateur, pas l'exception.
#
# -x : on ne franchit pas les points de montage. Ce qui est monte SOUS le home
# appartient a un autre disque ; le recopier ici le dupliquerait, et pas au bon
# endroit.
#
# --info=progress2 seulement sur un terminal : lance par systemd, sa barre de
# progression ecrit des dizaines de milliers de lignes dans le journal.
_rs=(-aHAX --sparse -x)
[ -t 1 ] && _rs+=(--info=progress2)
rsync "${_rs[@]}" "$MOUNT"/ "$TMP"/
# Le trousseau de root vit ici, et nulle part ailleurs : /root/.local/share/
# keyrings est un lien vers ce repertoire.
install -d -m 0700 "$TMP/.crypthome/root-keyrings"
chown -R root:root "$TMP/.crypthome"
sync
umount "$TMP"
rmdir "$TMP"

# ── 5. CABLAGE ──────────────────────────────────────────────────────
echo -e "${GREEN}[5/7]${NC} Ecriture de la configuration."
sed -i "s|^CRYPTHOME_UUID=.*|CRYPTHOME_UUID=\"$UUID\"|" "$CONF"

# crypttab : "noauto" - systemd ne doit RIEN demander de lui-meme au boot,
# c'est unlock-home qui pose la question et ouvre le volume.
sed -i "/^$MAPPER[[:space:]]/d" /etc/crypttab
printf '%-22s %-40s %-10s %s\n' "$MAPPER" "UUID=$UUID" "none" "luks,noauto" >> /etc/crypttab

# fstab : "noauto" pour la meme raison, et "nofail" en ceinture - une entree
# qui echoue au boot sur un systeme sans le disque ne doit pas basculer la
# machine en mode secours.
sed -i "\|^[^#].*[[:space:]]$MOUNT[[:space:]]|d" /etc/fstab
printf '%-42s %-14s %-7s %s %s %s\n' \
    "/dev/mapper/$MAPPER" "$MOUNT" "ext4" "defaults,noauto,nofail" "0" "2" >> /etc/fstab

# ── 6. L'ANCIEN /home ───────────────────────────────────────────────
# On n y touche pas, et ce n est pas un oubli : des processus le tiennent en ce
# moment meme (la session depuis laquelle ce script tourne, le plus souvent).
# C est unlock-home qui l evacuera vers /var/backups au demarrage suivant,
# AVANT de monter le disque chiffre - le seul instant ou personne ne l occupe.
echo -e "${GREEN}[6/7]${NC} Ancien $MOUNT laisse en place ; il sera mis de cote au redemarrage."

# ── 7. ACTIVATION ───────────────────────────────────────────────────
echo -e "${GREEN}[7/7]${NC} Activation du service de demarrage."
```

```
systemctl daemon-reload
systemctl enable unlock-home.service >/dev/null
/usr/local/sbin/refresh-owner-apps >/dev/null
# [2026-10-01] On ne retire plus "splash" de la ligne de commande du noyau :
# unlock-home quitte plymouth lui-meme juste avant de poser la question sur la
# console. Le decor de demarrage reste, la question est lisible, et une
# installation Calamares et un disque ajoute apres coup se comportent pareil.

cryptsetup close "$MAPPER"

echo
echo -e "${GREEN}${BOLD}Termine.${NC}"
echo -e "  Au prochain demarrage : ${CYAN}\"decrypt $MOUNT -passphrase- or Enter\"${NC}"
echo -e "  Taper la phrase (affichee en ${CYAN}****${NC}) monte $MOUNT et ouvre la session de $OWNER,"
echo -e "  avec son trousseau ; Firefox et les applications a secrets tourneront sous lui."
echo -e "  ${CYAN}Entree${NC} a vide (ou attendre ${CRYPTHOME_TIMEOUT:-60} s) demarre sans : la session root"
echo -e "  s'ouvre normalement, les memes applications tournent sans secrets."
echo
echo -e "  Apres coup, dans un terminal : ${CYAN}unlock-home${NC} / ${CYAN}lock-home${NC}"
echo -e "  Au redemarrage, l ancien $MOUNT part dans ${CYAN}/var/backups/home-avant-chiffrement-*${NC}"
echo -e "  - a effacer une fois le dechiffrement verifie."
echo
echo -e "  ${YELLOW}Verifiez le dechiffrement AVANT d'effacer la sauvegarde.${NC}"
exit 0
```

3.80 /opt/GSM/osmo-operator/configs/crypthome/lock-home

2308 octets, 56 lignes → 56 lignes

```
#!/bin/bash
# lock-home - demonte /home et referme le volume chiffre, en cours de session.
#
# A utiliser quand on veut rendre les secrets illisibles sans redemarrer. Les
# processus du compte proprietaire tiennent le point de montage : on les
# termine d'abord, sinon le demontage echoue avec "target is busy" et le
# volume reste ouvert - exactement ce qu'on cherchait a eviter.
set -u

CONF=/etc/default/crypthome
[ -r "$CONF" ] || { echo "lock-home: $CONF introuvable" >&2; exit 1; }
. "$CONF"

MAPPER="${CRYPTHOME_NAME:-crypthome}"
MOUNT="${CRYPTHOME_MOUNT:-/home}"
OWNER="${CRYPTHOME_OWNER:-}"

# ── LE MONTAGE DOIT VENIR DU VOLUME CHIFFRE ──────────────────────────
# "monte" ne suffit pas. /home peut etre une partition ORDINAIRE - une machine
# installee avec un /home separe, ou init-crypthome.sh n a jamais tourne. Traiter
# ce montage-la comme le disque chiffre ecrirait les secrets EN CLAIR sous le
# compte proprietaire tout en annoncant le contraire : le pire des etats, parce
# qu il a l air correct. On exige donc que la source du montage soit le mapping
# dechiffre lui-meme, et pas seulement qu il y ait quelque chose sur /home.
crypthome_monte() {
    local _src
    mountpoint -q "$MOUNT" || return 1
    _src="$(findmnt -fno SOURCE "$MOUNT" 2>/dev/null)"
    [ -n "$_src" ] || return 1
    [ "$(readlink -f "$_src")" = "$(readlink -f "/dev/mapper/$MAPPER")" ]
}

# Rien a fermer, ET rien a demonter qui nous appartienne : un /home ordinaire
# n est pas de notre ressort - le demonter tuerait des sessions pour rien.
if ! crypthome_monte && [ ! -e "/dev/mapper/$MAPPER" ]; then
```

```
    echo "lock-home: $MOUNT n'est pas dechiffre."
    exit 0
fi

if [ -n "$OWNER" ] && id -u "$OWNER" >/dev/null 2>&1; then
    loginctl disable-linger "$OWNER" >/dev/null 2>&1 || true
    loginctl terminate-user "$OWNER"  >/dev/null 2>&1 || true
    pkill -u "$OWNER" >/dev/null 2>&1 || true
    sleep 2
    pkill -KILL -u "$OWNER" >/dev/null 2>&1 || true
fi

if crypthome_monte && ! umount "$MOUNT"; then
    echo "lock-home: $MOUNT occupe - processus restants :" >&2
    fuser -vm "$MOUNT" 2>&1 | head -20 >&2
    exit 1
fi

cryptsetup close "$MAPPER" 2>/dev/null || true
echo "lock-home: $MOUNT demonte, volume referme."
exit 0
```

3.81 /opt/GSM/osmo-operator/configs/crypthome/refresh-owner-apps

2606 octets, 68 lignes → 68 lignes

```
#!/bin/bash
# refresh-owner-apps - redirige les lanceurs des applications a secrets vers
# run-as-owner.
#
# Pour chaque .desktop de CRYPTHOME_OWNER_APPS trouve dans /usr/share/
# applications, ecrit une COPIE dans /usr/local/share/applications dont chaque
# ligne Exec= (celle du lanceur ET celles des [Desktop Action ...]) est
# prefixee de run-as-owner. /usr/local/share precede /usr/share dans
# XDG_DATA_DIRS : la copie masque l'original sans le modifier, et un
# "apt upgrade" ne defait rien.
#
# A relancer apres avoir installe une application ou allonge la liste.
set -u

CONF=/etc/default/crypthome
[ -r "$CONF" ] && . "$CONF"
APPS="${CRYPTHOME_OWNER_APPS:-}"
DEST=/usr/local/share/applications
SRCDIRS="/usr/share/applications /var/lib/flatpak/exports/share/applications"

install -d "$DEST"

# Menage : les copies posees par ce script et dont la source a disparu, ou qui
# ne sont plus dans la liste. Le marqueur X-Crypthome-Generated distingue nos
# fichiers de ceux que l'operateur aurait mis la lui-meme - on ne touche
# jamais a ceux-la.
for f in "$DEST"/*.desktop; do
    [ -e "$f" ] || continue
    grep -q '^X-Crypthome-Generated=true' "$f" 2>/dev/null || continue
    nom="$(basename "$f" .desktop)"
    case " $APPS " in
        *" $nom "*) ;;
        *) rm -f "$f"; echo "  - $nom (retire de la liste)" ;;
    esac
done

poses=0
for app in $APPS; do
```

```
    src=""
    for d in $SRCDIRS; do
        [ -f "$d/$app.desktop" ] && { src="$d/$app.desktop"; break; }
    done
    [ -n "$src" ] || continue

    dst="$DEST/$app.desktop"
    # sed sur Exec= uniquement, et idempotent : une ligne deja prefixee n'est
    # pas prefixee deux fois (relancer le script ne doit rien empiler).
    sed -E 's#^Exec=(/usr/local/bin/run-as-owner )?#Exec=/usr/local/bin/run-as-owner #' "$src" > "$dst.tmp"
    # Le marqueur en fin de groupe [Desktop Entry] : une cle inconnue prefixee
    # X- est ignoree par les menus, c'est la convention de la specification.
    if ! grep -q '^X-Crypthome-Generated=true' "$dst.tmp"; then
        awk '
            /^\[/ && n++ == 1 { print "X-Crypthome-Generated=true"; print "" }
            { print }
            END { if (n < 2) print "X-Crypthome-Generated=true" }
        ' "$dst.tmp" > "$dst.tmp2" && mv "$dst.tmp2" "$dst.tmp"
    fi
    mv "$dst.tmp" "$dst"
    chmod 0644 "$dst"
    poses=$((poses + 1))
    echo "  + $app -> run-as-owner"
done

command -v update-desktop-database >/dev/null 2>&1 && \
    update-desktop-database "$DEST" >/dev/null 2>&1 || true

echo "refresh-owner-apps: $poses lanceur(s) redirige(s) vers ${CRYPTHOME_OWNER:-<aucun>}."
exit 0
```

3.82 /opt/GSM/osmo-operator/configs/crypthome/run-as-owner

6618 octets, 129 lignes → 129 lignes

```
#!/bin/bash
# run-as-owner <commande> [arguments...]
#
# Lance une application sous le compte dont le home vit sur le disque chiffre
# (CRYPTHOME_OWNER) QUAND ce disque est dechiffre ; sinon la lance telle quelle,
# sous le compte courant.
#
# POURQUOI. La session s'ouvre sur root : c'est le compte de travail du banc,
# et tout y est lisible. Mais un navigateur, un trousseau, un client de
# messagerie ecrivent des SECRETS - cookies, jetons, mots de passe. Ceux-la
# n'ont rien a faire sur le disque systeme en clair : ils appartiennent au
# compte proprietaire, dont tout le home est chiffre.
#
# Disque non dechiffre = pas de home chiffre = l'application demarre quand meme,
# en root, sur un profil neuf du disque systeme. On prefere une application qui
# s'ouvre sans secrets a une application qui refuse de s'ouvrir.
set -u

CONF=/etc/default/crypthome
[ -r "$CONF" ] && . "$CONF"
OWNER="${CRYPTHOME_OWNER:-}"
MOUNT="${CRYPTHOME_MOUNT:-/home}"
MAPPER="${CRYPTHOME_NAME:-crypthome}"

# ── LE MONTAGE DOIT VENIR DU VOLUME CHIFFRE ──────────────────────────────
# "monte" ne suffit pas. /home peut etre une partition ORDINAIRE - une machine
# installee avec un /home separe, ou init-crypthome.sh n a jamais tourne. Traiter
# ce montage-la comme le disque chiffre ecrirait les secrets EN CLAIR sous le
# compte proprietaire tout en annoncant le contraire : le pire des etats, parce
```

```
# qu il a l air correct. On exige donc que la source du montage soit le mapping
# dechiffre lui-meme, et pas seulement qu il y ait quelque chose sur /home.
crypthome_monte() {
    local _src
    mountpoint -q "$MOUNT" || return 1
    _src="$(findmnt -fno SOURCE "$MOUNT" 2>/dev/null)"
    [ -n "$_src" ] || return 1
    [ "$(readlink -f "$_src")" = "$(readlink -f "/dev/mapper/$MAPPER")" ]
}


[ "$#" -ge 1 ] || { echo "usage: run-as-owner <commande> [args...]" >&2; exit 2; }

# Deja sous un compte ordinaire : rien a basculer.
if [ "$(id -u)" -ne 0 ] || [ -z "$OWNER" ] || ! id -u "$OWNER" >/dev/null 2>&1; then
    exec "$@"
fi

OWNER_UID="$(id -u "$OWNER")"
OWNER_HOME="$(getent passwd "$OWNER" | cut -d: -f6)"
OWNER_SHELL="$(getent passwd "$OWNER" | cut -d: -f7)"
[ -n "$OWNER_SHELL" ] || OWNER_SHELL=/bin/bash

# Le volume chiffre doit etre monte ET le home exister dessus. Les deux : un
# /home monte mais vide (disque fraichement initialise) ne doit pas faire
# demarrer une application sur un home inexistant.
if ! crypthome_monte || [ ! -d "$OWNER_HOME" ]; then
    logger -t run-as-owner -p user.notice \
        "$1 lance sous root : $MOUNT n'est pas le volume chiffre dechiffre, ou pas de profil $OWNER."
    exec "$@"
fi

# ── L'ACCES A L'AFFICHAGE ──────────────────────────────────────────
# Le serveur X appartient a root. SI:localuser autorise le compte par ses
# credentials locaux, sans partager le cookie de root : on N'EXPORTE PAS
# XAUTHORITY, qui pointe vers un fichier que le compte ne peut pas lire et
# ferait echouer les bibliotheques qui le lisent avant de se rabattre.
if [ -n "${DISPLAY:-}" ] && command -v xhost >/dev/null 2>&1; then
    xhost "+SI:localuser:$OWNER" >/dev/null 2>&1 || true
fi

# ── LA SESSION UTILISATEUR DU COMPTE ───────────────────────────────
# lingering : logind cree /run/user/<uid> et demarre user@<uid>.service sans
# qu'une session soit ouverte - or personne ne se connecte jamais sous ce
# compte, les applications sont lancees depuis la session de root.
if [ ! -d "/run/user/$OWNER_UID" ]; then
    loginctl enable-linger "$OWNER" >/dev/null 2>&1 || true
    for _ in 1 2 3 4 5 6 7 8 9 10; do
        [ -d "/run/user/$OWNER_UID" ] && break
        sleep 0.3
    done
fi
if [ ! -d "/run/user/$OWNER_UID" ]; then
    install -d -m 0700 -o "$OWNER" -g "$(id -gn "$OWNER")" "/run/user/$OWNER_UID"
fi

RUNTIME="/run/user/$OWNER_UID"
BUS="unix:path=$RUNTIME/bus"
[ -S "$RUNTIME/bus" ] || BUS=""

# ── LE SON ─────────────────────────────────────────────────────────
# PulseAudio tourne en mode SYSTEME : un seul socket, /run/pulse/native, et un
# seul proprietaire - celui de la derniere session humaine ouverte, donc root
# ici. Or c'est le navigateur qui porte le micro du tableau de bord, et il
# tourne desormais sous le compte proprietaire : sans ce rappel, il obtient
# « NotFoundError » sur getUserMedia pendant que pactl liste les peripheriques
```

```
# en RUNNING - l'ecart qu'on cherche des heures du cote « permission micro ».
# osmo-pulse-link.sh pose le lien dans /run/user/<uid> ET donne le socket a cet
# uid. La session root n'a pas de navigateur : elle n'en perd rien.
if [ -x /usr/local/sbin/osmo-pulse-link.sh ]; then
    /usr/local/sbin/osmo-pulse-link.sh "$OWNER_UID" >/dev/null 2>&1 || true
fi
# Et l'adresse en clair, plutot que de s'en remettre au seul lien : c'est ce
# que fait deja le wrapper vlc de l'etape 8e. Deux chemins valent mieux qu'un
# quand le symptome d'un echec est un navigateur muet sans message.
PULSE=""
[ -S /run/pulse/native ] && PULSE="unix:/run/pulse/native"

# ── LE TROUSSEAU ──────────────────────────────────────────────────────
# Demarre sous le compte proprietaire, donc AVEC son home chiffre : c'est la
# seule facon d'avoir un service Secrets dont le magasin n'est pas sur le
# disque en clair. Deja lance = ne fait rien.
runuser -u "$OWNER" -- env -i \
    HOME="$OWNER_HOME" USER="$OWNER" LOGNAME="$OWNER" \
    XDG_RUNTIME_DIR="$RUNTIME" ${BUS:+DBUS_SESSION_BUS_ADDRESS="$BUS"} \
    PATH=/usr/local/bin:/usr/bin:/bin \
    gnome-keyring-daemon --start --components=secrets,ssh >/dev/null 2>&1 || true

exec runuser -u "$OWNER" -- env -i \
    HOME="$OWNER_HOME" USER="$OWNER" LOGNAME="$OWNER" SHELL="$OWNER_SHELL" \
    PATH=/usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/sbin:/sbin \
    DISPLAY="${DISPLAY:-:0}" \
    XDG_RUNTIME_DIR="$RUNTIME" \
    ${BUS:+DBUS_SESSION_BUS_ADDRESS="$BUS"} \
    XDG_SESSION_TYPE=x11 \
    ${PULSE:+PULSE_SERVER="$PULSE"} \
    XDG_CURRENT_DESKTOP="${XDG_CURRENT_DESKTOP:-ubuntu:GNOME}" \
    XDG_DATA_DIRS="${XDG_DATA_DIRS:-/usr/local/share:/usr/share}" \
    LANG="${LANG:-fr_FR.UTF-8}" LANGUAGE="${LANGUAGE:-}" \
    "$@"
```

3.83 /opt/GSM/osmo-operator/configs/crypthome/unlock-home

19607 octets, 415 lignes → 415 lignes

```
#!/bin/bash
# unlock-home - dechiffre et monte /home depuis le second disque (LUKS).
#
#   unlock-home --boot   demande la phrase au demarrage, sur la console texte
#   unlock-home          la meme chose depuis un terminal, apres coup
#
# CE SCRIPT NE RATE JAMAIS LE BOOT. Entree a vide, l absence du
# disque, une phrase de passe fausse trois fois ou un montage impossible, il
# sort 0 et la machine demarre - sur un /home vide. C est le point d Entree a vide :
# le systeme reste utilisable en root, dont le home (/root) est sur le disque
# systeme en clair.
#
# ET LE BUREAU S OUVRE SUR ROOT. [2026-09-25] Sur un disque installe, le compte
# de session a son home SUR le volume chiffre : sans /home, GDM l ouvrirait sur
# un home inexistant. Quand /home reste ferme au demarrage (Entree, delai, phrase
# refusee, disque absent), GDM ouvre donc une session ROOT, pour ce demarrage
# seulement - voir session_root plus bas.
set -u

CONF=/etc/default/crypthome
[ -r "$CONF" ] || { echo "unlock-home: $CONF introuvable" >&2; exit 0; }
. "$CONF"

MAPPER="${CRYPTHOME_NAME:-crypthome}"
MOUNT="${CRYPTHOME_MOUNT:-/home}"
```

```
TRIES="${CRYPTHOME_TRIES:-3}"
TIMEOUT="${CRYPTHOME_TIMEOUT:-60}"
OWNER="${CRYPTHOME_OWNER:-}"

BOOT=0
[ "${1:-}" = "--boot" ] && BOOT=1

log() { echo "unlock-home: $*" >&2; }

# ── LE MONTAGE DOIT VENIR DU VOLUME CHIFFRE ──────────────────────────
# "monte" ne suffit pas. /home peut etre une partition ORDINAIRE - une machine
# installee avec un /home separe, ou init-crypthome.sh n a jamais tourne. Traiter
# ce montage-la comme le disque chiffre ecrirait les secrets EN CLAIR sous le
# compte proprietaire tout en annoncant le contraire : le pire des etats, parce
# qu il a l air correct. On exige donc que la source du montage soit le mapping
# dechiffre lui-meme, et pas seulement qu il y ait quelque chose sur /home.
crypthome_monte() {
    local _src
    mountpoint -q "$MOUNT" || return 1
    _src="$(findmnt -fno SOURCE "$MOUNT" 2>/dev/null)"
    [ -n "$_src" ] || return 1
    [ "$(readlink -f "$_src")" = "$(readlink -f "/dev/mapper/$MAPPER")" ]
}

# ── LA SESSION ROOT QUAND /home RESTE FERME ──────────────────────────
# GDM n a pas de configuration "pour ce demarrage" : il lit custom.conf. On le
# modifie donc AVANT lui (unlock-home.service porte Before=display-manager), en
# mettant de cote les fichiers d origine dans $ETAT_GDM. Chaque demarrage
# commence par les remettre (session_normale) : la session root ne dure qu un
# demarrage, et une coupure de courant ne la rend pas permanente.
#
# Deux fichiers, parce que la cle live l a appris (80-chroot.sh) : GDM refuse
# l uid 0 non dans custom.conf mais dans PAM, par la regle
# "pam_succeed_if.so user != root". On ne la neutralise QUE dans gdm-autologin :
# l ecran de connexion avec mot de passe, lui, continue de refuser root.
#
# Sans danger ajoute : la racine est en clair par construction (partition.conf),
# qui a la machine en main la lit deja depuis une cle live. Ce qui est protege,
# c est /home - et c est justement lui qui n est pas monte.
GDM_CONF=/etc/gdm3/custom.conf
GDM_PAM=/etc/pam.d/gdm-autologin
ETAT_GDM=/var/lib/crypthome/gdm-avant-session-root

session_normale() {
    [ -d "$ETAT_GDM" ] || return 0
    if [ -f "$ETAT_GDM/custom.conf" ]; then
        cp -p "$ETAT_GDM/custom.conf" "$GDM_CONF"
    elif [ -f "$ETAT_GDM/custom.conf.absent" ]; then
        rm -f "$GDM_CONF"
    fi
    [ -f "$ETAT_GDM/gdm-autologin" ] && cp -p "$ETAT_GDM/gdm-autologin" "$GDM_PAM"
    rm -rf "$ETAT_GDM"
    log "GDM : configuration du compte de session retablie."
}

session_root() {
    [ "$BOOT" = 1 ] || return 0
    [ -d /etc/gdm3 ] || return 0
    # Deja fait (deux appels dans le meme demarrage) : ne pas ecraser la
    # sauvegarde des originaux par la version deja modifiee.
    if [ ! -d "$ETAT_GDM" ]; then
        install -d -m 0700 "$ETAT_GDM" || return 0
        if [ -f "$GDM_CONF" ]; then
            cp -p "$GDM_CONF" "$ETAT_GDM/custom.conf"
        else
            : > "$ETAT_GDM/custom.conf.absent"
```

```
        fi
        [ -f "$GDM_PAM" ] && cp -p "$GDM_PAM" "$ETAT_GDM/gdm-autologin"
    fi
    # Les deux cles, et elles seules, dans [daemon] : le reste (WaylandEnable,
    # le TimedLogin eventuel) est la configuration de la machine.
    awk '
        /^[[:space:]]*AutomaticLogin(Enable)?[[:space:]]*=/ { next }
        { print }
        /^[[:space:]]*\[daemon\][[:space:]]*$/ && !fait {
            print "AutomaticLoginEnable=true"; print "AutomaticLogin=root"; fait = 1
        }
        END { if (!fait) { print "[daemon]"; print "AutomaticLoginEnable=true"; print "AutomaticLogin=root" } }
    ' "${GDM_CONF}" 2>/dev/null > "$GDM_CONF.crypthome" \
        && mv "$GDM_CONF.crypthome" "$GDM_CONF"
    [ -f "$GDM_PAM" ] && \
        sed -i '/^[[:space:]]*auth.*pam_succeed_if.so user != root quiet_success/s/^/#/' "$GDM_PAM"
    log "GDM : session root pour ce demarrage ($MOUNT ferme)."
    root_dans_tmp
}

# ── CE QUE LA SESSION ROOT ECRIT RESTE DANS /tmp ──────────────────────
# Sans /home, la session root ne doit rien laisser sur le disque : ce qu elle
# ecrit dans son home (/root) - profils, historique, fichiers - vit dans
# $RAM_ROOT, un tmpfs sous /tmp, et disparait a l arret. /root est recouvert
# par un overlay : son contenu reste LISIBLE (lanceurs, scripts du banc,
# init-crypthome.sh), seules les ecritures partent dans /tmp.
#
# ET /opt/GSM AVEC LUI. [2026-10-14] Le banc entier vit dans /opt/GSM : depot,
# configs, hlr.db, journaux, captures. En session root (/home ferme), tout ce
# que le banc y ecrit part de la meme facon dans la RAM : /opt/GSM est "live",
# non persistant, et retrouve son etat du disque au redemarrage. La liste des
# repertoires ainsi recouverts est CRYPTHOME_RAM_DIRS (/etc/default/crypthome) ;
# /root en fait toujours partie. Un repertoire absent est ignore ; un
# repertoire qui est lui-meme un point de montage (second disque sur /opt/GSM,
# via fstab) se recouvre comme un autre : local-fs.target est passe.
#
# tmpfs et pas un simple repertoire de /tmp : sur Ubuntu /tmp est sur le disque
# racine, en clair ; en RAM, rien ne touche jamais un disque.
#
# Le trousseau : le lien /root/.local/share/keyrings pointe vers le volume
# ferme. Il est remplace, DANS l overlay seulement, par un vrai repertoire : la
# session a un trousseau qui marche, et qui s evapore avec elle.
RAM_ROOT=/tmp/crypthome-root
RAM_DIRS="${CRYPTHOME_RAM_DIRS-/opt/GSM}"

# en_ram REPERTOIRE : recouvre REPERTOIRE d un overlay dont upper et work
# vivent dans $RAM_ROOT. Mode et proprietaire de l upper = ceux du repertoire :
# la racine d un overlay prend les attributs de son upper, et un /opt/GSM passe
# en 0700 couperait asterisk (osmo-banc) et le compte de session de leur
# repertoire de travail.
en_ram() {
    local dir="$1" nom
    [ -d "$dir" ] || return 1
    [ "$(findmnt -fno FSTYPE "$dir" 2>/dev/null)" = overlay ] && return 0   # deja fait ce demarrage
    nom="$(printf '%s' "$dir" | sed 's#^/##; s#/#_#g')"
    install -d -m "$(stat -c %a "$dir")" -o "$(stat -c %u "$dir")" -g "$(stat -c %g "$dir")" \
            "$RAM_ROOT/$nom/upper" || return 1
    install -d -m 0700 "$RAM_ROOT/$nom/work" || return 1
    mount -t overlay overlay \
          -o "lowerdir=$dir,upperdir=$RAM_ROOT/$nom/upper,workdir=$RAM_ROOT/$nom/work" "$dir"
}

root_dans_tmp() {
    local lien d
    if ! mountpoint -q "$RAM_ROOT"; then
        install -d -m 0700 "$RAM_ROOT" || return 0
```

```
        if ! mount -t tmpfs -o mode=0700,size=50% tmpfs "$RAM_ROOT"; then
            log "ATTENTION : tmpfs $RAM_ROOT impossible - la session root ecrira sur le disque."
            return 0
        fi
        # systemd-tmpfiles-clean nettoie /tmp par age : il ne doit pas vider le
        # home d une session en cours.
        install -d /run/tmpfiles.d
        echo "x $RAM_ROOT" > /run/tmpfiles.d/crypthome-root.conf
    fi
    if en_ram /root; then
        for lien in ${CRYPTHOME_ROOT_LINKS:-.local/share/keyrings}; do
            if [ -L "/root/$lien" ]; then
                rm -f "/root/$lien"
                install -d -m 0700 "/root/$lien"
            fi
        done
        log "session root : les ecritures dans /root restent dans $RAM_ROOT (RAM)."
    else
        log "ATTENTION : overlay sur /root impossible - la session root ecrira sur le disque."
    fi
    for d in $RAM_DIRS; do
        [ "$d" = /root ] && continue
        [ -d "$d" ] || continue
        if en_ram "$d"; then
            log "session root : les ecritures dans $d restent dans $RAM_ROOT (RAM)."
        else
            log "ATTENTION : overlay sur $d impossible - $d ecrira sur le disque."
        fi
    done
}

# Chaque demarrage repart de la configuration du compte de session.
[ "$BOOT" = 1 ] && session_normale

# Deja monte (2e appel, ou dechiffrement manuel apres un "non") : rien a faire.
#
# Et si c est un AUTRE montage - un /home separe en clair - on sort aussi, sans
# rien deplacer : evacuer le contenu d une partition que l operateur n a pas
# demande a chiffrer serait une surprise, et la monter par-dessus, impossible.
# Le message dit laquelle, parce que le symptome sinon est "il ne demande
# jamais la phrase de passe" sans aucune explication.
if mountpoint -q "$MOUNT"; then
    if crypthome_monte; then
        log "$MOUNT est deja monte depuis le volume chiffre."
    else
        log "$MOUNT est deja monte, mais pas depuis /dev/mapper/$MAPPER" \
            "($(findmnt -fno SOURCE "$MOUNT" 2>/dev/null)) - rien n a ete touche."
    fi
    exit 0
fi

if [ -z "${CRYPTHOME_UUID:-}" ]; then
    log "aucun disque chiffre configure (CRYPTHOME_UUID vide dans $CONF)."
    exit 0
fi

# ── L ANCIEN /home EN CLAIR N A PLUS LIEU D ETRE ──────────────────────────
# Ce qui reste sous le point de montage devient invisible une fois le disque
# monte : invisible, mais toujours la - place occupee sur le disque systeme et,
# ce qui compte ici, ANCIENS TROUSSEAUX EN CLAIR. On l'evacue AVANT de monter.
#
# C est fait ICI et pas par init-crypthome.sh parce que c est le seul moment ou
# personne ne tient /home : au demarrage, avant la premiere session. Un
# init-crypthome.sh lance depuis un bureau, lui, travaille sur un /home que sa
# propre session occupe.
#
```

```
# Inconditionnel, et non "seulement si on repond oui" : des lors qu un disque
# chiffre est configure, le /home en clair a cesse d etre le bon. Repondre non
# doit laisser un /home VIDE, pas l ancien, sinon le "non" rendrait justement
# les secrets qu on voulait mettre a l abri.
if [ -d "$MOUNT" ] && [ -n "$(ls -A "$MOUNT" 2>/dev/null)" ]; then
    _sauve="/var/backups/home-avant-chiffrement-$(date +%Y%m%d-%H%M%S)"
    if install -d -m 0700 "$_sauve" 2>/dev/null; then
        ( shopt -s dotglob nullglob
          _reste=("$MOUNT"/*)
          [ "${#_reste[@]}" -gt 0 ] && mv "${_reste[@]}" "$_sauve"/ ) 2>/dev/null
        log "ancien contenu de $MOUNT mis de cote dans $_sauve."
    else
        log "ATTENTION : $MOUNT n est pas vide et n a pas pu etre evacue."
    fi
fi

DEV="/dev/disk/by-uuid/$CRYPTHOME_UUID"
if [ ! -e "$DEV" ]; then
    log "disque chiffre absent (UUID=$CRYPTHOME_UUID) - $MOUNT reste vide."
    session_root
    exit 0
fi

# ── LA SEULE QUESTION : LA PHRASE DE PASSE ──────────────────────────────
# [2026-10-01] Plus de « Dechiffrer /home ? [o/N] » suivi d une seconde invite.
# Une seule saisie, qui vaut reponse ET phrase de passe :
#   - ENTREE a vide (ou le delai qui expire)  : on demarre en ROOT, sans /home ;
#   - une phrase                              : on ouvre le volume, et la
#     session est celle du compte proprietaire.
# Une phrase refusee repose la question, $TRIES fois ; a tout moment, Entree a
# vide sort en root. Le delai qui expire vaut Entree : une machine qui
# redemarre seule finit son boot au lieu d attendre indefiniment.
#
# L invite tient en une ligne, la meme partout : « decrypt /home -passphrase-
# or Enter ». Ce qu on tape s affiche en « * » (lire_masquee ci-dessous -
# « read -s » n afficherait rien du tout, et l operateur ne saurait pas si sa
# frappe est prise).
#
# OU ELLE EST POSEE : SUR LA CONSOLE TEXTE, PAS DANS PLYMOUTH. [2026-10-01]
# Sur un disque installe par Calamares (« quiet splash »), la question passait
# par systemd-ask-password, donc par l agent plymouth de systemd et le theme
# osmo-bts : l invite s affichait, le curseur clignotait, et AUCUNE touche
# n arrivait - le delai expirait et la session root s ouvrait. plymouth
# 24.004.60 lit le clavier par son propre chemin (le tty qu il tient, ou
# evdev + xkb), et sur ce materiel (portable Intel + NVIDIA, pilote
# proprietaire charge apres l initrd) ce chemin est mort quand unlock-home
# tourne. On ne depend plus de lui : si plymouth tourne, on le QUITTE (le
# decor a fait son office), la console texte revient, et on lit la phrase
# NOUS-MEMES sur /dev/console - ni agent, ni condition d unite, ni clavier de
# plymouth. GDM demarre ensuite comme d habitude (son « plymouth quit
# --retain-splash » ne trouve plus personne et s en passe).
#
# CRYPTHOME_PROMPT=plymouth dans /etc/default/crypthome remet l ancien chemin
# (systemd-ask-password -> agent plymouth -> theme osmo-bts), pour une machine
# ou le clavier de plymouth repond. --key-file=- dans les deux cas : la phrase
# ne passe ni par la ligne de commande (visible dans ps) ni par le disque,
# seulement par un tube.
lire_masquee() {
    # lire_masquee VARIABLE INVITE : lit une ligne au clavier, un « * » par
    # caractere, Retour arriere pris en compte. Retourne 1 si le delai expire
    # (la variable est alors vide).
    local _var="$1" _invite="$2" _c _s="" _rc
    printf '%s' "$_invite" >&2
    while :; do
        IFS= read -r -s -n1 -t "$TIMEOUT" _c; _rc=$?
        if [ "$_rc" -gt 128 ]; then
```

```
            echo >&2; printf -v "$_var" '%s' ""; return 1
        fi
        [ "$_rc" -ne 0 ] && break            # fin d entree
        [ -z "$_c" ] && break                # Entree
        if [ "$_c" = $'\x7f' ] || [ "$_c" = $'\b' ]; then
            if [ -n "$_s" ]; then _s="${_s%?}"; printf '\b \b' >&2; fi
            continue
        fi
        _s+="$_c"; printf '*' >&2
    done
    echo >&2
    printf -v "$_var" '%s' "$_s"
    return 0
}

# La console ou l on pose la question au demarrage. /dev/console suit le
# « console= » du noyau (tty0 = la console virtuelle active, ou un port serie
# si c est lui qui a ete demande) ; a defaut, tty1.
CONSOLE=/dev/console
[ -c "$CONSOLE" ] || CONSOLE=/dev/tty1

demander_console() {
    # demander_console VARIABLE INVITE : la question sur la console texte, au
    # demarrage. plymouth, s il tourne, est quitte d abord : l ecran repasse en
    # texte et le clavier revient au tty.
    if command -v plymouth >/dev/null 2>&1 && plymouth --ping 2>/dev/null; then
        plymouth quit 2>/dev/null || true
        log "plymouth quitte - la question passe sur la console ($CONSOLE)."
    fi
    # [2026-10-24] UNE IMAGE DERRIERE LA QUESTION. La console texte efface (ESC[2J), crypthome-fond
    # peint CRYPTHOME_IMAGE sur /dev/fb0 (assombrie en bas), et l invite part sur la DERNIERE ligne
    # (ESC[999;1H) : fbcon ne repeint que les cases qu il ecrit. La saisie ne change pas, c est
    # toujours lire_masquee sur le tty. CRYPTHOME_IMAGE=none pour s en passer.
    if [ "${CRYPTHOME_IMAGE:-}" != none ] && [ -x /usr/local/sbin/crypthome-fond ]; then
        printf '\033[2J' > "$CONSOLE" 2>/dev/null
        /usr/local/sbin/crypthome-fond ${CRYPTHOME_IMAGE:+"$CRYPTHOME_IMAGE"} 2>&1 | while read -r l; do log "$l";
         ↪  done
        printf '\033[999;1H' > "$CONSOLE" 2>/dev/null
    fi
    lire_masquee "$1" "$2" < "$CONSOLE" 2> "$CONSOLE"
}

# ── LE CLAVIER, DIT DANS LE JOURNAL ──────────────────────────────────────
# Une phrase refusee trois fois, c est le plus souvent un clavier qui n est
# pas celui qu on croit : une image construite en « fr » tapee comme un QWERTY,
# ou l inverse. La console a ce moment-la a la disposition de
# /etc/default/keyboard (ecrite par build-iso, etape 10-clavier, puis par
# Calamares a l installation) - on la note dans le journal, pour que
# « phrase refusee » se lise avec la disposition en face. A defaut :
# /etc/vconsole.conf, puis « ? ».
clavier_console() {
    local _l _v
    _l="$(awk -F'"' '/^XKBLAYOUT=/{print $2}' /etc/default/keyboard 2>/dev/null)"
    _v="$(awk -F'"' '/^XKBVARIANT=/{print $2}' /etc/default/keyboard 2>/dev/null)"
    if [ -z "$_l" ]; then
        _l="$(awk -F= '/^KEYMAP=/{print $2}' /etc/vconsole.conf 2>/dev/null | tr -d '"')"
        _v=""
    fi
    [ -n "$_l" ] || { echo "?"; return 0; }
    _l="${_l%%,*}"; _v="${_v%%,*}"
    [ -n "$_v" ] && _l="$_l-$_v"
    echo "$_l"
}
log "clavier de la console : $(clavier_console)."

PROMPT_MODE="${CRYPTHOME_PROMPT:-console}"
```

```
ouvert=0
essai=0
while [ "$essai" -lt "$TRIES" ]; do
    essai=$((essai + 1))
    invite="decrypt $MOUNT -passphrase- or Enter"
    [ "$essai" -gt 1 ] && invite="$invite ($essai/$TRIES)"
    if [ "$BOOT" = 1 ] && [ "$PROMPT_MODE" = plymouth ]; then
        passe="$(systemd-ask-password --timeout="$TIMEOUT" \
                    --id="crypthome:passphrase" "$invite" 2>/dev/null)" || passe=""
    elif [ "$BOOT" = 1 ]; then
        demander_console passe "$invite : " || passe=""
    else
        lire_masquee passe "$invite : " || passe=""
    fi
    if [ -z "$passe" ]; then
        unset passe
        log "Entree sans phrase - demarrage en root sans $MOUNT (« unlock-home » pour le faire plus tard)."
        session_root
        exit 0
    fi
    if printf '%s' "$passe" | cryptsetup open --key-file=- "$DEV" "$MAPPER"; then
        passe=""; unset passe
        ouvert=1
        break
    fi
    passe=""; unset passe
    log "phrase de passe refusee."
done

if [ "$ouvert" != 1 ]; then
    log "abandon apres $TRIES essais - demarrage en root sans $MOUNT."
    session_root
    exit 0
fi

# ── 3. LE MONTAGE ──────────────────────────────────────────────────────
# La ligne /home de /etc/fstab porte "noauto" : c'est ce script qui monte, et
# lui seul. Si le montage echoue, on REFERME le mapping - laisser un volume
# dechiffre ouvert et non monte serait le pire des deux etats.
if ! mount "$MOUNT" 2>/dev/null && ! mount "/dev/mapper/$MAPPER" "$MOUNT"; then
    log "montage de $MOUNT impossible - le volume est referme."
    cryptsetup close "$MAPPER" 2>/dev/null || true
    session_root
    exit 0
fi

# Le compte proprietaire peut ouvrir une session utilisateur des maintenant :
# son home existe enfin. Sans lingering, /run/user/<uid> et son bus ne sont
# crees qu'a la premiere connexion - or run-as-owner lance ses applications
# depuis la session de root, sans jamais connecter ce compte.
if [ -n "$OWNER" ] && id -u "$OWNER" >/dev/null 2>&1; then
    loginctl enable-linger "$OWNER" >/dev/null 2>&1 || true
fi

# /home ouvert : la session est celle du compte, persistante sur le volume
# chiffre. Utile aussi hors demarrage - « unlock-home » tape depuis la session
# root remet GDM sur le compte pour la prochaine connexion.
session_normale

log "$MOUNT dechiffre et monte."
exit 0
```

3.84 /opt/GSM/osmo-operator/configs/crypthome/unlock-home.service

1353 octets, 30 lignes → 30 lignes

```
[Unit]
Description=Dechiffrement optionnel de /home (second disque, LUKS)
Documentation=file:/etc/default/crypthome
ConditionPathExists=/etc/default/crypthome
# APRES les systemes de fichiers locaux et udev (le lien /dev/disk/by-uuid
# doit exister), APRES plymouth (unlock-home le quitte avant de poser la
# question sur la console - il doit donc etre deja la, pas en train de
# demarrer), et APRES la mise en place du clavier : la disposition de
# /etc/default/keyboard doit etre celle qui est chargee quand on tape la phrase.
After=local-fs.target systemd-udev-settle.service plymouth-start.service
After=keyboard-setup.service console-setup.service
Wants=systemd-udev-settle.service
# AVANT tout ce qui ouvre une session : GDM ne doit pas demarrer sur un /home
# vide alors que l'operateur vient de repondre "oui".
Before=systemd-user-sessions.service display-manager.service multi-user.target graphical.target

[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/usr/local/sbin/unlock-home --boot
# La question et la saisie passent par /dev/console (ou, CRYPTHOME_PROMPT=
# plymouth, par systemd-ask-password), jamais par stdin.
StandardInput=null
StandardOutput=journal+console
StandardError=journal+console
# Large, mais borne : unlock-home applique deja son propre delai par question.
TimeoutStartSec=600

[Install]
WantedBy=multi-user.target
```

3.85 /opt/GSM/osmo-operator/configs/gprs/modem-gprs.cfg

890 octets, 35 lignes → 35 lignes

```
! modem-gprs.cfg - MS GPRS du banc (osmocom-bb modem), posee par scripts/egprs-dsp.sh
! IMSI 001010001000001 · MSISDN 100101 (celui du mobile : un seul des deux sur le Calypso) · APN internet ->
 ↪  tun_modem · ARFCN 714
line vty
 no login
 bind 127.0.0.1 4299
log stderr
 logging filter all 1
 logging color 0
 logging timestamp 1
 logging print category 1
 logging print level 1
 logging level all notice
 logging level cs info
 logging level rr debug
 logging level rlcmac info
 logging level gmm info
 logging level sm info
 logging level llc notice
 logging level sndcp notice
ms 1
 layer2-socket /tmp/modem_l2
 sim test
 imei 358925008659483 0
```

```
imei-fixed
test-sim
 imsi 001010001000001
 ki comp128 00 11 22 33 44 55 66 77 88 99 aa bb cc dd 01 01
 no barred-access
 rplmn 001 01
apn internet
 tun-device tun_modem
 tun-netns msgprs
 type-support v4
 no shutdown
no shutdown
```

3.86 /opt/GSM/osmo-operator/configs/gprs/osmo-bsc-bts2-egprs.frag

2102 octets, 81 lignes → 81 lignes

```
bts 2
 type osmo-bts
 band DCS1800
 cell_identity 6013
 location_area_code 0x0001
 base_station_id_code 9
 ms max power 15
 cell reselection hysteresis 4
 rxlev access min 0
 radio-link-timeout 32
 channel allocator mode chan-req descending
 channel allocator mode assignment ascending
 channel allocator mode handover ascending
 rach tx integer 9
 rach max transmission 7
 ipa unit-id 6013 0
 oml ipa stream-id 255 line 0
 neighbor-list mode automatic
 codec-support fr
 gprs mode egprs
 gprs routing area 0
 gprs network-control-order nc0
 gprs power-control alpha 0
 gprs cell bvci 13
 gprs cell timer blocking-timer 3
 gprs cell timer blocking-retries 3
 gprs cell timer unblocking-retries 3
 gprs cell timer reset-timer 3
 gprs cell timer reset-retries 3
 gprs cell timer suspend-timer 10
 gprs cell timer suspend-retries 3
 gprs cell timer resume-timer 10
 gprs cell timer resume-retries 3
 gprs cell timer capability-update-timer 10
 gprs cell timer capability-update-retries 3
 gprs nsei 11
 gprs ns timer tns-block 3
 gprs ns timer tns-block-retries 3
 gprs ns timer tns-reset 3
 gprs ns timer tns-reset-retries 3
 gprs ns timer tns-test 30
 gprs ns timer tns-alive 3
 gprs ns timer tns-alive-retries 10
 gprs nsvc 0 nsvci 11
 gprs nsvc 0 local udp port 23002
 gprs nsvc 0 remote ip __SGSN_IP__
 gprs nsvc 0 remote udp port 23000
```

```
  gprs nsvc 1 nsvci 0
  gprs nsvc 1 local udp port 0
  gprs nsvc 1 remote ip 0.0.0.0
  gprs nsvc 1 remote udp port 0
  trx 0
   rf_locked 0
   arfcn 714
   nominal power 23
   max_power_red 0
   rsl e1 tei 0
   timeslot 0
    phys_chan_config CCCH+SDCCH4
    hopping enabled 0
   timeslot 1
    phys_chan_config SDCCH8
    hopping enabled 0
   timeslot 2
    phys_chan_config TCH/F
    hopping enabled 0
   timeslot 3
    phys_chan_config TCH/F
    hopping enabled 0
   timeslot 4
    phys_chan_config TCH/F
    hopping enabled 0
   timeslot 5
    phys_chan_config PDCH
    hopping enabled 0
   timeslot 6
    phys_chan_config PDCH
    hopping enabled 0
   timeslot 7
    phys_chan_config PDCH
    hopping enabled 0
```

3.87 /opt/GSM/osmo-operator/configs/gprs/osmo-bts-virtual-gprs.cfg

1131 octets, 43 lignes → 43 lignes

```
! osmo-bts-virtual-gprs.cfg - BTS 2 virtuelle (Um multicast sur lo), cellule EGPRS du modem.
! Posee dans /etc/osmocom par scripts/egprs-dsp.sh. PCU : /tmp/pcu_bts3 (osmo-pcu-gprs.cfg).
! osmo-bts-virtual-gprs.cfg - BTS virtuel du MS GPRS, ecrit par 13-virt-bts-cfg.sh
! Um virtuel : GSMTAP multicast 239.193.23.1/2 udp 4729 sur lo (virtphy -D lo)
log file /var/log/osmocom/osmo-bts-virtual-gprs.log
 logging filter all 1
 logging color 0
 logging timestamp 1
 logging print category 1
 logging print level 1
 logging level all notice
 logging level pcu info
line vty
 no login
 bind 127.0.0.1 4253
ctrl
 bind 127.0.0.3
e1_input
 e1_line 0 driver ipa
 e1_line 0 port 0
 no e1_line 0 keepalive
phy 0
 virtual-um net-device lo
 virtual-um ttl 1
 instance 0
```

```
bts 0
 band DCS1800
 ipa unit-id 6013 0
 oml remote-ip 127.0.0.1
 pcu-socket /tmp/pcu_bts3
 rtp jitter-buffer 100
 paging queue-size 200
 paging lifetime 0
 min-qual-rach 50
 min-qual-norm -5
 gsmtap-sapi ccch
 gsmtap-sapi pdtch
 trx 0
  power-ramp max-initial 23000 mdBm
  power-ramp step-size 2000 mdB
  power-ramp step-interval 1
  ms-power-control osmo
  phy 0 instance 0
```

3.88 /opt/GSM/osmo-operator/configs/gprs/osmo-pcu-gprs.cfg

613 octets, 25 lignes → 25 lignes

```
! osmo-pcu-gprs.cfg - PCU de la BTS 2 virtuelle (socket /tmp/pcu_bts3, VTY 127.0.0.2:4240).
! Posee dans /etc/osmocom par scripts/egprs-dsp.sh ; __GSMTAP_IP__ = celui de osmo-pcu.cfg.
log file /var/log/osmocom/osmo-pcu-gprs.log
 logging color 1
 logging timestamp 1
 logging filter all 1
 logging print category 1
 logging print level 1
log gsmtap __GSMTAP_IP__
 logging filter all 1
 logging color 0
 logging timestamp 0
 logging print category 1
 logging print level 1
!
pcu
 flow-control-interval 10
 cs 2
 alloc-algorithm dynamic
 alpha 0
 gamma 0
 pcu-socket /tmp/pcu_bts3
line vty
 no login
 bind 127.0.0.2
```

3.89 /opt/GSM/osmo-operator/configs/open5gs/hss.yaml

533 octets, 19 lignes → 19 lignes

```
db_uri: mongodb://localhost/open5gs
logger:
  file:
    path: /opt/LTE/open5gs/install/var/log/open5gs/hss.log
#  level: info   # fatal|error|warn|info(default)|debug|trace

global:
  max:
    ue: 1024  # The number of UE can be increased depending on memory size.
```

```
#    peer: 64

hss:
  freeDiameter: /opt/LTE/open5gs/install/etc/freeDiameter/hss.conf
  metrics:
    server:
      - address: 127.0.0.8
        port: 9090
#  sms_over_ims: "sip:smsc.mnc001.mcc001.3gppnetwork.org:7060;transport=tcp"
#  use_mongodb_change_stream: true
```

3.90 /opt/GSM/osmo-operator/configs/open5gs/mme.yaml

3741 octets, 105 lignes → 105 lignes

```
# mme.yaml - le MME open5gs du banc (Open5GS v2.8.0, /opt/LTE/open5gs/install).
#
# [2026-09-08] REECRIT A PLAT. Le fichier livre par open5gs fait 360 lignes
# dont 300 d exemples commentes (roaming, DNS, SGSN, IPv6...) qui n ont rien
# a voir avec ce banc : on ne gardait plus que les 60 lignes utiles, noyees.
# Ici, seulement ce qui tourne, et pourquoi. La source est dans le depot,
# configs/open5gs/mme.yaml ; la copie en service est
# /opt/LTE/open5gs/install/etc/open5gs/mme.yaml (cp, puis relancer open5gs-mmed).
#
# LE PLAN D ADRESSES : tout le coeur 4G vit sur la boucle locale, une adresse
# par fonction, pour que les ports 2123/2152/8805 ne se marchent pas dessus :
#   127.0.0.2  MME (S1AP sctp 36412, GTP-C 2123, Diameter S6a, SGs vers le MSC)
#   127.0.0.3  SGW-C     127.0.0.4  SMF (= PGW-C)     127.0.0.6  SGW-U
#   127.0.0.7  UPF       127.0.0.8  HSS               127.0.0.9  PCRF
#   127.0.0.1  osmo-msc (SGs 29118), osmo-stp, osmo-sgsn - la 2G d osmocom.
# srsENB (ZeroMQ, 127.0.1.1) vient sur le S1AP de 127.0.0.2 (enb.conf : mme_addr).
#
# PIEGE VU LE 2026-09-08 : le PGW n est PAS choisi que par gtpc.client.smf.
# Si la souscription HSS (MongoDB, open5gs.subscribers, slice[].session[].smf)
# porte une adresse, elle l emporte - le WebUI y met 127.0.0.200 par defaut,
# et le SGW-C attendait un Create Session Response de personne (GTP Timeout,
# RRC Release, « Attach failed (attempt N/5) » cote srsUE). Laisser ce champ
# vide dans la base.

logger:
  file:
    path: /opt/LTE/open5gs/install/var/log/open5gs/mme.log

global:
  max:
    ue: 1024          # largement assez : un srsUE et le telephone du banc

mme:
  freeDiameter: /opt/LTE/open5gs/install/etc/freeDiameter/mme.conf   # S6a vers le HSS (127.0.0.8)

  # S1AP : ce qu ecoute le MME pour les eNB. srsENB s y connecte en SCTP.
  s1ap:
    server:
      - address: 127.0.0.2

  # GTP-C (S11 vers le SGW-C, S5 vers le SMF quand le HSS ne dit rien).
  gtpc:
    server:
      - address: 127.0.0.2
    client:
      sgwc:
        - address: 127.0.0.3
      smf:
        - address: 127.0.0.4
```

```
  metrics:
    server:
      - address: 127.0.0.2
        port: 9090

  # SGs : LE LIEN AVEC LA 2G. C est par lui que l UE pose sur la 4G est aussi
  # inscrit au VLR d osmo-msc (attach combine EPS + IMSI), et que le CSFB
  # marche : appel ou SMS, l UE redescend sur la BTS ; entre-temps le MSC
  # sait ou le joindre (paging par le MME). osmo-msc ecoute en SCTP sur
  # 127.0.0.1:29118 (osmo-msc.cfg, section « sgs »). La table « map » traduit
  # la zone 4G (TAI : PLMN 001-01, TAC 7 - celui de rr.conf de srsENB) en zone
  # 2G (LAI : PLMN 001-01, LAC 1 - location_area_code 0x0001 dans osmo-bsc.cfg) :
  # le VLR classe l abonne dans la LAC 1 alors qu il est sur la LTE.
  sgsap:
    client:
      - address: 127.0.0.1
        local_address: 127.0.0.2
        map:
          tai:
            plmn_id:
              mcc: 001
              mnc: 01
            tac: 7
          lai:
            plmn_id:
              mcc: 001
              mnc: 01
            lac: 1

  # L identite du MME (GUMMEI) : c est ce que porte le GUTI donne a l UE.
  gummei:
    - plmn_id:
        mcc: 001
        mnc: 01
      mme_gid: 2
      mme_code: 1

  # Les zones servies : une seule, celle qu annonce srsENB.
  tai:
    - plmn_id:
        mcc: 001
        mnc: 01
      tac: 7

  # Securite NAS : integrite obligatoire (EIA2 d abord), chiffrement nul
  # (EEA0) en tete - on veut LIRE le NAS dans Wireshark sur ce banc.
  security:
    integrity_order : [ EIA2, EIA1, EIA0 ]
    ciphering_order : [ EEA0, EEA1, EEA2 ]

  # Le nom que le telephone affiche pour la 4G (EMM Information).
  network_name:
    full: OsmoDirect
    short: OsmoDirect
  mme_name: open5gs-mme0
```

3.91 /opt/GSM/osmo-operator/configs/open5gs/nrf.yaml

3205 octets, 101 lignes → 101 lignes

```
logger:
  file:
    path: /opt/LTE/open5gs/install/var/log/open5gs/nrf.log
```

```
#  level: info   # fatal|error|warn|info(default)|debug|trace

global:
  max:
    ue: 1024  # The number of UE can be increased depending on memory size.
#    peer: 64

nrf:
  serving:  # 5G roaming requires PLMN in NRF
    - plmn_id:
        mcc: 999
        mnc: 70
  sbi:
    server:
      - address: 127.0.0.10
        port: 7777

#########################################################################################
# SBI Server
#########################################################################################
#  o Bind to the address on the eth0 and advertise as open5gs-nrf.svc.local
#  sbi:
#    server:
#      - dev:eth0
#        advertise: open5gs-nrf.svc.local
#
#  o Specify a custom port number 7777 while binding to the given address
#  sbi:
#    server:
#      - address: nrf.localdomain
#        port: 7777
#
#  o Bind to 127.0.0.10 and advertise as open5gs-nrf.svc.local
#  sbi:
#    server:
#      - address: 127.0.0.10
#        port: 7777
#        advertise: open5gs-nrf.svc.local
#
#  o Bind to port 7777 but advertise with a different port number 8888
#  sbi:
#    server:
#      - address: 127.0.0.10
#        port: 7777
#        advertise: open5gs-nrf.svc.local:8888
#
#########################################################################################
# HTTPS scheme with TLS
#########################################################################################
#  o Set as default if not individually set
#  default:
#    tls:
#      server:
#        scheme: https
#        private_key: /opt/LTE/open5gs/install/etc/open5gs/tls/nrf.key
#        cert: /opt/LTE/open5gs/install/etc/open5gs/tls/nrf.crt
#      client:
#        scheme: https
#        cacert: /opt/LTE/open5gs/install/etc/open5gs/tls/ca.crt
#  sbi:
#    server:
#      - address: nrf.localdomain
#
#  o Enable SSL key logging for Wireshark
#    - This configuration allows capturing SSL/TLS session keys
#      for debugging or analysis purposes using Wireshark.
```

```
#  default:
#    tls:
#      server:
#        scheme: https
#        private_key: /opt/LTE/open5gs/install/etc/open5gs/tls/nrf.key
#        cert: /opt/LTE/open5gs/install/etc/open5gs/tls/nrf.crt
#        sslkeylogfile: /opt/LTE/open5gs/install/var/log/open5gs/tls/nrf-server-sslkeylog.log
#      client:
#        scheme: https
#        cacert: /opt/LTE/open5gs/install/etc/open5gs/tls/ca.crt
#        client_sslkeylogfile: /opt/LTE/open5gs/install/var/log/open5gs/tls/nrf-client-sslkeylog.log
#  sbi:
#    server:
#      - address: nrf.localdomain
#
#  o Add client TLS verification
#  default:
#    tls:
#      server:
#        scheme: https
#        private_key: /opt/LTE/open5gs/install/etc/open5gs/tls/nrf.key
#        cert: /opt/LTE/open5gs/install/etc/open5gs/tls/nrf.crt
#        verify_client: true
#        verify_client_cacert: /opt/LTE/open5gs/install/etc/open5gs/tls/ca.crt
#      client:
#        scheme: https
#        cacert: /opt/LTE/open5gs/install/etc/open5gs/tls/ca.crt
#        client_private_key: /opt/LTE/open5gs/install/etc/open5gs/tls/nrf.key
#        client_cert: /opt/LTE/open5gs/install/etc/open5gs/tls/nrf.crt
#  sbi:
#    server:
#      - address: nrf.localdomain
```

3.92 /opt/GSM/osmo-operator/configs/open5gs/pcrf.yaml

4873 octets, 134 lignes → 134 lignes

```
db_uri: mongodb://localhost/open5gs
logger:
  file:
    path: /opt/LTE/open5gs/install/var/log/open5gs/pcrf.log
#  level: info   # fatal|error|warn|info(default)|debug|trace

global:
  max:
    ue: 1024  # The number of UE can be increased depending on memory size.
#    peer: 64
pcrf:
  freeDiameter: /opt/LTE/open5gs/install/etc/freeDiameter/pcrf.conf
  metrics:
    server:
      - address: 127.0.0.9
        port: 9090

###############################################################################
# PCRF Policy Configuration: SUPI Range Based Policies
###############################################################################
#
# This configuration applies policies based solely on the UE's SUPI range.
#
# supi_range: Specifies one or more ranges of SUPIs.
# session:   Defines the session configuration for each DNN.
#
# Example:
```

```
#
#  policy:
#    - supi_range:          # Filter policies by SUPI
#        - 999700000000001-999709999999999
#        - 315010000000001-315010999999999
#      session:              # Define session based on DNN
#        - name: internet
#          type: 3         # 1: IPv4, 2: IPv6, 3: IPv4v6
#          ambr:
#            downlink:
#              value: 1
#              unit: 3     # 0: bps, 1: Kbps, 2: Mbps, 3: Gbps, 4: Tbps
#            uplink:
#              value: 1
#              unit: 3
#          qos:
#            index: 9        # Allowed values: 1,2,3,4,65,66,67,75,71,72,
#                             # 73,74,76,5,6,7,8,9,69,70,79,80,82,83,
#                             # 84,85,86
#            arp:
#              priority_level: 8   # Allowed values: 1 to 15
#              pre_emption_vulnerability: 1  # 1: Disabled, 2: Enabled
#              pre_emption_capability: 1     # 1: Disabled, 2: Enabled
#        - name: ims
#          type: 3         # 1: IPv4, 2: IPv6, 3: IPv4v6
#          ambr:
#            downlink:
#              value: 1
#              unit: 3
#            uplink:
#              value: 1
#              unit: 3
#          qos:
#            index: 5        # Allowed values as above
#            arp:
#              priority_level: 1   # Allowed values: 1 to 15
#              pre_emption_vulnerability: 1  # 1: Disabled, 2: Enabled
#              pre_emption_capability: 1     # 1: Disabled, 2: Enabled
#          pcc_rule:
#            - qos:
#                index: 1    # Allowed values as above
#                arp:
#                  priority_level: 1
#                  pre_emption_vulnerability: 1
#                  pre_emption_capability: 1
#                mbr:
#                  downlink:
#                    value: 82
#                    unit: 1   # 0: bps, 1: Kbps, 2: Mbps, 3: Gbps, 4: Tbps
#                  uplink:
#                    value: 82
#                    unit: 1
#                gbr:
#                  downlink:
#                    value: 82
#                    unit: 1
#                  uplink:
#                    value: 82
#                    unit: 1
#              flow:
#                - direction: 2
#                  description: "permit out icmp from any to assigned"
#                - direction: 1
#                  description: "permit out icmp from any to assigned"
#                - direction: 2
#                  description: "permit out udp from 10.200.136.98/32 23455 to assigned 1-65535"
```

```
#                - direction: 1
#                  description: "permit out udp from 10.200.136.98/32 1-65535 to assigned 50021"
#            - qos:
#                index: 2    # Allowed values as above
#                arp:
#                  priority_level: 4   # Allowed values: 1 to 15
#                  pre_emption_vulnerability: 2  # 1: Disabled, 2: Enabled
#                  pre_emption_capability: 2     # 1: Disabled, 2: Enabled
#                mbr:
#                  downlink:
#                    value: 802
#                    unit: 1
#                  uplink:
#                    value: 802
#                    unit: 1
#                gbr:
#                  downlink:
#                    value: 802
#                    unit: 1
#                  uplink:
#                    value: 802
#                    unit: 1
#
#    - supi_range:         # Filter policies by SUPI only
#        - 001010000000001-001019999999999
#      session:              # Define session based on DNN
#        - name: internet
#          type: 3         # 1: IPv4, 2: IPv6, 3: IPv4v6
#          ambr:
#            downlink:
#              value: 1
#              unit: 3
#            uplink:
#              value: 1
#              unit: 3
#          qos:
#            index: 9        # Allowed values as above
#            arp:
#              priority_level: 8   # Allowed values: 1 to 15
#              pre_emption_vulnerability: 1  # 1: Disabled, 2: Enabled
#              pre_emption_capability: 1     # 1: Disabled, 2: Enabled
```

3.93 /opt/GSM/osmo-operator/configs/open5gs/sgwc.yaml

1947 octets, 70 lignes → 70 lignes

```
logger:
  file:
    path: /opt/LTE/open5gs/install/var/log/open5gs/sgwc.log
#  level: info   # fatal|error|warn|info(default)|debug|trace

global:
  max:
    ue: 1024  # The number of UE can be increased depending on memory size.
#    peer: 64

sgwc:
  gtpc:
    server:
      - address: 127.0.0.3
  pfcp:
    server:
      - address: 127.0.0.3
    client:
```

```
      sgwu:
        - address: 127.0.0.6

################################################################################
# GTP-C Server
################################################################################
#  o Listen on IPv4 and IPv6
#  gtpc:
#    server:
#      - address: 127.0.0.3
#      - address: fd69:f21d:873c:fa::2
#
################################################################################
# PFCP Server
################################################################################
#  o Override PFCP address to be advertised to SGW-U in PFCP association
#  pfcp:
#    server:
#      - dev: eth0
#        advertise: open5gs-sgwc.svc.local
#
################################################################################
# PFCP Client
################################################################################
#  o SGWU selection by eNodeB TAC
#   (either single TAC or multiple TACs, DECIMAL representation)
#  pfcp:
#    client:
#      sgwu:
#        - address: 127.0.0.6
#          tac: 1
#        - address: 127.0.0.12
#          tac: [3,5,8]
#
#  o SGWU selection by UE's APN (either single APN or multiple APNs)
#  pfcp:
#    client:
#      sgwu:
#        - address: 127.0.0.6
#          apn: ims
#        - address: 127.0.0.12
#          apn: [internet, web]
#
#  o SGWU selection by CellID(e_cell_id: 28bit)
#   (either single e_cell_id or multiple e_cell_id, HEX representation)
#  pfcp:
#    client:
#      sgwu:
#        - address: 127.0.0.6
#          e_cell_id: 463
#        - address: 127.0.0.12
#          e_cell_id: [123456789, 9413]
```

3.94 /opt/GSM/osmo-operator/configs/open5gs/sgwu.yaml

1505 octets, 52 lignes → 52 lignes

```
logger:
  file:
    path: /opt/LTE/open5gs/install/var/log/open5gs/sgwu.log
#  level: info   # fatal|error|warn|info(default)|debug|trace

global:
  max:
```

```
    ue: 1024  # The number of UE can be increased depending on memory size.
#    peer: 64

sgwu:
  pfcp:
    server:
      - address: 127.0.0.6
    client:
#      sgwc:    # SGW-U PFCP Client try to associate SGW-C PFCP Server
#        - address: 127.0.0.3
  gtpu:
    server:
      - address: 127.0.0.6

######################################################################################
# PFCP Server
######################################################################################
#  o Override PFCP address to be advertised to SGW-C in PFCP association
#  pfcp:
#    server:
#      - dev: eth0
#        advertise: open5gs-sgwu.svc.local
#
######################################################################################
# GTP-U Server
######################################################################################
#  o Override SGW-U GTP-U address to be advertised inside S1AP messages
#  gtpu:
#    server:
#      - dev: ens3
#        advertise: sgw1.epc.mnc001.mcc001.3gppnetwork.org
#
#  o User Plane IP Resource information
#  gtpu:
#    server:
#      - address: 127.0.0.6
#        teid_range_indication: 4
#        teid_range: 10
#        network_instance: internet
#        source_interface: 0
#      - address: 127.0.10.4
#        teid_range_indication: 4
#        teid_range: 5
#        network_instance: ims
#        source_interface: 1
```

3.95 /opt/GSM/osmo-operator/configs/open5gs/smf.yaml

11664 octets, 423 lignes → 422 lignes (1 groupes compactés)

```
logger:
  file:
    path: /opt/LTE/open5gs/install/var/log/open5gs/smf.log
#  level: info   # fatal|error|warn|info(default)|debug|trace

global:
  max:
    ue: 1024  # The number of UE can be increased depending on memory size.
#    peer: 64

smf:
  sbi:
    server:
      - address: 127.0.0.4
```

```
        port: 7777
    client:
      nrf:
        - uri: http://127.0.0.10:7777
#       scp:
#         - uri: http://127.0.0.200:7777
  pfcp:
    server:
      - address: 127.0.0.4
    client:
      upf:
        - address: 127.0.0.7
  gtpc:
    server:
      - address: 127.0.0.4
  gtpu:
    server:
      - address: 127.0.0.4
  metrics:
    server:
      - address: 127.0.0.4
        port: 9090
  session:
    - subnet: 10.45.0.0/16
      gateway: 10.45.0.1
    - subnet: 2001:db8:cafe::/48
      gateway: 2001:db8:cafe::1
  dns:
    - 8.8.8.8
    - 8.8.4.4
    - 2001:4860:4860::8888
    - 2001:4860:4860::8844
  mtu: 1400
#  p-cscf:
#    - 127.0.0.1
#    - ::1
#  ctf:
#    enabled: auto   # auto(default)|yes|no
  freeDiameter: /opt/LTE/open5gs/install/etc/freeDiameter/smf.conf

################################################################################
# SMF Info
################################################################################
#  <SMF Selection - 5G Core only>
#  1. SMF sends SmfInfo(S-NSSAI, DNN, TAI) to the NRF
#  2. NRF responds to AMF with SmfInfo during NF-Discovery.
#  3. AMF selects SMF based on S-NSSAI, DNN and TAI in SmfInfo.
#
#  Note that if there is no SmfInfo, any AMF can select this SMF.
#
#  o S-NSSAI[SST:1] and DNN[internet] - At least 1 DNN is required in S-NSSAI
#  info:
#    - s_nssai:
#        - sst: 1
#          dnn:
#            - internet
#
#  o S-NSSAI[SST:1 SD:009000] and DNN[internet or ims]
#  info:
#    - s_nssai:
#        - sst: 1
#          sd: 009000
#          dnn:
#            - internet
#            - ims
#
```

```
#  o S-NSSAI[SST:1] and DNN[internet] and TAI[PLMN-ID:99970 TAC:1]
#  info:
#    - s_nssai:
#        - sst: 1
#          dnn:
#            - internet
#      tai:
#        - plmn_id:
#            mcc: 999
#            mnc: 70
#          tac: 1
#
#  o If any of conditions below are met:
#   - S-NSSAI[SST:1] and DNN[internet] and TAI[PLMN-ID:99970 TAC:1-9]
#   - S-NSSAI[SST:2 SD:000080] and DNN[internet or ims]
#   - S-NSSAI[SST:4] and DNN[internet] and TAI[PLMN-ID:99970 TAC:10-20,30-40]
#  info:
#    - s_nssai:
#        - sst: 1
#          dnn:
#            - internet
#      tai:
#        - plmn_id:
#            mcc: 999
#            mnc: 70
#          tac:
#            - 1-9
#    - s_nssai:
#        - sst: 2
#          sd: 000080
#          dnn:
#            - internet
#            - ims
#    - s_nssai:
#        - sst: 4
#          dnn:
#            - internet
#      tai:
#        - plmn_id:
#            mcc: 999
#            mnc: 70
#          tac:
#            - 10-20
#            - 30-40
#
#  o Complex Example
#  info:
#    - s_nssai:
#        - sst: 1
#          dnn:
#            - internet
#        - sst: 1
#          sd: 000080
#          dnn:
#            - internet
#            - ims
#        - sst: 1
#          sd: 009000
#          dnn:
#            [internet, ims]
#        - sst: 2
#          dnn:
#            - internet
#        - sst: 3
#          sd: 123456
#          dnn:
```

```
#            - internet
#      tai:
#        - plmn_id:
#            mcc: 999
#            mnc: 70
#          tac: [1, 2, 3]
#        - plmn_id:
#            mcc: 999
#            mnc: 70
#          tac: 4
#        - plmn_id:
#            mcc: 999
#            mnc: 70
#          tac:
#            - 5
#            - 6
#        - plmn_id:
#            mcc: 999
#            mnc: 70
#          tac:
#            - 100-200
#            - 300-400
#        - plmn_id:
#            mcc: 999
#            mnc: 70
#          tac:
#            - ⟨1⟩-⟨2⟩  ×3
    ⟨⟩ = (500,600) (700,800) (900,1000)
#    - s_nssai:
#        - sst: 4
#          dnn:
#            - internet
#      tai:
#        - plmn_id:
#            mcc: 999
#            mnc: 70
#          tac: 99
#
################################################################################
# SBI Server
################################################################################
#  o Bind to the address on the eth0 and advertise as open5gs-smf.svc.local
#  sbi:
#    server:
#      - dev:eth0
#        advertise: open5gs-smf.svc.local
#
#  o Specify a custom port number 7777 while binding to the given address
#  sbi:
#    server:
#      - address: smf.localdomain
#        port: 7777
#
#  o Bind to 127.0.0.4 and advertise as open5gs-smf.svc.local
#  sbi:
#    server:
#      - address: 127.0.0.4
#        port: 7777
#        advertise: open5gs-smf.svc.local
#
#  o Bind to port 7777 but advertise with a different port number 8888
#  sbi:
#    server:
#      - address: 127.0.0.4
#        port: 7777
#        advertise: open5gs-smf.svc.local:8888
```

```
#
################################################################################
# SBI Client
################################################################################
#  o Direct Communication with NRF
#  sbi:
#    client:
#      nrf:
#        - uri: http://127.0.0.10:7777
#
#  o Indirect Communication by Delegating to SCP
#  sbi:
#    client:
#      scp:
#        - uri: http://127.0.0.200:7777
#
#  o Indirect Communication without Delegation
#  sbi:
#    client:
#      nrf:
#        - uri: http://127.0.0.10:7777
#      scp:
#        - uri: http://127.0.0.200:7777
#      delegated:
#        nrf:
#          nfm: no    # Directly communicate NRF management functions
#          disc: no   # Directly communicate NRF discovery
#        scp:
#          next: no   # Do not delegate to SCP for next-hop
#
#  o Indirect Communication with Delegated Discovery
#  sbi:
#    client:
#      nrf:
#        - uri: http://127.0.0.10:7777
#      scp:
#        - uri: http://127.0.0.200:7777
#      delegated:
#        nrf:
#          nfm: no    # Directly communicate NRF management functions
#          disc: yes  # Delegate discovery to SCP
#        scp:
#          next: yes  # Delegate to SCP for next-hop communications
#
#  o Default delegation: all communications are delegated to the SCP
#  sbi:
#    client:
#      nrf:
#        - uri: http://127.0.0.10:7777
#      scp:
#        - uri: http://127.0.0.200:7777
#      # No 'delegated' section; defaults to AUTO delegation
#
################################################################################
# HTTPS scheme with TLS
################################################################################
#  o Set as default if not individually set
#  default:
#    tls:
#      server:
#        scheme: https
#        private_key: /opt/LTE/open5gs/install/etc/open5gs/tls/smf.key
#        cert: /opt/LTE/open5gs/install/etc/open5gs/tls/smf.crt
#      client:
#        scheme: https
#        cacert: /opt/LTE/open5gs/install/etc/open5gs/tls/ca.crt
```

```
#  sbi:
#    server:
#      - address: smf.localdomain
#    client:
#      nrf:
#        - uri: https://nrf.localdomain
#
#  o Enable SSL key logging for Wireshark
#    - This configuration allows capturing SSL/TLS session keys
#      for debugging or analysis purposes using Wireshark.
#  default:
#    tls:
#      server:
#        scheme: https
#        private_key: /opt/LTE/open5gs/install/etc/open5gs/tls/smf.key
#        cert: /opt/LTE/open5gs/install/etc/open5gs/tls/smf.crt
#        sslkeylogfile: /opt/LTE/open5gs/install/var/log/open5gs/tls/smf-server-sslkeylog.log
#      client:
#        scheme: https
#        cacert: /opt/LTE/open5gs/install/etc/open5gs/tls/ca.crt
#        client_sslkeylogfile: /opt/LTE/open5gs/install/var/log/open5gs/tls/smf-client-sslkeylog.log
#  sbi:
#    server:
#      - address: smf.localdomain
#    client:
#      nrf:
#        - uri: https://nrf.localdomain
#
#  o Add client TLS verification
#  default:
#    tls:
#      server:
#        scheme: https
#        private_key: /opt/LTE/open5gs/install/etc/open5gs/tls/smf.key
#        cert: /opt/LTE/open5gs/install/etc/open5gs/tls/smf.crt
#        verify_client: true
#        verify_client_cacert: /opt/LTE/open5gs/install/etc/open5gs/tls/ca.crt
#      client:
#        scheme: https
#        cacert: /opt/LTE/open5gs/install/etc/open5gs/tls/ca.crt
#        client_private_key: /opt/LTE/open5gs/install/etc/open5gs/tls/smf.key
#        client_cert: /opt/LTE/open5gs/install/etc/open5gs/tls/smf.crt
#  sbi:
#    server:
#      - address: smf.localdomain
#    client:
#      nrf:
#        - uri: https://nrf.localdomain
#
###########################################################################
# PFCP Server
###########################################################################
#  o Override PFCP address to be advertised to UPF in PFCP association
#  pfcp:
#    server:
#      - dev: eth0
#        advertise: open5gs-smf.svc.local
#
###########################################################################
# PFCP Client
###########################################################################
#  o UPF selection by eNodeB TAC
#   (either single TAC or multiple TACs, DECIMAL representation)
#  pfcp:
#    client:
#      upf:
```

```
#        - address: 127.0.0.7
#          tac: 1
#        - address: 127.0.0.12
#          tac: [3,5,8]
#
#  o UPF selection by UE's DNN/APN (either single DNN/APN or multiple DNNs/APNs)
#  pfcp:
#    client:
#      upf:
#        - address: 127.0.0.7
#          dnn: ims
#        - address: 127.0.0.12
#          dnn: [internet, web]
#
#  o UPF selection by CellID(e_cell_id: 28bit, nr_cell_id: 36bit)
#    (either single enb_id or multiple enb_ids, HEX representation)
#  pfcp:
#    client:
#      upf:
#        - address: 127.0.0.7
#          e_cell_id: 463
#        - address: 127.0.0.12
#          nr_cell_id: [123456789, 9413]
#
##########################################################################################
# GTP-C Server
##########################################################################################
#  o Listen on IPv4 and IPv6
#  gtpc:
#    server:
#      - address: 127.0.0.4
#      - address: fd69:f21d:873c:fa::3
#
##########################################################################################
# GTP-U Server
##########################################################################################
#  o Listen on IPv4 and IPv6
#  gtpu:
#    server:
#      - address: 127.0.0.4
#      - address: ::1
#
##########################################################################################
# 3GPP Specification
##########################################################################################
#  o Specific DNN/APN(e.g 'ims') uses 10.46.0.1/16, 2001:db8:babe::1/48
#   (If the UE has unknown DNN/APN(not internet/ims), SMF/UPF will crash.)
#  session:
#    - subnet: 10.45.0.0/16
#      gateway: 10.45.0.1
#      dnn: internet
#    - subnet: 2001:db8:cafe::/48
#      dnn: internet
#    - subnet: 10.46.0.0/16
#      gateway: 10.46.0.1
#      dnn: ims
#    - subnet: 2001:db8:babe::/48
#      dnn: ims
#
#  o Pool Range
#  session:
#    - subnet: 10.45.0.0/16
#      gateway: 10.45.0.1
#      range:
#        - 10.45.0.100-10.45.0.200
#        - 10.45.1.100-
```

```
#        - -10.45.0.200
#    - subnet: 2001:db8:cafe::/48
#      range:
#        - 2001:db8:cafe:a0::0-2001:db8:cafe:b0::0
#        - 2001:db8:cafe:c0::0-2001:db8:cafe:d0::0
#
#  o Security Indication(5G Core only)
#  security_indication:
#    integrity_protection_indication: required|preferred|not-needed
#    confidentiality_protection_indication: required|preferred|not-needed
#    maximum_integrity_protected_data_rate_uplink: bitrate64kbs|maximum-UE-rate
#    maximum_integrity_protected_data_rate_downlink: bitrate64kbs|maximum-UE-rate
```

3.96 /opt/GSM/osmo-operator/configs/open5gs/subscribers.json

1423 octets, 79 lignes → 79 lignes


```
[
 {
  "ambr": {
   "downlink": {
    "value": 1,
    "unit": 3
   },
   "uplink": {
    "value": 1,
    "unit": 3
   }
  },
  "schema_version": 1,
  "msisdn": [],
  "imeisv": "3589250042736753",
  "mme_host": "mme.localdomain",
  "mme_realm": "localdomain",
  "purge_flag": false,
  "access_restriction_data": 32,
  "subscriber_status": 0,
  "operator_determined_barring": 0,
  "network_access_mode": 0,
  "subscribed_rau_tau_timer": 12,
  "imsi": "001010001000001",
  "security": {
   "k": "00112233445566778899aabbccdd0101",
   "amf": "8000",
   "op": null,
   "opc": "63BFA50EE6523365FF14C1F45F88737D",
   "sqn": {
    "high": 0,
    "low": 1152,
    "unsigned": false
   }
  },
  "slice": [
   {
    "_id": "6a9f17d9436b8b2e35f39745",
    "sst": 1,
    "default_indicator": true,
    "session": [
     {
      "qos": {
       "arp": {
        "priority_level": 8,
        "pre_emption_capability": 1,
        "pre_emption_vulnerability": 1
```

```
       },
       "index": 9
      },
      "ambr": {
       "downlink": {
        "value": 1,
        "unit": 3
       },
       "uplink": {
        "value": 1,
        "unit": 3
       }
      },
      "_id": "6a9f17d9436b8b2e35f39746",
      "name": "srsapn",
      "type": 1,
      "pcc_rule": [],
      "ue": {
       "ipv4": "10.45.0.2"
      }
     }
    ]
   }
  ],
  "__v": 0,
  "mme_timestamp": {
   "high": 416507,
   "low": -2117099773,
   "unsigned": false
  }
 }
]
```


3.97 /opt/GSM/osmo-operator/configs/open5gs/upf.yaml

2503 octets, 88 lignes → 88 lignes

```
logger:
  file:
    path: /opt/LTE/open5gs/install/var/log/open5gs/upf.log
#  level: info   # fatal|error|warn|info(default)|debug|trace

global:
  max:
    ue: 1024  # The number of UE can be increased depending on memory size.
#    peer: 64

upf:
  pfcp:
    server:
      - address: 127.0.0.7
    client:
#      smf:     #  UPF PFCP Client try to associate SMF PFCP Server
#        - address: 127.0.0.4
  gtpu:
    server:
      - address: 127.0.0.7
  session:
    - subnet: 10.45.0.0/16
      gateway: 10.45.0.1
    - subnet: 2001:db8:cafe::/48
      gateway: 2001:db8:cafe::1
  metrics:
    server:
```

```
      - address: 127.0.0.7
        port: 9090

################################################################################
# PFCP Server
################################################################################
# o Override PFCP address to be advertised to SMF in PFCP association
#  pfcp:
#    server:
#      - dev: eth0
#        advertise: open5gs-upf.svc.local
#
################################################################################
# GTP-U Server
################################################################################
#  o Override SGW-U GTP-U address to be advertised inside S1AP messages
#  gtpu:
#    server:
#      - dev: ens3
#        advertise: upf1.5gc.mnc001.mcc001.3gppnetwork.org
#
#  o User Plane IP Resource information
#  gtpu:
#    server:
#      - address:
#        - 127.0.0.7
#        - ::1
#        teid_range_indication: 4
#        teid_range: 10
#        network_instance: internet
#        source_interface: 0
#      - address: 127.0.10.4
#        teid_range_indication: 4
#        teid_range: 5
#        network_instance: ims
#        source_interface: 1
#
################################################################################
# 3GPP Specification
################################################################################
#
#  o Specific DNN/APN(e.g 'ims') uses 10.46.0.1/16, 2001:db8:babe::1/48
#  $ sudo ip addr add 10.45.0.1/16 dev ogstun
#  $ sudo ip addr add 2001:db8:cafe::1/48 dev ogstun2
#  $ sudo ip addr add 10.46.0.1/16 dev ogstun3
#  $ sudo ip addr add 2001:db8:babe::1/48 dev ogstun3
#
#  session:
#    - subnet: 10.45.0.0/16
#      gateway: 10.45.0.1
#      dnn: internet
#    - subnet: 2001:db8:cafe::/48
#      dnn: internet
#      dev: ogstun2
#    - subnet: 10.46.0.0/16
#      gateway: 10.46.0.1
#      dnn: ims
#      dev: ogstun3
#    - subnet: 2001:db8:babe::/48
#      dnn: ims
#      dev: ogstun3
```

3.98 /opt/GSM/osmo-operator/configs/open5gs/dump/open5gs/accounts.metadata.json

290 octets, 1 lignes → 1 lignes


```
{"indexes":[{"v":{"$numberInt":"2"},"key":{"_id":{"$numberInt":"1"}},"name":"_id_"},{"v":{"$numberInt":"2"},
↪  "key":{"username":{"$numberInt":"1"}},"name":"username_1","background":true,"unique":true}],"uuid":
↪  "735ad18ffaf64fd5bfb7d1ad99635967","collectionName":"accounts","type":"collection"}
```


3.99 /opt/GSM/osmo-operator/configs/open5gs/dump/open5gs/prelude.json

51 octets, 1 lignes → 1 lignes


```
{"ServerVersion":"8.0.29","ToolVersion":"100.18.0"}
```


3.100 /opt/GSM/osmo-operator/configs/open5gs/dump/open5gs/sessions.metadata.json

314 octets, 1 lignes → 1 lignes


```
{"indexes":[{"v":{"$numberInt":"2"},"key":{"_id":{"$numberInt":"1"}},"name":"_id_"},{"v":{"$numberInt":"2"},
↪  "key":{"expires":{"$numberInt":"1"}},"name":"expires_1","background":true,"expireAfterSeconds":{"$numberInt"
↪  :"0"}}],"uuid":"5c280e6a26b144d0a2832f3a97dde0a3","collectionName":"sessions","type":"collection"}
```


3.101 /opt/GSM/osmo-operator/configs/open5gs/dump/open5gs/subscribers.metadata.json

285 octets, 1 lignes → 1 lignes


```
{"indexes":[{"v":{"$numberInt":"2"},"key":{"_id":{"$numberInt":"1"}},"name":"_id_"},{"v":{"$numberInt":"2"},
↪  "key":{"imsi":{"$numberInt":"1"}},"name":"imsi_1","background":true,"unique":true}],"uuid":
↪  "cdee4054c5954e61a45edc7a6c67a5bf","collectionName":"subscribers","type":"collection"}
```


3.102 /opt/GSM/osmo-operator/configs/open5gs/freeDiameter/hss.conf

1275 octets, 20 lignes → 20 lignes

```
# hss.conf - freeDiameter du HSS : S6a depuis le MME (Open5GS v2.8.0, /opt/LTE/open5gs/install).
# Depot osmo-operator, configs/open5gs/freeDiameter ; pose par tools/osmo-lte-install.sh
# --configs dans $PREFIX/etc/freeDiameter. Le pourquoi est en tete de mme.conf.

Identity = "hss.localdomain";
Realm = "localdomain";
ListenOn = "127.0.0.8";
TLS_Cred = "/opt/LTE/open5gs/install/etc/open5gs/tls/hss.crt",
↪  "/opt/LTE/open5gs/install/etc/open5gs/tls/hss.key";
TLS_CA = "/opt/LTE/open5gs/install/etc/open5gs/tls/ca.crt";
NoRelay;

LoadExtension = "/opt/LTE/open5gs/install/lib/x86_64-linux-gnu/freeDiameter/dbg_msg_dumps.fdx" : "0x8888";
LoadExtension = "/opt/LTE/open5gs/install/lib/x86_64-linux-gnu/freeDiameter/dict_rfc5777.fdx";
LoadExtension = "/opt/LTE/open5gs/install/lib/x86_64-linux-gnu/freeDiameter/dict_mip6i.fdx";
LoadExtension = "/opt/LTE/open5gs/install/lib/x86_64-linux-gnu/freeDiameter/dict_nasreq.fdx";
LoadExtension = "/opt/LTE/open5gs/install/lib/x86_64-linux-gnu/freeDiameter/dict_nas_mipv6.fdx";
LoadExtension = "/opt/LTE/open5gs/install/lib/x86_64-linux-gnu/freeDiameter/dict_dcca.fdx";
LoadExtension = "/opt/LTE/open5gs/install/lib/x86_64-linux-gnu/freeDiameter/dict_dcca_3gpp.fdx";

ConnectPeer = "mme.localdomain" { ConnectTo = "127.0.0.2"; No_TLS; };
```

3.103 /opt/GSM/osmo-operator/configs/open5gs/freeDiameter/mme.conf

2642 octets, 40 lignes → 40 lignes

```
# mme.conf - freeDiameter du MME : S6a vers le HSS (Open5GS v2.8.0, /opt/LTE/open5gs/install).
# Depot osmo-operator, configs/open5gs/freeDiameter ; pose par tools/osmo-lte-install.sh
# --configs dans $PREFIX/etc/freeDiameter. Le pourquoi est en tete de mme.conf.
#
# [2026-09-09] CE FICHIER MANQUAIT SUR L ISO, ET LE DEMON MOURAIT A L INIT.
# Open5GS pose ses configs (etc/open5gs/*.yaml, etc/freeDiameter/*.conf,
# etc/open5gs/tls/*) par un script d installation meson qui NE FAIT RIEN sous
# DESTDIR (meson.build : install_conf = «... if not os.environ.get("DESTDIR")»).
# Or packaging/osmo-deb.sh pack empaquette par « DESTDIR=… ninja install » :
# le .deb osmo-build-open5gs n a que bin/ et lib/. Les yaml, osmo-lte-install
# les reposait depuis le depot ; freeDiameter, personne - et open5gs-mmed
# s arretait sur « Unable to open configuration file … freeDiameter/mme.conf »
# (mme_fd_init: Assertion `ret == 0'), donc pas de S1AP sur 127.0.0.2, donc
# « Received ConnectionReject » cote srsUE et jamais d attach. Meme sort pour
# le HSS, le PCRF et le SMF - et osmo-epc disait quand meme « demarre ».
# Les quatre .conf sont donc DANS LE DEPOT, avec les certificats TLS generes
# a cote a la pose ($PREFIX/etc/open5gs/tls, openssl).
#
# Les pairs sont en No_TLS (boucle locale : on veut lire le Diameter dans
# Wireshark) ; freeDiameter exige quand meme un TLS_Cred valide a l init.
# Adresses : celles de configs/open5gs/*.yaml (MME 127.0.0.2, HSS 127.0.0.8,
# SMF 127.0.0.4, PCRF 127.0.0.9). Le prefixe est substitue a la pose
# (OPEN5GS_PREFIX), le dossier des extensions aussi (lib/<triplet>/freeDiameter).

Identity = "mme.localdomain";
Realm = "localdomain";
ListenOn = "127.0.0.2";
TLS_Cred = "/opt/LTE/open5gs/install/etc/open5gs/tls/mme.crt",
 ↪  "/opt/LTE/open5gs/install/etc/open5gs/tls/mme.key";
TLS_CA = "/opt/LTE/open5gs/install/etc/open5gs/tls/ca.crt";
NoRelay;

LoadExtension = "/opt/LTE/open5gs/install/lib/x86_64-linux-gnu/freeDiameter/dbg_msg_dumps.fdx" : "0x8888";
LoadExtension = "/opt/LTE/open5gs/install/lib/x86_64-linux-gnu/freeDiameter/dict_rfc5777.fdx";
LoadExtension = "/opt/LTE/open5gs/install/lib/x86_64-linux-gnu/freeDiameter/dict_mip6i.fdx";
LoadExtension = "/opt/LTE/open5gs/install/lib/x86_64-linux-gnu/freeDiameter/dict_nasreq.fdx";
LoadExtension = "/opt/LTE/open5gs/install/lib/x86_64-linux-gnu/freeDiameter/dict_nas_mipv6.fdx";
LoadExtension = "/opt/LTE/open5gs/install/lib/x86_64-linux-gnu/freeDiameter/dict_dcca.fdx";
LoadExtension = "/opt/LTE/open5gs/install/lib/x86_64-linux-gnu/freeDiameter/dict_dcca_3gpp.fdx";

ConnectPeer = "hss.localdomain" { ConnectTo = "127.0.0.8"; No_TLS; };
```

3.104 /opt/GSM/osmo-operator/configs/open5gs/freeDiameter/pcrf.conf

1279 octets, 20 lignes → 20 lignes

```
# pcrf.conf - freeDiameter du PCRF : Gx depuis le SMF (Open5GS v2.8.0, /opt/LTE/open5gs/install).
# Depot osmo-operator, configs/open5gs/freeDiameter ; pose par tools/osmo-lte-install.sh
# --configs dans $PREFIX/etc/freeDiameter. Le pourquoi est en tete de mme.conf.

Identity = "pcrf.localdomain";
Realm = "localdomain";
ListenOn = "127.0.0.9";
TLS_Cred = "/opt/LTE/open5gs/install/etc/open5gs/tls/pcrf.crt",
 ↪  "/opt/LTE/open5gs/install/etc/open5gs/tls/pcrf.key";
TLS_CA = "/opt/LTE/open5gs/install/etc/open5gs/tls/ca.crt";
NoRelay;

LoadExtension = "/opt/LTE/open5gs/install/lib/x86_64-linux-gnu/freeDiameter/dbg_msg_dumps.fdx" : "0x8888";
```

```
LoadExtension = "/opt/LTE/open5gs/install/lib/x86_64-linux-gnu/freeDiameter/dict_rfc5777.fdx";
LoadExtension = "/opt/LTE/open5gs/install/lib/x86_64-linux-gnu/freeDiameter/dict_mip6i.fdx";
LoadExtension = "/opt/LTE/open5gs/install/lib/x86_64-linux-gnu/freeDiameter/dict_nasreq.fdx";
LoadExtension = "/opt/LTE/open5gs/install/lib/x86_64-linux-gnu/freeDiameter/dict_nas_mipv6.fdx";
LoadExtension = "/opt/LTE/open5gs/install/lib/x86_64-linux-gnu/freeDiameter/dict_dcca.fdx";
LoadExtension = "/opt/LTE/open5gs/install/lib/x86_64-linux-gnu/freeDiameter/dict_dcca_3gpp.fdx";

ConnectPeer = "smf.localdomain" { ConnectTo = "127.0.0.4"; No_TLS; };
```

3.105 /opt/GSM/osmo-operator/configs/open5gs/freeDiameter/smf.conf

1282 octets, 20 lignes → 20 lignes

```
# smf.conf - freeDiameter du SMF (PGW-C) : Gx vers le PCRF (Open5GS v2.8.0, /opt/LTE/open5gs/install).
# Depot osmo-operator, configs/open5gs/freeDiameter ; pose par tools/osmo-lte-install.sh
# --configs dans $PREFIX/etc/freeDiameter. Le pourquoi est en tete de mme.conf.

Identity = "smf.localdomain";
Realm = "localdomain";
ListenOn = "127.0.0.4";
TLS_Cred = "/opt/LTE/open5gs/install/etc/open5gs/tls/smf.crt",
 ↪  "/opt/LTE/open5gs/install/etc/open5gs/tls/smf.key";
TLS_CA = "/opt/LTE/open5gs/install/etc/open5gs/tls/ca.crt";
NoRelay;

LoadExtension = "/opt/LTE/open5gs/install/lib/x86_64-linux-gnu/freeDiameter/dbg_msg_dumps.fdx" : "0x8888";
LoadExtension = "/opt/LTE/open5gs/install/lib/x86_64-linux-gnu/freeDiameter/dict_rfc5777.fdx";
LoadExtension = "/opt/LTE/open5gs/install/lib/x86_64-linux-gnu/freeDiameter/dict_mip6i.fdx";
LoadExtension = "/opt/LTE/open5gs/install/lib/x86_64-linux-gnu/freeDiameter/dict_nasreq.fdx";
LoadExtension = "/opt/LTE/open5gs/install/lib/x86_64-linux-gnu/freeDiameter/dict_nas_mipv6.fdx";
LoadExtension = "/opt/LTE/open5gs/install/lib/x86_64-linux-gnu/freeDiameter/dict_dcca.fdx";
LoadExtension = "/opt/LTE/open5gs/install/lib/x86_64-linux-gnu/freeDiameter/dict_dcca_3gpp.fdx";

ConnectPeer = "pcrf.localdomain" { ConnectTo = "127.0.0.9"; No_TLS; };
```

3.106 /opt/GSM/osmo-operator/configs/plymouth/osmo-bts/osmo-bts.plymouth

477 octets, 12 lignes → 12 lignes

```
# osmo-bts - le theme Plymouth du banc.
# Les images sont GENEREES par tools/plymouth-render.py au build de l ISO
# (iso_modules/86-finitions.sh) : ce depot ne porte que le dessin, en Python,
# et le script d animation ci-contre.
[Plymouth Theme]
Name=osmo-operator (antenne)
Description=Une antenne et des cercles qui s ouvrent, en 2D pure
ModuleName=script

[script]
ImageDir=/usr/share/plymouth/themes/osmo-bts
ScriptFile=/usr/share/plymouth/themes/osmo-bts/osmo-bts.script
```

3.107 /opt/GSM/osmo-operator/configs/plymouth/osmo-bts/osmo-bts.script

7506 octets, 175 lignes → 175 lignes

```
# osmo-bts.script - l animation du demarrage.
#
# Plymouth appelle Plymouth.SetRefreshFunction ~50 fois par seconde. Tout ce
# qu on fait ici, c est avancer un compteur et changer l image de deux sprites :
# la bande des ondes (24 frames : les anneaux qui s ouvrent depuis le pylone
# et depuis le mobile - juste les ronds, rien d autre) et la pastille de la
# barre d avancement. Le reste - pylone, mobile, titre - est PEINT DANS LE FOND
# (background.png), une seule image : un initrd n a pas a porter 24 fois le
# meme decor. Apercu anime : wiki/plymouth-osmo-bts.gif (plymouth-render.py
# --gif).
#
# Les coordonnees sont celles d un ecran 1920x1080 ramenees a l ecran reel :
# Window.GetWidth()/GetHeight() donnent la taille vraie, et tout est place en
# FRACTION - sur un 1366x768 comme sur un 4K, la rafale reste entre les deux
# antennes. CE FICHIER EST UN GABARIT : les @@...@@ sont remplaces par
# tools/plymouth-render.py, qui est l auteur du decor et connait sa geometrie.

sw = Window.GetWidth();
sh = Window.GetHeight();

# ── LE DECOR ──────────────────────────────────────────────────────────
bg_image = Image("background.png");
bg = Sprite(bg_image.Scale(sw, sh));
bg.SetX(0);
bg.SetY(0);
bg.SetZ(-100);

# ── LES ONDES ENTRE LES DEUX ANTENNES ─────────────────────────────────
# 24 images, jouees a ~12 images/seconde (une frame toutes les 4 rafraichis).
# Les fichiers gardent leur nom historique (burst-NN.png).
n_frames = 24;
for (i = 0; i < n_frames; i++) {
    if (i < 10) {
        burst_image[i] = Image("burst-0" + i + ".png");
    } else {
        burst_image[i] = Image("burst-" + i + ".png");
    }
    burst_scaled[i] = burst_image[i].Scale(sw * @@BW@@, sh * @@BH@@);
}
# @@BX@@ / @@BY@@ / @@BW@@ / @@BH@@ : la bande entre les deux antennes, en
# FRACTIONS de l ecran, injectees par tools/plymouth-render.py - c est lui qui
# a place le pylone et le mobile dans background.png, donc lui seul sait ou
# tombe l espace entre les deux. Un gabarit plutot que des nombres recopies :
# on deplace le pylone dans le dessin, les ondes suivent.
burst = Sprite();
burst.SetX(sw * @@BX@@);
burst.SetY(sh * @@BY@@);
burst.SetZ(10);

# ── LA BARRE D AVANCEMENT ─────────────────────────────────────────────
box_image = Image("progress-box.png");
bar_image = Image("progress-bar.png");
box_w = sw * 0.27;
box_h = sh * 0.008;
box = Sprite(box_image.Scale(box_w, box_h));
box.SetX((sw - box_w) / 2);
box.SetY(sh * 0.90);
box.SetZ(10);
bar_w = box_w * 0.16;
bar = Sprite(bar_image.Scale(bar_w, box_h));
bar.SetY(sh * 0.90);
bar.SetZ(11);

tick = 0;

# ── LA SAISIE : LA PHRASE DE PASSE LUKS DE /home ──────────────────────
```

```
# [2026-10-01] Une seule invite, « decrypt /home -passphrase- or Enter » : la
# phrase ouvre /home, Entree a vide demarre en root sans /home. Avant : un
# champ vide et des pastilles rondes, sans rien qui dise « j attends que tu
# tapes ».
#
# CE CHAMP N EST PLUS LE CHEMIN PAR DEFAUT. unlock-home pose sa question sur
# la console texte, apres avoir quitte plymouth : sur un portable Intel +
# NVIDIA, le clavier de plymouth etait muet (curseur qui clignote, aucune
# touche prise, delai qui expire). Il ne sert que si /etc/default/crypthome
# dit CRYPTHOME_PROMPT=plymouth - alors unlock-home passe par
# systemd-ask-password, dont l agent plymouth appelle ce qui suit.
# Maintenant : un CURSEUR QUI CLIGNOTE dans le champ des que la question est
# posee, et chaque caractere tape s affiche en « * » - la saisie d un terminal,
# celle qu on reconnait, sur l ecran graphique.
#
# Tout est declare ICI, avant refresh_callback : c est lui qui fait clignoter
# le curseur, et il tourne des la premiere image.
entry_image = Image("entry.png");
entry = Sprite();
entry.SetZ(20);
prompt_sprite = Sprite();
prompt_sprite.SetZ(20);
stars_sprite = Sprite();
stars_sprite.SetZ(21);
# Le curseur : le trait bas d un terminal, dans la police de l image (Ubuntu
# est dans l initrd ; a defaut, fontconfig rend la police par defaut).
cursor_image = Image.Text("_", 0.95, 0.96, 0.98, 1, "Ubuntu Mono 16");
cursor_sprite = Sprite(cursor_image);
cursor_sprite.SetZ(22);
cursor_sprite.SetOpacity(0);
saisie_visible = 0;

fun refresh_callback() {
    tick++;
    # Les ondes : une image toutes les 4 passes (~12 i/s).
    frame = Math.Int(tick / 4) % n_frames;
    burst.SetImage(burst_scaled[frame]);
    # La pastille : un va-et-vient regulier dans la barre. Plymouth ne nous dit
    # pas ou en est le boot en mode script sans plugin de progression - une
    # navette honnete vaut mieux qu un pourcentage invente.
    u = (tick % 200) / 200;
    if (u > 0.5) {
        u = 1 - u;
    }
    bar.SetX((sw - box_w) / 2 + u * 2 * (box_w - bar_w));
    # Le curseur : allume une demi-seconde, eteint une demi-seconde (~50 Hz).
    if (saisie_visible == 1) {
        if (Math.Int(tick / 25) % 2 == 0) {
            cursor_sprite.SetOpacity(1);
        } else {
            cursor_sprite.SetOpacity(0);
        }
    } else {
        cursor_sprite.SetOpacity(0);
    }
}
Plymouth.SetRefreshFunction(refresh_callback);

fun display_password_callback(prompt, bullets) {
    ew = sw * 0.32;
    eh = sh * 0.052;
    entry.SetImage(entry_image.Scale(ew, eh));
    entry.SetX((sw - ew) / 2);
    entry.SetY(sh * 0.72);
    entry.SetOpacity(1);
    text = Image.Text(prompt, 0.90, 0.93, 0.95, 1, "Ubuntu 15");
```

```
    prompt_sprite.SetImage(text);
    prompt_sprite.SetX((sw - text.GetWidth()) / 2);
    prompt_sprite.SetY(sh * 0.66);
    prompt_sprite.SetOpacity(1);
    # Les « * » : un par caractere tape, dans le champ, a gauche.
    marge = sh * 0.016;
    stars_w = 0;
    if (bullets > 0) {
        stars = "";
        for (i = 0; i < bullets; i++) {
            stars = stars + "*";
        }
        simg = Image.Text(stars, 0.95, 0.96, 0.98, 1, "Ubuntu Mono 16");
        stars_sprite.SetImage(simg);
        stars_sprite.SetX((sw - ew) / 2 + marge);
        stars_sprite.SetY(sh * 0.72 + (eh - simg.GetHeight()) / 2);
        stars_sprite.SetOpacity(1);
        stars_w = simg.GetWidth();
    } else {
        stars_sprite.SetOpacity(0);
    }
    # Le curseur suit le dernier « * » ; refresh_callback le fait clignoter.
    cursor_sprite.SetX((sw - ew) / 2 + marge + stars_w);
    cursor_sprite.SetY(sh * 0.72 + (eh - cursor_image.GetHeight()) / 2);
    saisie_visible = 1;
}

fun display_normal_callback() {
    entry.SetOpacity(0);
    prompt_sprite.SetOpacity(0);
    stars_sprite.SetOpacity(0);
    cursor_sprite.SetOpacity(0);
    saisie_visible = 0;
}

Plymouth.SetDisplayPasswordFunction(display_password_callback);
Plymouth.SetDisplayNormalFunction(display_normal_callback);

# ── PAS DE MESSAGES ──────────────────────────────────────────────
# [2026-10-01] Les messages que systemd et fsck envoient a plymouth ne sont
# plus affiches : juste les ondes. Ils restent dans le journal (journalctl -b),
# et un demarrage qui a vraiment besoin de parler le fait sur la console texte
# - c est ce que fait unlock-home pour la phrase de passe.
```

3.108 /opt/GSM/osmo-operator/configs/plymouth/vortex-ubuntu/SOURCE

295 octets, 4 lignes → 4 lignes

```
Theme Plymouth « Vortex Ubuntu », repris tel quel (GPL-2, voir LICENSE).
Origine : https://github.com/emanuele-scarsella/vortex-ubuntu-plymouth-theme
Commit   : 3072445ee35f10a0268baa9aaa326c0abd54af3e (2025-06-20)
Installe par iso_modules/86-finitions.sh comme theme de demarrage par defaut.
```

3.109 /opt/GSM/osmo-operator/configs/plymouth/vortex-ubuntu/logo.svg

3272 octets, 90 lignes → 90 lignes

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!-- Created with Inkscape (http://www.inkscape.org/) -->

<svg
   width="376"
```

```
  height="376"
  viewBox="0 0 99.483332 99.483333"
  version="1.1"
  id="svg308"
  inkscape:version="1.2.2 (b0a8486541, 2022-12-01)"
  sodipodi:docname="Ubuntu People.svg"
  inkscape:export-filename="../../../Code/Themes/vortex-ubuntu-plymouth-theme/vortex-ubuntu/logo.png"
  inkscape:export-xdpi="96"
  inkscape:export-ydpi="96"
  xmlns:inkscape="http://www.inkscape.org/namespaces/inkscape"
  xmlns:sodipodi="http://sodipodi.sourceforge.net/DTD/sodipodi-0.dtd"
  xmlns="http://www.w3.org/2000/svg"
  xmlns:svg="http://www.w3.org/2000/svg">
 <sodipodi:namedview
    id="namedview310"
    pagecolor="#ffffff"
    bordercolor="#666666"
    borderopacity="1.0"
    inkscape:showpageshadow="2"
    inkscape:pageopacity="0"
    inkscape:pagecheckerboard="true"
    inkscape:deskcolor="#d1d1d1"
    inkscape:document-units="px"
    showgrid="false"
    inkscape:zoom="0.49219345"
    inkscape:cx="-428.69323"
    inkscape:cy="358.59884"
    inkscape:window-width="1920"
    inkscape:window-height="1011"
    inkscape:window-x="0"
    inkscape:window-y="32"
    inkscape:window-maximized="1"
    inkscape:current-layer="g277" />
 <defs
    id="defs305" />
 <g
    inkscape:label="Layer 1"
    inkscape:groupmode="layer"
    id="layer1">
   <g
     id="g277"
     transform="matrix(0.45734222,0,0,0.45734222,-7.1305763,-69.626942)"
     style="fill:#ebebeb;fill-opacity:1">
    <g
      id="g300">
     <circle
        class="cls-3"
        cx="58.09"
        cy="255.92"
        r="26.299999"
        id="circle44"
        style="fill:#ebebeb;fill-opacity:1" />
     <circle
        class="cls-3"
        cx="167.63"
        cy="198.25"
        r="26.299999"
        id="circle46"
        style="fill:#ebebeb;fill-opacity:1" />
     <path
        class="cls-3"
        d="M 117.41,323.62 C 98.46,319.56 82.63,307.46 73.73,290.3 c -7.01,3.19 -14.9,4.16 -22.49,2.76
↪  10.77,26.45 33.61,45.66 61.65,51.67 6.15,1.32 12.42,1.96 18.68,1.92 -4.83,-6.35 -7.52,-14.01 -7.7,-21.99
↪  -2.17,-0.24 -4.33,-0.59 -6.45,-1.05 z"
        id="path48"
        style="fill:#ebebeb;fill-opacity:1" />
```

```
        <circle
           class="cls-3"
           cx="161.71001"
           cy="323.76001"
           r="26.299999"
           id="circle50"
           style="fill:#ebebeb;fill-opacity:1" />
        <path
           class="cls-3"
           d="m 198.31,314.1 c 8.18,-10.31 13.94,-22.5 16.71,-35.45 4.84,-22.59 0.32,-46.29 -12.4,-65.51
↪  -3.03,7.14 -8.17,13.17 -14.79,17.32 7.1,13.38 9.26,28.82 6.08,43.67 -1.56,7.27 -4.31,14.12 -8.18,20.38
↪  6.19,5.08 10.56,11.91 12.59,19.6 z"
           id="path52"
           style="fill:#ebebeb;fill-opacity:1" />
        <path
           class="cls-3"
           d="m 56.06,218.2 c 0.67,-0.04 1.34,-0.05 2,-0.05 2.66,0 5.31,0.28 7.94,0.85 4.29,0.92 8.32,2.54
↪  12.01,4.84 11.84,-17.03 30.95,-27.25 51.65,-27.63 0.11,-1.99 0.38,-3.97 0.79,-5.92 1.21,-5.63 3.67,-10.9
↪  7.19,-15.41 -33.08,-2.62 -65.22,14.45 -81.59,43.33 z"
           id="path54"
           style="fill:#ebebeb;fill-opacity:1" />
      </g>
    </g>
  </g>
</svg>
```

3.110 /opt/GSM/osmo-operator/configs/plymouth/vortex-ubuntu/vortex-ubuntu.grub

46 octets, 3 lignes → 3 lignes

```
if background_color 00,00,00; then
  clear
fi
```

3.111 /opt/GSM/osmo-operator/configs/plymouth/vortex-ubuntu/vortex-ubuntu.plymouth

269 octets, 9 lignes → 9 lignes

```
[Plymouth Theme]
Name=Vortex Ubuntu
Description=Ubuntu logo with spinning colored stripes
ModuleName=script

[script]
ImageDir=/usr/share/plymouth/themes/vortex-ubuntu
ScriptFile=/usr/share/plymouth/themes/vortex-ubuntu/vortex-ubuntu.script
UseFirmwareBackground=false
```

3.112 /opt/GSM/osmo-operator/configs/plymouth/vortex-ubuntu/vortex-ubuntu.script

5502 octets, 190 lignes → 190 lignes

```
/*
 * Configuration
 */

logo_spin = -0.009;
```

```
/*
 * Screen dimensions
 */

// Screen size
screen.w = Window.GetWidth(0);
screen.h = Window.GetHeight(0);
screen.half.w = screen.w / 2;
screen.half.h = screen.h / 2;

/*
 * Background
 */

Window.SetBackgroundTopColor(0, 0, 0);
Window.SetBackgroundBottomColor(0, 0, 0);
bg = Sprite(Image("bg.png").Scale(screen.w, screen.h));
bg.SetZ(-10);

/*
 * Global state
 */

// Message
message = null;

// Question and password prompts
prompt = null;
response = null;
bullet.image = Image("dot.png").Scale(10, 10);

text_line_height = Image.Text("Dummy text abcxyz").GetHeight();
prompt_y = screen.h * 0.75;
response_y = prompt_y + text_line_height * 2;
message_y = prompt_y + text_line_height * 4;

// Flow
time_counter = 0.0;

// Stop animation flag
stop_animation = 0;

/*
 * Splash image
 */

splash_scale = 0.75;
splash_y_position = 0.4;
if (Plymouth.GetMode() == "shutdown") {
    splash_y_position = 0.5;
}

fun make_splash_sprite_from_image(img) {
    local.splash_sprite.image = img;
    local.splash_sprite.sprite = make_splash_sprite_sprite(local.splash_sprite.image);
    return local.splash_sprite;
}

fun make_splash_sprite_sprite(img) {
    local.sprite = Sprite(local.img);
    local.sprite.SetX(screen.half.w - local.img.GetWidth() / 2);
    local.sprite.SetY((screen.h * splash_y_position) - local.img.GetHeight() / 2);
    return local.sprite;
}

fun make_splash_sprite(fn) {
```

```
    local.img = Image(local.fn);
    local.splash_sprite.image = local.img.Scale(img.GetWidth() * splash_scale, local.img.GetHeight() *
↪  splash_scale);
    local.splash_sprite.sprite = make_splash_sprite_sprite(local.splash_sprite.image);
    return local.splash_sprite;
}

spins = [0.009, 0.007, 0.006, 0.0053, 0.0048, 0.004, 0.0035, 0.003, 0.0025, 0.002];
spin_sprite_count = 10;

logo_image = Image("logo.png").Scale(120 * splash_scale, 120 * splash_scale);
logo_sprite = make_splash_sprite_from_image(logo_image);

splash_static_bg_image = Image("static.png");
splash_static_bg_image = splash_static_bg_image.Scale(splash_static_bg_image.GetWidth() * splash_scale * 0.3,
↪  splash_static_bg_image.GetHeight() * splash_scale * 0.3);
splash_static_bg = make_splash_sprite_from_image(splash_static_bg_image);

for (i = 0; i < spin_sprite_count; i++) {
    splash_sprites[i] = make_splash_sprite((i + 1) + ".png");
}

fun rotate_splash_sprite(splash_sprite, rotation) {
    splash_sprite.sprite.SetImage(splash_sprite.image.Rotate(rotation));
}

fun update_splash_sprites() {
    if (stop_animation == 0) {
      rotate_splash_sprite(logo_sprite, time_counter * logo_spin);
      for (i = 0; i < spin_sprite_count; i++) {
          rotate_splash_sprite(splash_sprites[i], time_counter * spins[i]);
      }
    }
}

/*
 * Callback functions
 */

fun refresh_callback() {
    time_counter++;
    update_splash_sprites();
}
Plymouth.SetRefreshFunction(refresh_callback);

// Progress bar
if (Plymouth.GetMode () == "boot") {
    ipb = Image("pb.png");
    pb = Sprite(ipb);
    pb.SetX(screen.half.w - 50);
    pb.SetY(screen.h * 0.6 - ipb.GetHeight() / 2);

    fun boot_progress_callback(duration, progress) {
         pb.SetImage(ipb.Scale(progress * 100, 3));
    }
    Plymouth.SetBootProgressFunction(boot_progress_callback);
}

fun display_prompt(prompt_text) {
    prompt.image = Image.Text(prompt_text, 1, 1, 1);
    prompt.sprite = Sprite(prompt.image);
    prompt.sprite.SetX(screen.half.w - prompt.image.GetWidth() / 2);
    prompt.sprite.SetY(prompt_y);
}

fun display_question_callback(prompt_text, entry) {
```

```
    stop_animation = 1;
    display_prompt(prompt_text);

    local.entry_config.alpha = 1.0;
    local.entry_config.font = "Sans Bold 12";
    if (entry == "") {
        entry = "Type your response";
        local.entry_config.alpha = 0.5;
        local.entry_config.font = "Sans 12";
    }

    response = null;
    response.image = Image.Text(entry, 1, 1, 1, local.entry_config.alpha, local.entry_config.font);
    response.sprite = Sprite(response.image);
    response.sprite.SetX(screen.half.w - response.image.GetWidth() / 2);
    response.sprite.SetY(response_y);
}
Plymouth.SetDisplayQuestionFunction(display_question_callback);

fun display_password_callback(prompt_text, dot_count) {
    stop_animation = 1;
    display_prompt(prompt_text);

    response = null;
    if (dot_count > 0) {
        dot_start_position = screen.half.w - (dot_count * bullet.image.GetWidth()) / 2;
        for (i = 0; i < dot_count; i++) {
            response[i].sprite = Sprite(bullet.image);
            response[i].sprite.SetX(dot_start_position + i * bullet.image.GetWidth());
            response[i].sprite.SetY(response_y + (text_line_height - bullet.image.GetHeight()) / 2);
        }
    } else {
        response.image = Image.Text("Enter passphrase", 1, 1, 1, 0.5);
        response.sprite = Sprite(response.image);
        response.sprite.SetX(screen.half.w - response.image.GetWidth() / 2);
        response.sprite.SetY(response_y);
    }
}
Plymouth.SetDisplayPasswordFunction(display_password_callback);

fun display_normal_callback() {
    prompt = null;
    message = null;
    response = null;
    stop_animation = 0;
}
Plymouth.SetDisplayNormalFunction(display_normal_callback);

fun message_callback(text) {
    message.image = Image.Text(text, 1, 1, 1, 0.75, "Sans 10");
    message.sprite = Sprite(message.image);
    message.sprite.SetPosition(screen.half.w - message.image.GetWidth() / 2, message_y);
}
Plymouth.SetMessageFunction(message_callback);
```

3.113 /opt/GSM/osmo-operator/configs/pmos/pmbootstrap_v3.cfg

361 octets, 16 lignes → 16 lignes

```
[pmbootstrap]
aports = @APORTS@
build_pkgs_on_install = False
extra_packages = alsa-utils,alsa-ucm-conf,pipewire,pipewire-
 ↪ alsa,wireplumber,socat,minicom,picocom,networkmanager,networkmanager-ppp,ppp
```

```
hostname = pmos-gsm
is_default_channel = False
locale = fr_FR.UTF-8
timezone = Europe/Paris
ui = phosh
device = qemu-amd64
work = @WORK@

[providers]

[mirrors]
```

3.114 /opt/GSM/osmo-operator/configs/screenfetch/osmo-lab.sh

13142 octets, 31 lignes → 31 lignes

```
#!/bin/bash
# configs/screenfetch/osmo-lab.sh - le logo pixel art du banc, pour screenfetch -a.
# GENERE par tools/screenfetch-art.py : c est la qu on retouche le dessin.
# Source par screenfetch (asciiText) : startline, fulloutput (une ligne = un %s
# en queue, 32 colonnes visibles chacune), labelcolor / textcolor.
#
# Les couleurs des INFOS ne se posent pas ici : screenfetch a deja fige ses
# lignes quand il source ce fichier. Elles sont posees par osmo-banner, qui
# filtre la sortie (utilisateur jaune, hote vert ; root rouge, hote jaune).
labelcolor=$'\e[1;33m'
textcolor=$'\e[0m'
startline="0"
logowidth="32"
fulloutput=(
"
 ↪  \e[38;2;16;16;24m▄\e[0m\e[38;2;16;16;24m\e[48;2;16;16;24m▄\e[0m\e[38;2;16;16;24m\e[48;2;200;208;228m▄\e[0m\e[38;2;16;16;24m
 ↪  \e[38;2;16;16;24m▄\e[0m\e[38;2;16;16;24m▄\e[0m        %s"
"
 ↪  \e[38;2;16;16;24m▄\e[0m\e[38;2;16;16;24m\e[48;2;16;16;24m▄\e[0m\e[38;2;200;208;228m\e[48;2;84;96;130m▄\e[0m\e[38;2;84;96;13
 ↪  \e[38;2;16;16;24m▄\e[0m\e[38;2;200;70;60m\e[48;2;216;204;160m▄\e[0m\e[38;2;232;110;100m\e[48;2;200;70;60m▄\e[0m\e[38;2;16;1
 ↪  %s"
"
 ↪  \e[38;2;16;16;24m\e[48;2;16;16;24m▄\e[0m\e[38;2;200;208;228m\e[48;2;16;16;24m▄\e[0m\e[38;2;84;96;130m\e[48;2;84;96;130m▄\e[
 ↪  %s"
"
 ↪  \e[38;2;16;16;24m\e[48;2;16;16;24m▄\e[0m\e[38;2;200;208;228m\e[48;2;138;148;180m▄\e[0m\e[38;2;84;96;130m\e[48;2;84;96;130m▄
 ↪  %s"
"
 ↪  \e[38;2;16;16;24m▄\e[0m\e[38;2;16;16;24m\e[48;2;16;16;24m▄\e[0m\e[38;2;200;208;228m\e[48;2;138;148;180m▄\e[0m\e[38;2;84;96;
 ↪  %s"
"
 ↪  \e[38;2;16;16;24m▄\e[0m\e[38;2;16;16;24m\e[48;2;16;16;24m▄\e[0m\e[38;2;138;148;180m\e[48;2;16;16;24m▄\e[0m\e[38;2;84;96;130
 ↪  \e[38;2;255;176;0m▄\e[0m     %s"
"
 ↪  \e[38;2;16;16;24m▄\e[0m\e[38;2;138;148;180m\e[48;2;16;16;24m▄\e[0m\e[38;2;138;148;180m\e[48;2;16;16;24m▄\e[0m\e[38;2;138;14
 ↪  \e[38;2;255;176;0m▄\e[0m   %s"
"
 ↪  \e[38;2;16;16;24m▄\e[0m\e[38;2;16;16;24m\e[48;2;150;156;176m▄\e[0m\e[38;2;150;156;176m\e[48;2;150;156;176m▄\e[0m\e[38;2;16;
 ↪  \e[38;2;16;16;24m\e[48;2;16;16;24m▄\e[0m   %s"
"
 ↪  \e[38;2;16;16;24m▄\e[0m\e[38;2;16;16;24m▄\e[0m\e[38;2;16;16;24m\e[48;2;16;16;24m▄\e[0m\e[38;2;150;156;176m\e[48;2;150;156;1
 ↪  %s"
"
 ↪  \e[38;2;16;16;24m▄\e[0m\e[38;2;16;16;24m\e[48;2;150;156;176m▄\e[0m\e[38;2;16;16;24m\e[48;2;16;16;24m▄\e[0m\e[38;2;200;70;60
 ↪  \e[38;2;16;16;24m▄\e[0m\e[38;2;16;16;24m▄\e[0m\e[38;2;16;16;24m▄\e[0m\e[38;2;16;16;24m▄\e[0m\e[38;2;16;16;24m\e[48;2;16;16;
 ↪  %s"
```

```
"
 ↪  \e[38;2;16;16;24m\e[48;2;16;16;24m▀\e[0m\e[38;2;150;156;176m\e[48;2;150;156;176m▀\e[0m\e[38;2;150;156;176m\e[48;2;150;156;1
 ↪  \e[38;2;16;16;24m\e[48;2;16;16;24m▀\e[0m\e[38;2;60;66;88m\e[48;2;60;66;88m▀\e[0m\e[38;2;70;220;120m\e[48;2;70;220;120m▀\e[0
 ↪  %s"
"
 ↪  \e[38;2;16;16;24m\e[48;2;16;16;24m▀\e[0m\e[38;2;150;156;176m\e[48;2;150;156;176m▀\e[0m\e[38;2;150;156;176m\e[48;2;150;156;1
 ↪  \e[38;2;16;16;24m\e[48;2;16;16;24m▀\e[0m\e[38;2;60;66;88m\e[48;2;60;66;88m▀\e[0m\e[38;2;60;66;88m\e[48;2;96;104;134m▀\e[0m\
 ↪  %s"
"
 ↪  \e[38;2;16;16;24m\e[48;2;16;16;24m▀\e[0m\e[38;2;150;156;176m\e[48;2;16;16;24m▀\e[0m\e[38;2;150;156;176m\e[48;2;16;16;24m▀\e
 ↪  \e[38;2;16;16;24m\e[48;2;16;16;24m▀\e[0m\e[38;2;60;66;88m\e[48;2;60;66;88m▀\e[0m\e[38;2;96;104;134m\e[48;2;96;104;134m▀\e[0
m▀\e[0m   %s"
"
 ↪  \e[38;2;16;16;24m\e[48;2;16;16;24m▀\e[0m\e[38;2;200;208;228m\e[48;2;140;180;140m▀\e[0m\e[38;2;200;208;228m\e[48;2;140;180;1
 ↪  \e[38;2;16;16;24m\e[48;2;16;16;24m▀\e[0m\e[38;2;60;66;88m\e[48;2;16;16;24m▀\e[0m\e[38;2;60;66;88m\e[48;2;16;16;24m▀\e[0m\e[
]
 ↪  88m\e[48;2;16;16;24m▀\e[0m\e[38;2;60;66;88m\e[48;2;16;16;24m▀\e[0m\e[38;2;60;66;88m\e[48;2;16;16;24m▀\e[0m\e[38;2;16;16;24m
 ↪  %s"
"
 ↪  \e[38;2;16;16;24m\e[48;2;16;16;24m▀\e[0m\e[38;2;90;138;90m\e[48;2;16;16;24m▀\e[0m\e[38;2;90;138;90m\e[48;2;16;16;24m▀\e[0m\
 ↪  %s")
```

3.115 /opt/GSM/osmo-operator/configs/srsran/enb.conf

20734 octets, 481 lignes → 479 lignes (1 groupes compactés)

```
#####################################################################
#                   srsENB configuration file
#####################################################################

#####################################################################
# eNB configuration
#
# enb_id:               20-bit eNB identifier.
# mcc:                  Mobile Country Code
# mnc:                  Mobile Network Code
# mme_addr:             IP address of MME for S1 connnection
# gtp_bind_addr:        Local IP address to bind for GTP connection
# gtp_advertise_addr:   IP address of eNB to advertise for DL GTP-U Traffic
# s1c_bind_addr:        Local IP address to bind for S1AP connection
# s1c_bind_port:        Source port for S1AP connection (0 means any)
# n_prb:                Number of Physical Resource Blocks (6,15,25,50,75,100)
# tm:                   Transmission mode 1-4 (TM1 default)
# nof_ports:            Number of Tx ports (1 port default, set to 2 for TM2/3/4)
#
#####################################################################
[enb]
enb_id = 0x19B
mcc = 001
mnc = 01
mme_addr = 127.0.0.2
gtp_bind_addr = 127.0.1.1
s1c_bind_addr = 127.0.1.1
s1c_bind_port = 0
n_prb = 50
#tm = 4
#nof_ports = 2

#####################################################################
# eNB configuration files
#
# sib_config:  SIB1, SIB2 and SIB3 configuration file
# note: When enabling MBMS, use the sib.conf.mbsfn configuration file which includes SIB13
# rr_config:   *Radio Resources configuration file
```

```
# rb_config:   SRB/DRB configuration file
#####################################################################
[enb_files]
sib_config = sib.conf
rr_config  = rr.conf
rb_config = rb.conf

#####################################################################
# RF configuration
#
# dl_earfcn: EARFCN code for DL (only valid if a single cell is configured in rr.conf)
# tx_gain: Transmit gain (dB).
# rx_gain: Optional receive gain (dB). If disabled, AGC if enabled
#
# Optional parameters:
# dl_freq:            Override DL frequency corresponding to dl_earfcn
# ul_freq:            Override UL frequency corresponding to dl_earfcn (must be set if dl_freq is set)
# device_name:        Device driver family
#                     Supported options: "auto" (uses first driver found), "UHD", "bladeRF", "soapy" or "zmq"
# device_args:        Arguments for the device driver. Options are "auto" or any string.
#                     Default for UHD: "recv_frame_size=9232,send_frame_size=9232"
#                     Default for bladeRF: ""
# time_adv_nsamples:  Transmission time advance (in number of samples) to compensate for RF delay
#                     from antenna to timestamp insertion.
#                     Default "auto". B210 USRP: 100 samples, bladeRF: 27
#####################################################################
[rf]
dl_earfcn = 3350
tx_gain = 80
rx_gain = 40

#device_name = zmq

# For best performance in 2x2 MIMO and >= 15 MHz use the following device_args settings:
#     USRP B210: num_recv_frames=64,num_send_frames=64
#     And for 75 PRBs, also append ",master_clock_rate=15.36e6" to the device args

# For best performance when BW<5 MHz (25 PRB), use the following device_args settings:
#     USRP B210: send_frame_size=512,recv_frame_size=512

#device_args = auto
#time_adv_nsamples = auto

# Example for ZMQ-based operation with TCP transport for I/Q samples
device_name = zmq
device_args = fail_on_disconnect=true,tx_port=tcp://*:2000,rx_port=tcp://localhost:2001,id=enb,base_srate=11.52e6

#####################################################################
# Packet capture configuration
#
# MAC-layer packets are captured to a file in the compact format which can
# be decoded by Wireshark. For decoding, use the UDP dissector and the UDP
# heuristic dissection. Edit the preferences (Edit > Preferences >
# Protocols > DLT_USER) for DLT_USER to add an entry for DLT=149 with
# Protocol=udp. Further, enable the heuristic dissection in UDP under:
# Analyze > Enabled Protocols > MAC-LTE > mac_lte_udp and MAC-NR > mac_nr_udp
# For more information see: https://wiki.wireshark.org/MAC-LTE
# Configuring this Wireshark preferences is needed for decoding the MAC PCAP
# files as well as for the live network capture option.
#
# Please note that this setting will by default only capture MAC
# frames on dedicated channels, and not SIB.  You have to build with
# WRITE_SIB_PCAP enabled in srsenb/src/stack/mac/mac.cc if you want
# SIB to be part of the MAC pcap file.
#
# S1AP Packets are captured to a file in the compact format which can
```

```
# be decoded by the Wireshark s1ap dissector with DLT 150.
# To use the dissector, edit the preferences for DLT_USER to
# add an entry with DLT=150, Payload Protocol=s1ap.
#
# enable:        Enable MAC layer packet captures (true/false)
# filename:      File path to use for LTE MAC packet captures
# nr_filename:   File path to use for NR MAC packet captures
# s1ap_enable:   Enable or disable the PCAP.
# s1ap_filename: File name where to save the PCAP.
#
# mac_net_enable: Enable MAC layer packet captures sent over the network (true/false default: false)
# bind_ip: Bind IP address for MAC network trace (default: "0.0.0.0")
# bind_port: Bind port for MAC network trace (default: 5687)
# client_ip: Client IP address for MAC network trace (default: "127.0.0.1")
# client_port Client IP address for MAC network trace (default: 5847)
############################################################################
[pcap]
#enable = false
#filename = /tmp/enb_mac.pcap
#nr_filename = /tmp/enb_mac_nr.pcap
#s1ap_enable = false
#s1ap_filename = /tmp/enb_s1ap.pcap

# [2026-09-16] LES TRAMES 4G, POUR LE BANC. Chaque PDU MAC (DL et UL, donc
# RRC et NAS dedans : open5gs chiffre en EEA0, tout est lisible) part en UDP
# vers 127.0.0.1:5847 au format « mac-lte-framed » de Wireshark. Le tableau
# de bord (osmo-egprs-web) le capture avec le GSMTAP et le SCTP, et l appli
# Network Monitor du telephone pmOS les affiche. Sans lecteur, l UDP se perd :
# aucun cout.
mac_net_enable = true
bind_ip = 0.0.0.0
bind_port = 5687
client_ip = 127.0.0.1
client_port = 5847
#bind_ip = 0.0.0.0
#bind_port = 5687
#client_ip = 127.0.0.1
#client_port = 5847

############################################################################
# Log configuration
#
# Log levels can be set for individual layers. "all_level" sets log
# level for all layers unless otherwise configured.
# Format: e.g. phy_level = info
#
# In the same way, packet hex dumps can be limited for each level.
# "all_hex_limit" sets the hex limit for all layers unless otherwise
# configured.
# Format: e.g. phy_hex_limit = 32
#
# Logging layers: rf, phy, phy_lib, mac, rlc, pdcp, rrc, gtpu, s1ap, stack, all
# Logging levels: debug, info, warning, error, none
#
# filename: File path to use for log output. Can be set to stdout
#           to print logs to standard output
# file_max_size: Maximum file size (in kilobytes). When passed, multiple files are created.
#                If set to negative, a single log file will be created.
############################################################################
[log]
all_level = warning
all_hex_limit = 32
filename = /tmp/enb.log
file_max_size = -1

[gui]
```

```
enable = false

#####################################################################
# Scheduler configuration options
#
# sched_policy:      User MAC scheduling policy (E.g. time_rr, time_pf)
# min_aggr_level:    Optional minimum aggregation level index (l=log2(L) can be 0, 1, 2 or 3)
# max_aggr_level:    Optional maximum aggregation level index (l=log2(L) can be 0, 1, 2 or 3)
# adaptive_aggr_level: Boolean flag to enable/disable adaptive aggregation level based on target BLER
# pdsch_mcs:         Optional fixed PDSCH MCS (ignores reported CQIs if specified)
# pdsch_max_mcs:     Optional PDSCH MCS limit
# pusch_mcs:         Optional fixed PUSCH MCS (ignores reported CQIs if specified)
# pusch_max_mcs:     Optional PUSCH MCS limit
# min_nof_ctrl_symbols: Minimum number of control symbols
# max_nof_ctrl_symbols: Maximum number of control symbols
# pucch_multiplex_enable: Allow PUCCH HARQ to collide with PUSCH and other PUCCH
# pucch_harq_max_rb: Maximum number of RB to be used for PUCCH on the edges of the grid.
#                    If defined and greater than 0, the scheduler will avoid DL PDCCH allocations if
#                    PUCCH HARQ falls outside this region
# target_bler:       Target BLER (in decimal) to achieve via adaptive link
# max_delta_dl_cqi:  Maximum shift in CQI for adaptive DL link
# max_delta_ul_snr:  Maximum shift in UL SNR for adaptive UL link
# adaptive_dl_mcs_step_size: Step size or learning rate used in adaptive DL MCS link
# adaptive_ul_mcs_step_size: Step size or learning rate used in adaptive UL MCS link
# min_tpc_tti_interval: Minimum TTI interval between TPCs different than 1
# ul_snr_avg_alpha:  Exponential Average alpha coefficient used in estimation of UL SNR
# init_ul_snr_value: Initial UL SNR value used for computing MCS in the first UL grant
# init_dl_cqi:       DL CQI value used before any CQI report is available to the eNB
# max_sib_coderate:  Upper bound on SIB and RAR grants coderate
# pdcch_cqi_offset:  CQI offset in derivation of PDCCH aggregation level
# nr_pdsch_mcs:      Optional fixed NR PDSCH MCS (ignores reported CQIs if specified)
# nr_pusch_mcs:      Optional fixed NR PUSCH MCS (ignores reported CQIs if specified)
#
#####################################################################
[scheduler]
#policy     = time_pf
#policy_args = 2
#min_aggr_level   = 0
#max_aggr_level   = 3
#adaptive_aggr_level = false
#pdsch_mcs        = -1
#pdsch_max_mcs    = -1
#pusch_mcs        = -1
#pusch_max_mcs    = 16
#min_nof_ctrl_symbols = 1
#max_nof_ctrl_symbols = 3
#pucch_multiplex_enable = false
#pucch_harq_max_rb = 0
#target_bler = 0.05
#max_delta_dl_cqi = 5
#max_delta_ul_snr = 5
#adaptive_dl_mcs_step_size = 0.001
#adaptive_ul_mcs_step_size = 0.001
#min_tpc_tti_interval = 1
#ul_snr_avg_alpha=0.05
#init_ul_snr_value=5
#init_dl_cqi=5
#max_sib_coderate=0.3
#pdcch_cqi_offset=0
#nr_pdsch_mcs=28
#nr_pusch_mcs=28

#####################################################################
# Slicing configuration
# enable_eMBB           Enables enhanced mobile broadband (eMBB) slice in the gNodeB
# enable_URLLC          Enables Ultra Reliable Low Latency Communications (URLLC) slice in the gNodeB
```

```
# enable_MIoT           Enables Massive Internet of Things (MIoT) slice in the gNodeB
# eMBB_sd               eMBB slice differentiator
# URLLC_sd              URLLC slice differentiator
# MIoT_sd               MIoT slice differentiator
###################################################################
[slicing]
#enable_eMBB = false
#enable_URLLC = false
#enable_MIoT = false
#eMBB_sd = 1
#URLLC_sd = 1
#MIoT_sd = 1

###################################################################
# eMBMS configuration options
#
# enable:               Enable MBMS transmission in the eNB
# m1u_multiaddr:        Multicast address the M1-U socket will register to
# m1u_if_addr:          Address of the interface the M1-U interface will listen to for multicast packets
# mcs:                  Modulation and Coding scheme for MBMS traffic
#
###################################################################
[embms]
#enable = false
#m1u_multiaddr = 239.255.0.1
#m1u_if_addr = 127.0.1.201
#mcs = 20
  ×3
###################################################################
# Channel emulator options:
# enable:            Enable/disable internal Downlink/Uplink channel emulator
#
# -- AWGN Generator
# awgn.enable:       Enable/disable AWGN generator
# awgn.snr:          Target SNR in dB
#
# -- Fading emulator
# fading.enable:     Enable/disable fading simulator
# fading.model:      Fading model + maximum doppler (E.g. none, epa5, eva70, etu300, etc)
#
# -- Delay Emulator     delay(t) = delay_min + (delay_max - delay_min) * (1 + sin(2pi*t/period)) / 2
#                       Maximum speed [m/s]: (delay_max - delay_min) * pi * 300 / period
# delay.enable:      Enable/disable delay simulator
# delay.period_s:    Delay period in seconds
# delay.init_time_s: Delay initial time in seconds
# delay.maximum_us:  Maximum delay in microseconds
# delay.minumum_us:  Minimum delay in microseconds
#
# -- Radio-Link Failure (RLF) Emulator
# rlf.enable:        Enable/disable RLF simulator
# rlf.t_on_ms:       Time for On state of the channel (ms)
# rlf.t_off_ms:      Time for Off state of the channel (ms)
#
# -- High Speed Train Doppler model simulator
# hst.enable:        Enable/disable HST simulator
# hst.period_s:      HST simulation period in seconds
# hst.fd_hz:         Doppler frequency in Hz
# hst.init_time_s:   Initial time in seconds
###################################################################
[channel.dl]
#enable        = false

[channel.dl.awgn]
#enable        = false
#snr            = 30
```

```
[channel.dl.fading]
#enable        = false
#model         = none

[channel.dl.delay]
#enable        = false
#period_s      = 3600
#init_time_s   = 0
#maximum_us    = 100
#minimum_us    = 10

[channel.dl.rlf]
#enable        = false
#t_on_ms       = 10000
#t_off_ms      = 2000

[channel.dl.hst]
#enable        = false
#period_s      = 7.2
#fd_hz         = 750.0
#init_time_s   = 0.0

[channel.ul]
#enable        = false

[channel.ul.awgn]
#enable        = false
#n0            = -30

[channel.ul.fading]
#enable        = false
#model         = none

[channel.ul.delay]
#enable        = false
#period_s      = 3600
#init_time_s   = 0
#maximum_us    = 100
#minimum_us    = 10

[channel.ul.rlf]
#enable        = false
#t_on_ms       = 10000
#t_off_ms      = 2000

[channel.ul.hst]
#enable        = false
#period_s      = 7.2
#fd_hz         = -750.0
#init_time_s   = 0.0

######################################################################
# CFR configuration options
#
# The CFR module provides crest factor reduction for the transmitted signal.
#
# enable:           Enable or disable the CFR. Default: disabled
#
# mode:             manual:   CFR threshold is set by cfr_manual_thres (default).
#                   auto_ema: CFR threshold is adaptive based on the signal PAPR. Power avg. with Exponential
 ↪  Moving Average.
#                             The time constant of the averaging can be tweaked with the ema_alpha parameter.
#                  auto_cma: CFR threshold is adaptive based on the signal PAPR. Power avg. with Cumulative Moving
 ↪  Average.
#                            Use with care, as CMA's increasingly slow response may be unsuitable for most use cases.
#
```

```
# strength:         Ratio between amplitude-limited vs unprocessed signal (0 to 1). Default: 1
# manual_thres:     Fixed manual clipping threshold for CFR manual mode. Default: 0.5
# auto_target_papr: Signal PAPR target (in dB) in CFR auto modes. output PAPR can be higher due to peak smoothing.
 ↪  Default: 8
# ema_alpha:        Alpha coefficient for the power average in auto_ema mode. Default: 1/7
#
######################################################################
[cfr]
#enable           = false
#mode             = manual
#manual_thres     = 0.5
#strength         = 1
#auto_target_papr = 8
#ema_alpha        = 0.0143

# E2 Agent configuration options
#
# ric_ip:                IP address of the RIC controller
# ric_port:              Port of the RIC controller
# ric_bind_ip:           Local IP address to bind for RIC connection
# ric_bind_port:         Local port to bind for RIC connection
# max_ric_setup_retries: Maximum amount of retries to setup the RIC connection. If this value is exceeded, an
 ↪  alarm is written to the log. -1 means infinity.
# ric_connect_timer:     Connection Retry Timer for RIC connection (seconds)
######################################################################
[e2_agent]
#enable = false
#ric_ip =  127.0.0.1
#ric_port = 36421
#ric_bind_ip =  127.0.0.1
#ric_bind_port = 36425
#max_ric_setup_retries = -1
#ric_connect_timer = 10

######################################################################
# Expert configuration options
#
# pusch_max_its:        Maximum number of turbo decoder iterations (default: 4)
# nr_pusch_max_its:     Maximum number of LDPC iterations for NR (Default 10)
# pusch_8bit_decoder:   Use 8-bit for LLR representation and turbo decoder trellis computation (experimental)
# nof_phy_threads:      Selects the number of PHY threads (maximum: 4, minimum: 1, default: 3)
# metrics_period_secs:  Sets the period at which metrics are requested from the eNB
# metrics_csv_enable:   Write eNB metrics to CSV file.
# metrics_csv_filename: File path to use for CSV metrics
# report_json_enable:   Write eNB report to JSON file (default: disabled)
# report_json_filename: Report JSON filename (default: /tmp/enb_report.json)
# report_json_asn1_oct: Prints ASN1 messages encoded as an octet string instead of plain text in the JSON report
 ↪  file
# alarms_log_enable:    Enable Alarms logging (default: disabled)
# alarms_filename:      Alarms logging filename (default: /tmp/alarms.log)
# tracing_enable:       Write source code tracing information to a file
# tracing_filename:     File path to use for tracing information
# tracing_buffcapacity: Maximum capacity in bytes the tracing framework can store
# stdout_ts_enable:     Prints once per second the timestamp into stdout
# tx_amplitude:         Transmit amplitude factor (set 0-1 to reduce PAPR)
# rrc_inactivity_timer  Inactivity timeout used to remove UE context from RRC (in milliseconds)
# max_mac_dl_kos:       Maximum number of consecutive KOs in DL before triggering the UE's release (default: 100)
# max_mac_ul_kos:       Maximum number of consecutive KOs in UL before triggering the UE's release (default: 100)
# max_prach_offset_us:  Maximum allowed RACH offset (in us)
# nof_prealloc_ues:     Number of UE memory resources to preallocate during eNB initialization for faster UE
 ↪  creation (default: 8)
# rlf_release_timer_ms: Time taken by eNB to release UE context after it detects an RLF
# eea_pref_list:        Ordered preference list for the selection of encryption algorithm (EEA) (default: EEA0,
 ↪  EEA2, EEA1)
# eia_pref_list:        Ordered preference list for the selection of integrity algorithm (EIA) (default: EIA2,
 ↪  EIA1, EIA0)
```

```
# gtpu_tunnel_timeout:  Time that GTPU takes to release indirect forwarding tunnel since the last received GTPU
 ↪  PDU (0 for no timer)
# ts1_reloc_prep_timeout: S1AP TS 36.413 TS1RelocPrep Expiry Timeout value in milliseconds
# ts1_reloc_overall_timeout: S1AP TS 36.413 TS1RelocOverall Expiry Timeout value in milliseconds
# rlf_release_timer_ms: Time taken by eNB to release UE context after it detects a RLF
# rlf_min_ul_snr_estim: SNR threshold in dB below which the enb is notified with RLF ko
# s1_setup_max_retries: Maximum amount of retries to setup the S1AP connection. If this value is exceeded, an
 ↪  alarm is written to the log. -1 means infinity.
# s1_connect_timer:     Connection Retry Timer for S1 connection (seconds)
# rx_gain_offset:       RX Gain offset to add to rx_gain to calibrate RSRP readings
# use_cedron_f_est_alg: Whether to use Cedron algorithm for TA estimation or not (Default: false)
#####################################################################
[expert]
#pusch_max_its        = 8 # These are half iterations
#nr_pusch_max_its     = 10
#pusch_8bit_decoder   = false
#nof_phy_threads      = 3
#metrics_period_secs  = 1
#metrics_csv_enable   = false
#metrics_csv_filename = /tmp/enb_metrics.csv
#report_json_enable   = true
#report_json_filename = /tmp/enb_report.json
#report_json_asn1_oct = false
#alarms_log_enable    = true
#alarms_filename      = /tmp/enb_alarms.log
#tracing_enable       = true
#tracing_filename     = /tmp/enb_tracing.log
#tracing_buffcapacity = 1000000
#stdout_ts_enable     = false
#tx_amplitude         = 0.6
# [2026-09-07] LA 4G QUI NE TIENT PAS. Par defaut srsENB libere le
# contexte RRC apres 30 s SANS DATA : le log EPC montre alors un
# « UE Context Release Request » ~33 s apres l'attach, et le mobile
# se rattache en boucle. Un ping permanent masquait le probleme en
# reamorcant le timer. 2 h : la session tient toute une manip.
rrc_inactivity_timer = 7200000
#max_mac_dl_kos       = 100
#max_mac_ul_kos       = 100
#max_prach_offset_us  = 30
#nof_prealloc_ues     = 8
#rlf_release_timer_ms = 4000
#lcid_padding         = 3
#eea_pref_list = EEA0, EEA2, EEA1
#eia_pref_list = EIA2, EIA1, EIA0
#gtpu_tunnel_timeout = 0
#extended_cp         = false
#ts1_reloc_prep_timeout = 10000
#ts1_reloc_overall_timeout = 10000
#rlf_release_timer_ms = 4000
#rlf_min_ul_snr_estim = -2
#s1_setup_max_retries = -1
#s1_connect_timer = 10
#rx_gain_offset = 62
#mac_prach_bi         = 0
#use_cedron_f_est_alg = false
```

3.116 /opt/GSM/osmo-operator/configs/srsran/rb.conf

3386 octets, 187 lignes → 187 lignes

```
// All times are in ms. Use -1 for infinity, where available

// 4G Section
```

```
// srb1_config = {
//   rlc_config = {
//     ul_am = {
//       t_poll_retx = 45;
//       poll_pdu = -1;
//       poll_byte = -1;
//       max_retx_thresh = 4;
//     };
//     dl_am = {
//       t_reordering = 35;
//       t_status_prohibit = 0;
//     };
//     enb_specific = {
//      dl_max_retx_thresh = 32;
//     };
//   };
// }

// srb2_config = {
//   rlc_config = {
//     ul_am = {
//       t_poll_retx = 45;
//       poll_pdu = -1;
//       poll_byte = -1;
//       max_retx_thresh = 4;
//     };
//     dl_am = {
//       t_reordering = 35;
//       t_status_prohibit = 0;
//     };
//     enb_specific = {
//      dl_max_retx_thresh = 32;
//     };
//   };
// }

qci_config = (
{
  qci = 7;
  pdcp_config = {
    discard_timer = -1;
    pdcp_sn_size = 12;
  }
  rlc_config = {
    ul_um = {
      sn_field_length = 10;
    };
    dl_um = {
      sn_field_length = 10;
      t_reordering    = 45;
    };
  };
  logical_channel_config = {
    priority = 13;
    prioritized_bit_rate   = -1;
    bucket_size_duration  = 100;
    log_chan_group = 2;
  };
  enb_specific = {
    dl_max_retx_thresh = 32;
  };
},
{
  qci = 9;
  pdcp_config = {
    discard_timer = 150;
```

```
    status_report_required = true;
  }
  rlc_config = {
    ul_am = {
      t_poll_retx = 120;
      poll_pdu = 64;
      poll_byte = 750;
      max_retx_thresh = 16;
    };
    dl_am = {
      t_reordering = 50;
      t_status_prohibit = 50;
    };
  };
  logical_channel_config = {
    priority = 11;
    prioritized_bit_rate   = -1;
    bucket_size_duration  = 100;
    log_chan_group = 3;
  };
  enb_specific = {
    dl_max_retx_thresh = 32;
  };
}
);

// 5G Section
srb1_5g_config = {
 rlc_config = {
   ul_am = {
     sn_field_len = 12;
     t_poll_retx = 45;
     poll_pdu = -1;
     poll_byte = -1;
     max_retx_thres = 8;
   };
   dl_am = {
     sn_field_len = 12;
     t_reassembly = 35;
     t_status_prohibit = 10;
   };
 };
}

srb2_5g_config = {
 rlc_config = {
   ul_am = {
     sn_field_len = 12;
     t_poll_retx = 45;
     poll_pdu = -1;
     poll_byte = -1;
     max_retx_thres = 8;
   };
   dl_am = {
     sn_field_len = 12;
     t_reassembly = 35;
     t_status_prohibit = 10;
   };
 };
}

five_qi_config = (
{
  five_qi = 7;
  pdcp_nr_config = {
    drb = {
```

```
      pdcp_sn_size_ul = 18;
      pdcp_sn_size_dl = 18;
      discard_timer = 50;
      integrity_protection = false;
      status_report = false;
    };
    t_reordering = 50;
  };
  rlc_config = {
    um_bi_dir = {
      ul_um = {
        sn_field_len = 12;
      };
      dl_um = {
        sn_field_len = 12;
        t_reassembly = 50;
      };
    };
  };
},
{
  five_qi = 9;
  pdcp_nr_config = {
    drb = {
      pdcp_sn_size_ul = 18;
      pdcp_sn_size_dl = 18;
      discard_timer = 50;
      integrity_protection = false;
      status_report = false;
    };
    t_reordering = 50;
  };
  rlc_config = {
    am = {
      ul_am = {
        sn_field_len = 12;
        t_poll_retx = 50;
        poll_pdu = 4;
        poll_byte = 3000;
        max_retx_thres = 4;
      };
      dl_am = {
        sn_field_len = 12;
        t_reassembly = 50;
        t_status_prohibit = 50;
      };
    };
  };
}
);
```

3.117 /opt/GSM/osmo-operator/configs/srsran/rr.conf

3031 octets, 118 lignes → 118 lignes

```
mac_cnfg =
{
  phr_cnfg =
  {
    dl_pathloss_change = "dB3"; // Valid: 1, 3, 6 or INFINITY
    periodic_phr_timer = 50;
    prohibit_phr_timer = 0;
  };
  ulsch_cnfg =
```

```
  {
    max_harq_tx = 4;
    periodic_bsr_timer = 20; // in ms
    retx_bsr_timer = 320;   // in ms
  };

  time_alignment_timer = -1; // -1 is infinity
};

phy_cnfg =
{
  phich_cnfg =
  {
    duration  = "Normal";
    resources = "1/6";
  };

  pusch_cnfg_ded =
  {
    beta_offset_ack_idx = 6;
    beta_offset_ri_idx  = 6;
    beta_offset_cqi_idx = 6;
  };

  // PUCCH-SR resources are scheduled on time-frequeny domain first, then multiplexed in the same resource.
  sched_request_cnfg =
  {
    dsr_trans_max = 64;
    period = 20;          // in ms
    //subframe = [1, 11]; // Optional vector of subframe indices allowed for SR transmissions (default uses all)
    nof_prb = 1;          // number of PRBs on each extreme used for SR (total prb is twice this number)
  };
  cqi_report_cnfg =
  {
    mode = "periodic";
    simultaneousAckCQI = true;
    period = 40;                   // in ms
    //subframe = [0, 10, 20, 30];  // Optional vector of subframe indices every period where CQI resources will be
 ↪  allocated (default uses all)
    m_ri = 8; // RI period in CQI period
    //subband_k = 1; // If enabled and > 0, configures sub-band CQI reporting and defines K (see 36.213 7.2.2). If
 ↪  disabled, configures wideband CQI
  };
};

cell_list =
(
  {
    // rf_port = 0;
    cell_id = 0x01;
    tac = 0x0007;
    pci = 1;
    // root_seq_idx = 204;
    dl_earfcn = 3350;
    //ul_earfcn = 21400;
    ho_active = false;
    //meas_gap_period = 0; // 0 (inactive), 40 or 80
    //meas_gap_offset_subframe = [6, 12, 18, 24, 30];
    // target_pusch_sinr = -1;
    // target_pucch_sinr = -1;
    // enable_phr_handling = false;
    // min_phr_thres = 0;
    // allowed_meas_bw = 6;
    // t304 = 2000; // in msec. possible values: 50, 100, 150, 200, 500, 1000, 2000
    // tx_gain = 20.0; // in dB. This gain is set by scaling the source signal.
```

```
    // CA cells
    scell_list = (
      // {cell_id = 0x02; cross_carrier_scheduling = false; scheduling_cell_id = 0x02; ul_allowed = true}
    )

    // Cells available for handover
    meas_cell_list =
    (
      {
        eci = 0x19C02;
        dl_earfcn = 2850;
        pci = 2;
        //direct_forward_path_available = false;
        //allowed_meas_bw = 6;
        //cell_individual_offset = 0;
      }
    );

    // Select measurement report configuration (all reports are combined with all measurement objects)
    meas_report_desc =
    (
        {
          eventA = 3
          a3_offset = 6;
          hysteresis = 0;
          time_to_trigger = 480;
          trigger_quant = "RSRP";
          max_report_cells = 1;
          report_interv = 120;
          report_amount = 1;
        }
    );
    meas_quant_desc = {
        // averaging filter coefficient
        rsrq_config = 4;
        rsrp_config = 4;
     };
  }
  // Add here more cells
);

nr_cell_list =
(
  // no NR cells
);
```

3.118 /opt/GSM/osmo-operator/configs/srsran/sib.conf

10583 octets, 303 lignes → 303 lignes

```
#######################################################################
# sib1 configuration options (See TS 36.331)
#
# additional_plmns: A list of additional PLMN identities.
#     mcc: MCC
#     mnc: MNC
#     cell_reserved_for_oper:  One of "reserved" or "notReserved", default is "notReserved"
#
#######################################################################
sib1 =
{
    intra_freq_reselection = "Allowed";
    q_rx_lev_min = -65;
    //p_max = 3;
```

```
    cell_barred = "NotBarred"
    si_window_length = 20;
    sched_info =
    (
        {
            si_periodicity = 16;

           // comma-separated array of SIB-indexes (from 3 to 13), leave empty or commented to just scheduler sib2
            si_mapping_info = [ 3, 7 ];
        }
    );
    system_info_value_tag = 0;
};

sib2 =
{
    rr_config_common_sib =
    {
        rach_cnfg =
        {
            num_ra_preambles = 52;
            preamble_init_rx_target_pwr = -104;
            pwr_ramping_step = 6;  // in dB
            preamble_trans_max = 10;
            ra_resp_win_size = 10;  // in ms
            mac_con_res_timer = 64; // in ms
            max_harq_msg3_tx = 4;
        };
        bcch_cnfg =
        {
            modification_period_coeff = 16; // in ms
        };
        pcch_cnfg =
        {
            default_paging_cycle = 32; // in rf
            nB = "1";
        };
        prach_cnfg =
        {
            root_sequence_index = 128;
            prach_cnfg_info =
            {
                high_speed_flag = false;
                prach_config_index = 3;
                prach_freq_offset = 4;
                zero_correlation_zone_config = 5;
            };
        };
        pdsch_cnfg =
        {
            /* Warning: Currently disabled and forced to p_b=1 for TM2/3/4 and p_b=0 for TM1
             */
            p_b = 1;
            rs_power = 0;
        };
        pusch_cnfg =
        {
            n_sb = 1;
            hopping_mode = "inter-subframe";
            pusch_hopping_offset = 2;
            enable_64_qam = false; // 64QAM PUSCH is not currently enabled
            ul_rs =
            {
                cyclic_shift = 0;
                group_assignment_pusch = 0;
                group_hopping_enabled = false;
```

```
                sequence_hopping_enabled = false;
            };
        };
        pucch_cnfg =
        {
            delta_pucch_shift = 1;
            n_rb_cqi = 1;
            n_cs_an = 0;
            n1_pucch_an = 12;
        };
        ul_pwr_ctrl =
        {
            p0_nominal_pusch = -85;
            alpha = 0.7;
            p0_nominal_pucch = -107;
            delta_flist_pucch =
            {
                format_1  = 0;
                format_1b = 3;
                format_2  = 1;
                format_2a = 2;
                format_2b = 2;
            };
            delta_preamble_msg3 = 6;
        };
        ul_cp_length = "len1";
    };

    ue_timers_and_constants =
    {
        t300 = 2000; // in ms
        t301 = 100;  // in ms
        t310 = 200; // in ms
        n310 = 1;
        t311 = 10000; // in ms
        n311 = 1;
    };

    freqInfo =
    {
        ul_carrier_freq_present = true;
        ul_bw_present = true;
        additional_spectrum_emission = 1;
    };

    time_alignment_timer = "INFINITY"; // use "sf500", "sf750", etc.
};

sib3 =
{
    cell_reselection_common = {
        q_hyst = 2; // in dB
    },
    cell_reselection_serving = {
        s_non_intra_search = 3,
        thresh_serving_low = 2,
        cell_resel_prio = 6
    },
    intra_freq_reselection = {
        q_rx_lev_min = -61,
        p_max = 23,
        s_intra_search = 5,
        presence_ant_port_1 = true,
        neigh_cell_cnfg = 1,
        t_resel_eutra = 1
    }
```

```
};

######################################################################
# sib5 configuration options (See TS 36.331)
# Contains information relevant for inter-frequency cell re-selection.
# Must be added to sib1::sched_info::si_mapping_info array parameter to be transmitted
#
# inter_freq_carrier_freq_list: A list of neighbouring inter-frequencies.
#     dl_carrier_freq: The EARFCN for the EUTRA carrier frequency.
#     q_rx_lev_min: Minimum received RSRP level in the E-UTRA cell, ([field_val] * 2) = [level in dBm].
#     p_max: Optional maximum allowed transmission power for the neighbouring E-UTRA cells on this carrier
 ↪  frequency.
#     t_resel_eutra: Cell reselection timer (seconds).
#     t_resel_eutra_sf: Optional speed dependent ScalingFactor for t_resel_eutra.
#         sf_medium: Scaling factor if the UE is in Medium Mobility state, one of "0.25", "0.5", "0.75" or "1.0".
#         sf_high: Scaling factor if the UE is in High Mobility state, one of "0.25", "0.5", "0.75" or "1.0".
#     thresh_x_high: Srclev threshold (dB) to select to a higher-priority RAT/Frequency.
#     thresh_x_low: Srclev threshold (dB) to select to a lower-priority RAT/Frequency.
#     allowed_meas_bw: Maximum allowed measurement bandwidth on a carrier frequency .
#     presence_ant_port_1: whether all the neighbouring cells use Antenna Port 1.
#     cell_resel_prio: Optional absolute priority of the carrier frequency group.
#     neigh_cell_cfg: Information related to MBSFN and TDD UL/DL configuration of neighbour cells.
#     q_offset_freq: Frequency specific offset for equal priority E-UTRAN frequencies.
#     inter_freq_neigh_cell_list: A List of inter-frequency neighbouring cells with specific cell re-selection
 ↪  parameters.
#         phys_cell_id: Physical layer identity of the cell.
#         q_offset_cell: Cell spcific offset.
#     inter_freq_black_cell_list: A List of blacklisted inter-frequency neighbouring cells.
#         start: The lowest physical cell identity in the range.
#         range: The number of physical cell identities in the range.
#
######################################################################
sib5 =
{
    inter_freq_carrier_freq_list =
    (
        {
            dl_carrier_freq = 1450;
            q_rx_lev_min = -70;
            t_resel_eutra = 2;
            t_resel_eutra_sf = {
                sf_medium = "0.25";
                sf_high = "1.0";
            };
            thresh_x_high = 3;
            thresh_x_low = 2;
            allowed_meas_bw = 75;
            presence_ant_port_1 = True;
            cell_resel_prio = 4;
            neigh_cell_cfg = 2;
            q_offset_freq = -6;
            inter_freq_neigh_cell_list =
            (
                {
                    phys_cell_id = 500;
                    q_offset_cell = 2;
                }
            );
            inter_freq_black_cell_list =
            (
                {
                    start = 123;
                    range = 4;
                }
            );
        }
```

```
    );
};

################################################################################
# sib6 configuration options (See TS 36.331)
# Contains UTRA neighbor information for inter-rat handover.
# Must be added to sib1::sched_info::si_mapping_info array parameter to be transmitted
#
# t_resel_utra: Cell reselection timer (seconds)
# t_resel_utra_sf: Optional speed dependent ScalingFactor for t_resel_utra.
#     sf_medium: Scaling factor if the UE is in Medium Mobility state, one of "0.25", "0.5", "0.75" or "1.0".
#     sf_high: Scaling factor if the UE is in High Mobility state, one of "0.25", "0.5", "0.75" or "1.0".
# carrier_freq_list_utra_fdd / carrier_freq_list_utra_tdd: A list of carrier frequencies of UTRA FDD / TDD.
#     carrier_freq: The UARFCN for the UTRA carrier frequency.
#     cell_resel_prio: Optional absolute priority of the carrier frequency group.
#     thresh_x_high: Srclev threshold (dB) to select to a higher-priority RAT/Frequency.
#     thresh_x_low: Srclev threshold (dB) to select to a lower-priority RAT/Frequency.
#     q_rx_lev_min: Minimum receive level in UTRA cell, ([field_val] * 2) + 1 = [level in dBm].
#     p_max_utra: The maximum allowed transmission power on the (uplink) carrier frequency.
#     q_qual_min: Minimum required quality leve in UTRA cell, applicable only for FDD cells.
#
################################################################################
sib6 =
{
    t_resel_utra = 1;
    t_resel_utra_sf = {
        sf_medium = "0.25";
        sf_high = "1.0";
    }
    carrier_freq_list_utra_fdd =
    (
        {
            carrier_freq = 9613;
            cell_resel_prio = 6;
            thresh_x_high = 3;
            thresh_x_low = 2;
            q_rx_lev_min = -50;
            p_max_utra = 4;
            q_qual_min = -10;
        }
    );
    carrier_freq_list_utra_tdd =
    (
        {
            carrier_freq = 9505;
            thresh_x_high = 1;
            thresh_x_low = 2;
            q_rx_lev_min = -50;
            p_max_utra = -3;
        }
    );
};

################################################################################
# sib7 configuration options (See TS 36.331)
# Contains GERAN neighbor information for CSFB and inter-rat handover.
# Must be added to sib1::sched_info::si_mapping_info array parameter to be transmitted
#
# t_resel_geran: Cell reselection timer (seconds)
# t_resel_geran_sf: Optional speed dependent ScalingFactor for t_resel_geran.
#     sf_medium: Scaling factor if the UE is in Medium Mobility state, one of "0.25", "0.5", "0.75" or "1.0".
#     sf_high: Scaling factor if the UE is in High Mobility state, one of "0.25", "0.5", "0.75" or "1.0".
# carrier_freqs_info_list: A list of carrier frequency groups.
#     cell_resel_prio: Absolute priority of the carrier frequency group
#     ncc_permitted: 8-bit bitmap of NCC carriers permitted for monitoring
#     q_rx_lev_min: Minimum receive level in gsm cell, ([field_val] * 2) - 115 = [level in dBm]
```

```
#     thresh_x_high: Srclev threshold (dB) to select to a higher-priority RAT/Frequency
#     thresh_x_low: Srclev threshold (dB) to select to a lower-priority RAT/Frequency
#     start_arfcn: Initial search ARFCN value
#    band_ind: One of "dcs1800" or "pcs1900" Disambiguates ARFCNs in these bands, has no meaning for other ARFCNs.
#     explicit_list_of_arfcns: List of ARFCN numbers in the group
#
######################################################################
sib7 =
{
    t_resel_geran = 1;
    carrier_freqs_info_list =
    (
        {
            cell_resel_prio = 0;
            ncc_permitted = 255;
            q_rx_lev_min = 0;
            thresh_x_high = 2;
            thresh_x_low = 2;

            start_arfcn = 1;
            band_ind = "dcs1800";
            explicit_list_of_arfcns = (
                1
            );
        }
    );
};
```

3.119 /opt/GSM/osmo-operator/configs/srsran/ue.conf

19904 octets, 487 lignes → 487 lignes

```
#####################################################################
#                   srsUE configuration file
#####################################################################

#####################################################################
# RF configuration
#
# freq_offset: Uplink and Downlink optional frequency offset (in Hz)
# tx_gain: Transmit gain (dB).
# rx_gain: Optional receive gain (dB). If disabled, AGC if enabled
# srate: Optional fixed sampling rate (Hz), corresponding to cell bandwidth. Must be set for 5G-SA.
#
# nof_antennas:       Number of antennas per carrier (all carriers have the same number of antennas)
# device_name:        Device driver family. Supported options: "auto" (uses first found), "UHD" or "bladeRF"
# device_args:        Arguments for the device driver. Options are "auto" or any string.
#                     Default for UHD: "recv_frame_size=9232,send_frame_size=9232"
#                     Default for bladeRF: ""
# device_args_2:      Arguments for the RF device driver 2.
# device_args_3:      Arguments for the RF device driver 3.
# time_adv_nsamples:  Transmission time advance (in number of samples) to compensate for RF delay
#                     from antenna to timestamp insertion.
#                     Default "auto". B210 USRP: 100 samples, bladeRF: 27.
# continuous_tx:      Transmit samples continuously to the radio or on bursts (auto/yes/no).
#                     Default is auto (yes for UHD, no for rest)
#####################################################################
[rf]
freq_offset = 0
tx_gain = 80
rx_gain = 40
srate = 11.52e6

nof_antennas = 1
```

```
# For best performance in 2x2 MIMO and >= 15 MHz use the following device_args settings:
#     USRP B210: num_recv_frames=64,num_send_frames=64

# For best performance when BW<5 MHz (25 PRB), use the following device_args settings:
#     USRP B210: send_frame_size=512,recv_frame_size=512

#device_args = auto
time_adv_nsamples = auto
continuous_tx     = auto

# Example for ZMQ-based operation with TCP transport for I/Q samples
device_name = zmq
device_args = tx_port=tcp://*:2001,rx_port=tcp://localhost:2000,id=ue,base_srate=11.52e6

#####################################################################
# EUTRA RAT configuration
#
# dl_earfcn:   Downlink EARFCN list.
#
# Optional parameters:
# dl_freq:            Override DL frequency corresponding to dl_earfcn
# ul_freq:            Override UL frequency corresponding to dl_earfcn
# nof_carriers:       Number of carriers
#####################################################################
[rat.eutra]
dl_earfcn = 3350
nof_carriers = 1

#####################################################################
# NR RAT configuration
#
# Optional parameters:
# bands:           List of support NR bands seperated by a comma (default 78)
# nof_carriers:    Number of NR carriers (must be at least 1 for NR support)
#####################################################################
[rat.nr]
# bands = 78
# nof_carriers = 0

#####################################################################
# Packet capture configuration
#
# Packet capture is supported at the MAC, MAC_NR, and NAS layer.
# MAC-layer packets are captured to file a the compact format decoded
# by the Wireshark. For decoding, use the UDP dissector and the UDP
# heuristic dissection. Edit the preferences (Edit > Preferences >
# Protocols > DLT_USER) for DLT_USER to add an entry for DLT=149 with
# Protocol=udp. Further, enable the heuristic dissection in UDP under:
# Analyze > Enabled Protocols > MAC-LTE > mac_lte_udp and MAC-NR > mac_nr_udp
# For more information see: https://wiki.wireshark.org/MAC-LTE
# Using the same filename for mac_filename and mac_nr_filename writes both
# MAC-LTE and MAC-NR to the same file allowing a better analysis.
# NAS-layer packets are dissected with DLT=148, and Protocol = nas-eps.
#
# enable:            Enable packet captures of layers (mac/mac_nr/nas/none) multiple option list
# mac_filename:      File path to use for MAC packet capture
# mac_nr_filename:   File path to use for MAC NR packet capture
# nas_filename:      File path to use for NAS packet capture
#####################################################################
[pcap]
enable = none
mac_filename = /tmp/ue_mac.pcap
mac_nr_filename = /tmp/ue_mac_nr.pcap
nas_filename = /tmp/ue_nas.pcap
```

```
#####################################################################
# Log configuration
#
# Log levels can be set for individual layers. "all_level" sets log
# level for all layers unless otherwise configured.
# Format: e.g. phy_level = info
#
# In the same way, packet hex dumps can be limited for each level.
# "all_hex_limit" sets the hex limit for all layers unless otherwise
# configured.
# Format: e.g. phy_hex_limit = 32
#
# Logging layers: rf, phy, mac, rlc, pdcp, rrc, nas, gw, usim, stack, all
# Logging levels: debug, info, warning, error, none
#
# filename: File path to use for log output. Can be set to stdout
#           to print logs to standard output
# file_max_size: Maximum file size (in kilobytes). When passed, multiple files are created.
#                If set to negative, a single log file will be created.
#####################################################################
[log]
all_level = warning
# [2026-09-09] NAS en info, sans dump hexa : c est ce que montre la console
# srsUE du bureau (tools/osmo-fft-snap.py) - attach, EMM, TAU, PDN. En warning
# seul, le journal restait VIDE apres un attach reussi. RRC reste en warning :
# en info il ecrit une mesure de cellule toutes les 20 ms (5 Ko/s) et noie tout.
nas_level = info
nas_hex_limit = 0
phy_lib_level = none
all_hex_limit = 32
# tools/osmo-lte.sh impose --log.filename=/tmp/osmo-lte-ue.log ; ceci ne sert
# qu a un srsue lance a la main.
filename = /tmp/osmo-lte-ue.log
file_max_size = -1

#####################################################################
# USIM configuration
#
# mode:   USIM mode (soft/pcsc)
# algo:   Authentication algorithm (xor/milenage)
# op/opc: 128-bit Operator Variant Algorithm Configuration Field (hex)
#         - Specify either op or opc (only used in milenage)
# k:      128-bit subscriber key (hex)
# imsi:   15 digit International Mobile Subscriber Identity
# imei:   15 digit International Mobile Station Equipment Identity
# pin:    PIN in case real SIM card is used
# reader: Specify card reader by it's name as listed by 'pcsc_scan'. If empty, try all available readers.
#####################################################################
[usim]
mode = soft
algo = milenage
opc  = 63BFA50EE6523365FF14C1F45F88737D
k    = 00112233445566778899aabbccdd0101
imsi = 001010001000001
imei = 358925004273675
#reader =
#pin  = 1234

#####################################################################
# RRC configuration
#
# ue_category:          Sets UE category (range 1-5). Default: 4
# release:              UE Release (8 to 15)
# feature_group:        Hex value of the featureGroupIndicators field in the
#                       UECapabilityInformation message. Default 0xe6041000
# mbms_service_id:      MBMS service id for autostarting MBMS reception
```

```
#                       (default -1 means disabled)
# mbms_service_port:    Port of the MBMS service
# nr_measurement_pci:   NR PCI for the simulated NR measurement. Default: 500
# nr_short_sn_support:  Announce PDCP short SN support. Default: true
#####################################################################
[rrc]
#ue_category       = 4
#release           = 8
#feature_group     = 0xe6041000
#mbms_service_id   = -1
#mbms_service_port = 4321

#####################################################################
# NAS configuration
#
# apn:               Set Access Point Name (APN)
# apn_protocol:      Set APN protocol (IPv4, IPv6 or IPv4v6.)
# user:              Username for CHAP authentication
# pass:              Password for CHAP authentication
# force_imsi_attach: Whether to always perform an IMSI attach
# eia:               List of integrity algorithms included in UE capabilities
#                      Supported: 1 - Snow3G, 2 - AES, 3 - ZUC
# eea:               List of ciphering algorithms included in UE capabilities
#                      Supported: 0 - NULL, 1 - Snow3G, 2 - AES, 3 - ZUC
#####################################################################
[nas]
apn = srsapn
#apn_protocol = ipv4
#user = srsuser
#pass = srspass
#force_imsi_attach = false
#eia = 1,2,3
#eea = 0,1,2,3

#####################################################################
# Slice configuration
#
# enable:           Enable a specific slice
# nssai-sst:        Specfic Slice Type
# nssai-sd:         Slice diffentiator
#####################################################################
[slicing]
#enable = false
#nssai-sst = 1
#nssai-sd = 1

#####################################################################
# GW configuration
#
# netns:                Network namespace to create TUN device. Default: empty
# ip_devname:           Name of the tun_srsue device. Default: tun_srsue
# ip_netmask:           Netmask of the tun_srsue device. Default: 255.255.255.0
#####################################################################
[gw]
#netns =
#ip_devname = tun_srsue
#ip_netmask = 255.255.255.0

#####################################################################
# GUI configuration
#
# Simple GUI displaying PDSCH constellation and channel freq response.
# (Requires building with srsGUI)
# enable:               Enable the graphical interface (true/false)
#####################################################################
[gui]
```

```
enable = false

######################################################################
# Channel emulator options:
# enable:            Enable/Disable internal Downlink/Uplink channel emulator
#
# -- AWGN Generator
# awgn.enable:       Enable/disable AWGN generator
# awgn.snr:          SNR in dB
# awgn.signal_power: Received signal power in decibels full scale (dBfs)
#
# -- Fading emulator
# fading.enable:     Enable/disable fading simulator
# fading.model:      Fading model + maximum doppler (E.g. none, epa5, eva70, etu300, etc)
#
# -- Delay Emulator     delay(t) = delay_min + (delay_max - delay_min) * (1 + sin(2pi*t/period)) / 2
#                       Maximum speed [m/s]: (delay_max - delay_min) * pi * 300 / period
# delay.enable:      Enable/disable delay simulator
# delay.period_s:    Delay period in seconds.
# delay.init_time_s: Delay initial time in seconds.
# delay.maximum_us:  Maximum delay in microseconds
# delay.minumum_us:  Minimum delay in microseconds
#
# -- Radio-Link Failure (RLF) Emulator
# rlf.enable:        Enable/disable RLF simulator
# rlf.t_on_ms:       Time for On state of the channel (ms)
# rlf.t_off_ms:      Time for Off state of the channel (ms)
#
# -- High Speed Train Doppler model simulator
# hst.enable:        Enable/Disable HST simulator
# hst.period_s:      HST simulation period in seconds
# hst.fd_hz:         Doppler frequency in Hz
# hst.init_time_s:   Initial time in seconds
######################################################################
[channel.dl]
#enable        = false

[channel.dl.awgn]
#enable        = false
#snr           = 30

[channel.dl.fading]
#enable        = false
#model         = none

[channel.dl.delay]
#enable        = false
#period_s      = 3600
#init_time_s   = 0
#maximum_us    = 100
#minimum_us    = 10

[channel.dl.rlf]
#enable        = false
#t_on_ms       = 10000
#t_off_ms      = 2000

[channel.dl.hst]
#enable        = false
#period_s      = 7.2
#fd_hz         = 750.0
#init_time_s   = 0.0

[channel.ul]
#enable        = false
```

```
[channel.ul.awgn]
#enable        = false
#n0            = -30

[channel.ul.fading]
#enable        = false
#model         = none

[channel.ul.delay]
#enable        = false
#period_s      = 3600
#init_time_s   = 0
#maximum_us    = 100
#minimum_us    = 10

[channel.ul.rlf]
#enable        = false
#t_on_ms       = 10000
#t_off_ms      = 2000

[channel.ul.hst]
#enable        = false
#period_s      = 7.2
#fd_hz         = -750.0
#init_time_s   = 0.0

######################################################################
# PHY configuration options
#
# rx_gain_offset:       RX Gain offset to add to rx_gain to calibrate RSRP readings
# prach_gain:           PRACH gain (dB). If defined, forces a gain for the tranmsission of PRACH only.,
#                       Default is to use tx_gain in [rf] section.
# cqi_max:              Upper bound on the maximum CQI to be reported. Default 15.
# cqi_fixed:            Fixes the reported CQI to a constant value. Default disabled.
# snr_ema_coeff:        Sets the SNR exponential moving average coefficient (Default 0.1)
# snr_estim_alg:        Sets the noise estimation algorithm. (Default refs)
#                          Options: pss:   use difference between received and known pss signal,
#                                   refs:  use difference between noise references and noiseless (after filtering)
#                                   empty: use empty subcarriers in the boarder of pss/sss signal
# pdsch_max_its:        Maximum number of turbo decoder iterations (Default 4)
# pdsch_meas_evm:       Measure PDSCH EVM, increases CPU load (default false)
# nof_phy_threads:      Selects the number of PHY threads (maximum 4, minimum 1, default 3)
# equalizer_mode:       Selects equalizer mode. Valid modes are: "mmse", "zf" or any
#                       non-negative real number to indicate a regularized zf coefficient.
#                       Default is MMSE.
# correct_sync_error:   Channel estimator measures and pre-compensates time synchronization error. Increases CPU
 ↪  usage,
#                       improves PDSCH decoding in high SFO and high speed UE scenarios.
# sfo_ema:              EMA coefficient to average sample offsets used to compute SFO
# sfo_correct_period:   Period in ms to correct sample time to adjust for SFO
# sss_algorithm:        Selects the SSS estimation algorithm. Can choose between
#                       {full, partial, diff}.
# estimator_fil_auto:   The channel estimator smooths the channel estimate with an adaptative filter.
# estimator_fil_stddev: Sets the channel estimator smooth gaussian filter standard deviation.
# estimator_fil_order:  Sets the channel estimator smooth gaussian filter order (even values perform better).
#                       The taps are [w, 1-2w, w]
#
# snr_to_cqi_offset:    Sets an offset in the SNR to CQI table. This is used to adjust the reported CQI.
#
# interpolate_subframe_enabled: Interpolates in the time domain the channel estimates within 1 subframe. Default
 ↪  is to average.
#
# pdsch_csi_enabled:     Stores the Channel State Information and uses it for weightening the softbits. It is only
#                        used in TM1. It is True by default.
#
# pdsch_8bit_decoder:    Use 8-bit for LLR representation and turbo decoder trellis computation (Experimental)
```

```
# force_ul_amplitude:    Forces the peak amplitude in the PUCCH, PUSCH and SRS (set 0.0 to 1.0, set to 0 or
 ↪  negative for disabling)
#
# in_sync_rsrp_dbm_th:    RSRP threshold (in dBm) above which the UE considers to be in-sync
# in_sync_snr_db_th:      SNR threshold (in dB) above which the UE considers to be in-sync
# nof_in_sync_events:     Number of PHY in-sync events before sending an in-sync event to RRC
# nof_out_of_sync_events: Number of PHY out-sync events before sending an out-sync event to RRC
#
# force_N_id_2: Force using a specific PSS (set to -1 to allow all PSSs).
# force_N_id_1: Force using a specific SSS (set to -1 to allow all SSSs).
#
######################################################################
[phy]
#rx_gain_offset      = 62
#prach_gain          = 30
#cqi_max             = 15
#cqi_fixed           = 10
#snr_ema_coeff       = 0.1
#snr_estim_alg       = refs
#pdsch_max_its       = 8    # These are half iterations
#pdsch_meas_evm      = false
#nof_phy_threads     = 3
#equalizer_mode      = mmse
#correct_sync_error  = false
#sfo_ema             = 0.1
#sfo_correct_period  = 10
#sss_algorithm       = full
#estimator_fil_auto  = false
#estimator_fil_stddev  = 1.0
#estimator_fil_order  = 4
#snr_to_cqi_offset   = 0.0
#interpolate_subframe_enabled = false
#pdsch_csi_enabled  = true
#pdsch_8bit_decoder = false
#force_ul_amplitude = 0
#detect_cp          = false

#in_sync_rsrp_dbm_th    = -130.0
#in_sync_snr_db_th      = 3.0
#nof_in_sync_events     = 10
#nof_out_of_sync_events = 20

#force_N_id_2           = 1
#force_N_id_1           = 10

######################################################################
# PHY NR specific configuration options
#
# store_pdsch_ko:       Dumps the PDSCH baseband samples into a file on KO reception
#
######################################################################
[phy.nr]
#store_pdsch_ko = false

######################################################################
# CFR configuration options
#
# The CFR module provides crest factor reduction for the transmitted signal.
#
# enable:           Enable or disable the CFR. Default: disabled
#
# mode:             manual:   CFR threshold is set by cfr_manual_thres (default).
#                   auto_ema: CFR threshold is adaptive based on the signal PAPR. Power avg. with Exponential
 ↪  Moving Average.
#                             The time constant of the averaging can be tweaked with the ema_alpha parameter.
```

```
#                  auto_cma: CFR threshold is adaptive based on the signal PAPR. Power avg. with Cumulative Moving
 ↪  Average.
#                          Use with care, as CMA's increasingly slow response may be unsuitable for most use cases.
#
# strength:         Ratio between amplitude-limited vs unprocessed signal (0 to 1). Default: 1
# manual_thres:     Fixed manual clipping threshold for CFR manual mode. Default: 2
# auto_target_papr: Signal PAPR target (in dB) in CFR auto modes. output PAPR can be higher due to peak smoothing.
 ↪  Default: 7
# ema_alpha:        Alpha coefficient for the power average in auto_ema mode. Default: 1/7
#
#############################################################################
[cfr]
#enable           = false
#mode             = manual
#manual_thres     = 2.0
#strength         = 1.0
#auto_target_papr = 7.0
#ema_alpha        = 0.0143

#############################################################################
# Simulation configuration options
#
# The UE simulation supports turning on and off airplane mode in the UE.
# The actions are carried periodically until the UE is stopped.
#
# airplane_t_on_ms:   Time to leave airplane mode turned on (in ms)
#
# airplane_t_off_ms:  Time to leave airplane mode turned off (in ms)
#
#############################################################################
[sim]
#airplane_t_on_ms  = -1
#airplane_t_off_ms = -1

#############################################################################
# General configuration options
#
# metrics_csv_enable:    Write UE metrics to CSV file.
#
# metrics_period_secs:   Sets the period at which metrics are requested from the UE.
#
# metrics_csv_filename:  File path to use for CSV metrics.
#
# tracing_enable:        Write source code tracing information to a file.
#
# tracing_filename:      File path to use for tracing information.
#
# tracing_buffcapacity:  Maximum capacity in bytes the tracing framework can store.
#
# have_tti_time_stats:   Calculate TTI execution statistics using system clock
#
# metrics_json_enable:   Write UE metrics to JSON file.
#
# metrics_json_filename: File path to use for JSON metrics.
#
#############################################################################
[general]
#metrics_csv_enable    = false
#metrics_period_secs   = 1
#metrics_csv_filename  = /tmp/ue_metrics.csv
#have_tti_time_stats   = true
#tracing_enable        = true
#tracing_filename      = /tmp/ue_tracing.log
#tracing_buffcapacity  = 1000000
#metrics_json_enable   = false
#metrics_json_filename = /tmp/ue_metrics.json
```

3.120 /opt/GSM/osmo-operator/data/c123-zones.json

1528 octets, 135 lignes → 113 lignes (22 groupes compactés)

```
{
  "_doc": "Zones cliquables de la photo du C123 (data/c123.jpg, 150x331), en coordonnees normalisees [x, y,
  ↪  largeur, hauteur] (0..1 de l image). Recalees a la main sur la photo le 21/10 (les touches etaient une
  ↪  demi-touche trop bas). `osmo-c123-gtk.py --calibrate` les reecrit a la souris.",
  "lcd": [
    0.⟨1⟩,  ×3
    ⟨⟩ = (16) (195) (67)
    0.215
  ],
  "lsb": [
    0.⟨1⟩,  ×3
    ⟨⟩ = (15) (495) (27)
    0.045
  ],
  "rsb": [
    0.⟨1⟩,  ×3
    ⟨⟩ = (58) (495) (27)
    0.045
  ],
  "up": [
    0.⟨1⟩,  ×3
    ⟨⟩ = (4) (52) (2)
    0.048
  ],
  "left": [
    0.⟨1⟩,  ×3
    ⟨⟩ = (335) (565) (09)
    0.06
  ],
  "menu": [
    0.⟨1⟩,  ×3
    ⟨⟩ = (435) (56) (13)
    0.07
  ],
  "right": [
    0.⟨1⟩,  ×3
    ⟨⟩ = (575) (565) (09)
    0.06
  ],
  "down": [
    0.⟨1⟩,  ×3
    ⟨⟩ = (4) (618) (2)
    0.048
  ],
  "ok": [
    0.⟨1⟩,  ×3
    ⟨⟩ = (115) (555) (175)
    0.09
  ],
  "power": [
    0.⟨1⟩,  ×3
    ⟨⟩ = (705) (555) (175)
    0.09
  ],
  "1": [
    0.⟨1⟩,  ×3
    ⟨⟩ = (11) (662) (23)
    0.052
  ],
  "2": [
    0.⟨1⟩,  ×3
    ⟨⟩ = (36) (662) (27)
```

```
    0.052
  ],
  "3": [
    0.⟨1⟩,  ×3
    ⟨⟩ = (66) (662) (23)
    0.052
  ],
  "4": [
    0.⟨1⟩,  ×3
    ⟨⟩ = (11) (726) (23)
    0.052
  ],
  "5": [
    0.⟨1⟩,  ×3
    ⟨⟩ = (36) (726) (27)
    0.052
  ],
  "6": [
    0.⟨1⟩,  ×3
    ⟨⟩ = (66) (726) (23)
    0.052
  ],
  "7": [
    0.⟨1⟩,  ×3
    ⟨⟩ = (11) (79) (23)
    0.052
  ],
  "8": [
    0.⟨1⟩,  ×3
    ⟨⟩ = (36) (79) (27)
    0.052
  ],
  "9": [
    0.⟨1⟩,  ×3
    ⟨⟩ = (66) (79) (23)
    0.052
  ],
  "star": [
    0.⟨1⟩,  ×3
    ⟨⟩ = (11) (852) (23)
    0.052
  ],
  "0": [
    0.⟨1⟩,  ×3
    ⟨⟩ = (36) (852) (27)
    0.052
  ],
  "hash": [
    0.⟨1⟩,  ×3
    ⟨⟩ = (66) (852) (23)
    0.052
  ]
}
```

3.121 /opt/GSM/osmo-operator/data/c139-zones.json

1122 octets, 25 lignes → 19 lignes (1 groupes compactés)

```
{
  "_doc": "Zones cliquables de la photo du C139 (data/c139.jpg, 720x1561, recadree de « Motorola C139-1170.jpg »
  ↪  de Raimond Spekking, Wikimedia Commons, CC BY-SA 4.0), en coordonnees normalisees [x, y, largeur, hauteur].
  ↪  Posees le 21/10/2026 sur une grille ; `osmo-c123-gtk.py --modele c139 --calibrate` les reecrit a la souris.",
  "lcd": [0.15, 0.20, 0.69, 0.22],
  "lsb": [0.11, 0.50, 0.22, 0.045],
```

```
  "rsb": [0.67, 0.50, 0.22, 0.045],
  "up": [0.44, 0.535, 0.12, 0.04],
  "left": [0.34, 0.575, 0.08, 0.055],
  "menu": [0.43, 0.575, 0.14, 0.055],
  "right": [0.58, 0.575, 0.08, 0.055],
  "down": [0.44, 0.63, 0.12, 0.04],
  "ok": [0.12, 0.565, 0.18, 0.07],
  "power": [0.70, 0.565, 0.18, 0.07],
  "⟨1⟩": [0.⟨2⟩, 0.⟨3⟩, 0.⟨4⟩, 0.045],  ×9
    ⟨⟩ = (1,12,655,23) (2,40,69,20) (3,65,655,23) (4,12,72,23) (5,40,755,20) (6,65,72,23) (7,12,785,23)
         (8,40,82,20) (9,65,785,23)
  "star": [0.12, 0.85, 0.23, 0.045],
  "0": [0.40, 0.885, 0.20, 0.045],
  "hash": [0.65, 0.85, 0.23, 0.045]
}
```

3.122 /opt/GSM/osmo-operator/data/claude.svg

1075 octets, 23 lignes → 23 lignes

```
<?xml version="1.0" encoding="UTF-8"?>
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 256 256" width="256" height="256">
  <!-- Icone Claude : tuile clay (#D97757) + sunburst Anthropic creme -->
  <rect x="8" y="8" width="240" height="240" rx="56" fill="#D97757"/>
  <g transform="translate(128,128)" fill="#F5EFE6">
    <!-- rayon long et rayon court, alternes, pointe vers l exterieur -->
    <path id="rl" d="M0,-96 L7,-16 L-7,-16 Z"/>
    <path id="rs" d="M0,-74 L6,-14 L-6,-14 Z"/>
    <use href="#rl" transform="rotate(0)"/>
    <use href="#rs" transform="rotate(30)"/>
    <use href="#rl" transform="rotate(60)"/>
    <use href="#rs" transform="rotate(90)"/>
    <use href="#rl" transform="rotate(120)"/>
    <use href="#rs" transform="rotate(150)"/>
    <use href="#rl" transform="rotate(180)"/>
    <use href="#rs" transform="rotate(210)"/>
    <use href="#rl" transform="rotate(240)"/>
    <use href="#rs" transform="rotate(270)"/>
    <use href="#rl" transform="rotate(300)"/>
    <use href="#rs" transform="rotate(330)"/>
    <circle r="17"/>
  </g>
</svg>
```

3.123 /opt/GSM/osmo-operator/data/deka-toy.svg

7511 octets, 96 lignes → 32 lignes (6 groupes compactés)

```
<?xml version="1.0" encoding="UTF-8"?>
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 256 256" width="256" height="256">
<rect x="8" y="8" width="240" height="240" rx="54" fill="#ffffff"/>
<g fill="none" stroke="#0a0d10" stroke-width="2">
<circle cx="128" cy="128" r="9"/>
<circle cx="128" cy="128" r="15"/>
<circle cx="⟨1⟩.⟨2⟩" cy="⟨3⟩.⟨4⟩" r="1.6"/>  ×8
    ⟨⟩ = (143,0,128,0) (138,6,138,6) (128,0,143,0) (117,4,138,6) (113,0,128,0) (117,4,117,4) (128,0,113,0)
         (138,6,117,4)
<circle cx="⟨1⟩.0" cy="⟨2⟩.0" r="3"/>  ×12
    ⟨⟩ = (158,128) (154,143) (143,154) (128,158) (113,154) (102,143) (98,128) (102,113) (113,102) (128,98)
         (143,102) (154,113)
<rect x="⟨1⟩.⟨2⟩" y="⟨3⟩.⟨4⟩" width="5" height="5" rx="1" transform="rotate(⟨5⟩.⟨6⟩ ⟨7⟩.⟨8⟩ ⟨9⟩.⟨10⟩)"/>  ×16
    ⟨⟩ = (168,7,134,1,11,2,171,2,136,6) (162,1,149,9,33,8,164,6,152,4) (149,9,162,1,56,2,152,4,164,6)
```

```
        (134,1,168,7,78,8,136,6,171,2) (116,9,168,7,101,2,119,4,171,2) (101,1,162,1,123,7,103,6,164,6)
        (88,9,149,9,146,2,91,4,152,4) (82,3,134,1,168,8,84,8,136,6) (82,3,116,9,191,2,84,8,119,4)
        (88,9,101,1,213,8,91,4,103,6) (101,1,88,9,236,2,103,6,91,4) (116,9,82,3,258,8,119,4,84,8)
        (134,1,82,3,281,2,136,6,84,8) (149,9,88,9,303,8,152,4,91,4) (162,1,101,1,326,2,164,6,103,6)
        (168,7,116,9,348,8,171,2,119,4)
<circle cx="⟨1⟩.⟨2⟩" cy="⟨3⟩.⟨4⟩" r="⟨5⟩"/>  ×36
    ⟨⟩ = (186,0,128,0,3) (183,2,145,9,3) (174,9,162,1,3) (162,1,174,9,3) (145,9,183,2,3) (128,0,186,0,3)
        (110,1,183,2,3) (93,9,174,9,3) (81,1,162,1,3) (72,8,145,9,3) (70,0,128,0,3) (72,8,110,1,3)
        (81,1,93,9,3) (93,9,81,1,3) (110,1,72,8,3) (128,0,70,0,3) (145,9,72,8,3) (162,1,81,1,3) (174,9,93,9,3)
        (183,2,110,1,3) (200,6,142,4,4) (189,5,169,1,4) (169,1,189,5,4) (142,4,200,6,4) (113,6,200,6,4)
        (86,9,189,5,4) (66,5,169,1,4) (55,4,142,4,4) (55,4,113,6,4) (66,5,86,9,4) (86,9,66,5,4) (113,6,55,4,4)
        (142,4,55,4,4) (169,1,66,5,4) (189,5,86,9,4) (200,6,113,6,4)
<g transform="rotate(⟨1⟩.0 128 128)"><rect x="120.0" y="32.0" width="16.0" height="8" rx="1.5"/><line x1="122.0"
 ↪  y1="36.0" x2="134.0" y2="36.0"/><rect x="121.5" y="22.0" width="13.0" height="8" rx="1.5"/><line x1="123.5"
 ↪  y1="26.0" x2="132.5" y2="26.0"/><rect x="123.0" y="12.0" width="10.0" height="8" rx="1.5"/><line x1="125.0"
 ↪  y1="16.0" x2="131.0" y2="16.0"/></g>  ×8
    ⟨⟩ = (0) (45) (90) (135) (180) (225) (270) (315)
<g transform="rotate(⟨1⟩.5 128 128)"><line x1="128" y1="48" x2="128" y2="38"/></g>  ×8
    ⟨⟩ = (22) (67) (112) (157) (202) (247) (292) (337)
</g>
</svg>
```

3.124 /opt/GSM/osmo-operator/data/deka.svg

7511 octets, 96 lignes → 32 lignes (6 groupes compactés)

```
<?xml version="1.0" encoding="UTF-8"?>
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 256 256" width="256" height="256">
<rect x="8" y="8" width="240" height="240" rx="54" fill="#0a0d10"/>
<g fill="none" stroke="#9db8c9" stroke-width="2">
<circle cx="128" cy="128" r="9"/>
<circle cx="128" cy="128" r="15"/>
<circle cx="⟨1⟩.⟨2⟩" cy="⟨3⟩.⟨4⟩" r="1.6"/>  ×8
    ⟨⟩ = (143,0,128,0) (138,6,138,6) (128,0,143,0) (117,4,138,6) (113,0,128,0) (117,4,117,4) (128,0,113,0)
        (138,6,117,4)
<circle cx="⟨1⟩.0" cy="⟨2⟩.0" r="3"/>  ×12
    ⟨⟩ = (158,128) (154,143) (143,154) (128,158) (113,154) (102,143) (98,128) (102,113) (113,102) (128,98)
        (143,102) (154,113)
<rect x="⟨1⟩.⟨2⟩" y="⟨3⟩.⟨4⟩" width="5" height="5" rx="1" transform="rotate(⟨5⟩.⟨6⟩ ⟨7⟩.⟨8⟩ ⟨9⟩.⟨10⟩)"/>  ×16
    ⟨⟩ = (168,7,134,1,11,2,171,2,136,6) (162,1,149,9,33,8,164,6,152,4) (149,9,162,1,56,2,152,4,164,6)
        (134,1,168,7,78,8,136,6,171,2) (116,9,168,7,101,2,119,4,171,2) (101,1,162,1,123,7,103,6,164,6)
        (88,9,149,9,146,2,91,4,152,4) (82,3,134,1,168,8,84,8,136,6) (82,3,116,9,191,2,84,8,119,4)
        (88,9,101,1,213,8,91,4,103,6) (101,1,88,9,236,2,103,6,91,4) (116,9,82,3,258,8,119,4,84,8)
        (134,1,82,3,281,2,136,6,84,8) (149,9,88,9,303,8,152,4,91,4) (162,1,101,1,326,2,164,6,103,6)
        (168,7,116,9,348,8,171,2,119,4)
<circle cx="⟨1⟩.⟨2⟩" cy="⟨3⟩.⟨4⟩" r="⟨5⟩"/>  ×36
    ⟨⟩ = (186,0,128,0,3) (183,2,145,9,3) (174,9,162,1,3) (162,1,174,9,3) (145,9,183,2,3) (128,0,186,0,3)
        (110,1,183,2,3) (93,9,174,9,3) (81,1,162,1,3) (72,8,145,9,3) (70,0,128,0,3) (72,8,110,1,3)
        (81,1,93,9,3) (93,9,81,1,3) (110,1,72,8,3) (128,0,70,0,3) (145,9,72,8,3) (162,1,81,1,3) (174,9,93,9,3)
        (183,2,110,1,3) (200,6,142,4,4) (189,5,169,1,4) (169,1,189,5,4) (142,4,200,6,4) (113,6,200,6,4)
        (86,9,189,5,4) (66,5,169,1,4) (55,4,142,4,4) (55,4,113,6,4) (66,5,86,9,4) (86,9,66,5,4) (113,6,55,4,4)
        (142,4,55,4,4) (169,1,66,5,4) (189,5,86,9,4) (200,6,113,6,4)
<g transform="rotate(⟨1⟩.0 128 128)"><rect x="120.0" y="32.0" width="16.0" height="8" rx="1.5"/><line x1="122.0"
 ↪  y1="36.0" x2="134.0" y2="36.0"/><rect x="121.5" y="22.0" width="13.0" height="8" rx="1.5"/><line x1="123.5"
 ↪  y1="26.0" x2="132.5" y2="26.0"/><rect x="123.0" y="12.0" width="10.0" height="8" rx="1.5"/><line x1="125.0"
 ↪  y1="16.0" x2="131.0" y2="16.0"/></g>  ×8
    ⟨⟩ = (0) (45) (90) (135) (180) (225) (270) (315)
<g transform="rotate(⟨1⟩.5 128 128)"><line x1="128" y1="48" x2="128" y2="38"/></g>  ×8
    ⟨⟩ = (22) (67) (112) (157) (202) (247) (292) (337)
</g>
</svg>
```

3.125 /opt/GSM/osmo-operator/data/osmo-dino.svg

1144 octets, 25 lignes → 25 lignes

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 64 64" width="64" height="64">
  <!-- osmo-dino.svg - le coureur, en pixels de 4 px sur une grille de 16.
       Gris #535353 : le meme sur un menu clair et sur un dock sombre, sans
       avoir a fournir deux variantes. Le cadre reste transparent. -->
  <g fill="#535353" shape-rendering="crispEdges">
    <!-- tete et museau -->
    <rect x="36" y="8"  width="20" height="16"/>
    <rect x="52" y="20" width="8"  height="4"/>
    <!-- cou -->
    <rect x="36" y="24" width="12" height="8"/>
    <!-- corps -->
    <rect x="16" y="28" width="32" height="16"/>
    <!-- queue -->
    <rect x="8"  y="24" width="12" height="8"/>
    <rect x="4"  y="28" width="8"  height="4"/>
    <!-- bras -->
    <rect x="32" y="36" width="8"  height="4"/>
    <!-- pattes : l'une posee, l'autre levee - il court -->
    <rect x="20" y="44" width="8"  height="12"/>
    <rect x="16" y="52" width="12" height="4"/>
    <rect x="36" y="44" width="8"  height="8"/>
  </g>
  <!-- l'oeil : un trou dans la tete, pas un point clair pose dessus -->
  <rect x="48" y="12" width="4" height="4" fill="#fff"/>
</svg>
```

3.126 /opt/GSM/osmo-operator/data/osmo-info.svg

288 octets, 1 lignes → 1 lignes

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 128 128"><circle cx="64" cy="64" r="58" fill="#d62828"
 ↪  stroke="#ff6b6b" stroke-width="4"/><text x="64" y="94" font-family="Ubuntu,DejaVu Sans,sans-serif"
 ↪  font-weight="700" font-size="92" text-anchor="middle" fill="#fff">?</text></svg>
```

3.127 /opt/GSM/osmo-operator/data/osmo-launch.svg

2521 octets, 51 lignes → 51 lignes

```
<?xml version="1.0" encoding="UTF-8"?>
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 64 64" width="64" height="64">
<!-- ⚠ CE COMMENTAIRE EST *DANS* <svg>, ET IL DOIT Y RESTER.
     gdk-pixbuf ne RENIFLE le type d un fichier que sur ses ~256 premiers
     octets : il y cherche la balise <svg. Un commentaire d en-tete un peu
     long la repousse au-dela, et le chargeur repond alors
         Couldn't recognize the image file format
     sur un SVG parfaitement valide - `file` le reconnait, un parseur XML
     l accepte, et l icone se remplace quand meme par la page blanche
     generique sur le bureau. Constate le 31/08 sur les trois icones.
     Tout texte d explication va donc APRES la balise ouvrante. -->
<!-- osmo-launch.svg - icone du lanceur du banc osmo-operator.
     Combine dessine a la main plutot qu un nom d icone du theme : "call-start"
     est VERT dans Adwaita et n a pas de variante rouge, et une icone de theme
     absente se remplace en silence par un rectangle gris - le lanceur devient
     alors introuvable sur le bureau. Un SVG du depot ne peut pas manquer. -->
  <defs>
    <linearGradient id="corps" x1="0" y1="0" x2="0" y2="1">
      <stop offset="0%"   stop-color="#e8462f"/>
```

```
      <stop offset="100%" stop-color="#b3160b"/>
    </linearGradient>
    <linearGradient id="brillance" x1="0" y1="0" x2="0" y2="1">
      <stop offset="0%"   stop-color="#ffffff" stop-opacity="0.30"/>
      <stop offset="55%"  stop-color="#ffffff" stop-opacity="0.05"/>
      <stop offset="100%" stop-color="#ffffff" stop-opacity="0"/>
    </linearGradient>
  </defs>

  <!-- Pastille de fond -->
  <rect x="3" y="3" width="58" height="58" rx="14" ry="14" fill="url(#corps)"/>
  <rect x="3" y="3" width="58" height="58" rx="14" ry="14" fill="url(#brillance)"/>
  <rect x="3.75" y="3.75" width="56.5" height="56.5" rx="13.5" ry="13.5"
        fill="none" stroke="#7d0d05" stroke-width="1.5" opacity="0.55"/>

  <!-- Combine : ecouteur haut-gauche, micro bas-droite, cordon en arc.
       Trace en une seule forme pour rester net a 24 px comme a 256 px. -->
  <g transform="rotate(-32 32 32)">
    <path fill="#ffffff" d="
      M 20.2 14.0
      c 3.9 0 7.0 3.1 7.0 7.0
      c 0 2.6 -1.4 4.8 -3.4 6.1
      c -1.1 0.7 -1.6 2.0 -1.2 3.2
      c 1.6 4.9 5.4 8.7 10.3 10.3
      c 1.2 0.4 2.5 -0.1 3.2 -1.2
      c 1.3 -2.0 3.5 -3.4 6.1 -3.4
      c 3.9 0 7.0 3.1 7.0 7.0
      c 0 3.9 -3.1 7.0 -7.0 7.0
      C 27.4 50.0 14.0 36.6 14.0 21.0
      c 0 -3.9 3.1 -7.0 6.2 -7.0 z"/>
  </g>
</svg>
```

3.128 /opt/GSM/osmo-operator/data/osmo-lte.svg

755 octets, 11 lignes → 11 lignes

```
<?xml version="1.0" encoding="UTF-8"?>
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 64 64" width="64" height="64">
<!-- osmo-lte : la 4G du banc. Commentaire court, APRES <svg (gdk-pixbuf ne
     renifle que les ~256 premiers octets ; voir osmo-launch.svg). -->
<rect x="4" y="4" width="56" height="56" rx="12" fill="#1e3a8a"/>
<rect x="12" y="38" width="6" height="12" rx="1.5" fill="#93c5fd"/>
<rect x="21" y="31" width="6" height="19" rx="1.5" fill="#bfdbfe"/>
<rect x="30" y="24" width="6" height="26" rx="1.5" fill="#dbeafe"/>
<rect x="39" y="17" width="6" height="33" rx="1.5" fill="#ffffff"/>
<text x="55" y="19" font-family="DejaVu Sans,Arial,sans-serif" font-size="13" font-weight="bold" fill="#fbbf24"
 ↪  text-anchor="end">4G</text>
</svg>
```

3.129 /opt/GSM/osmo-operator/data/osmo-multi.svg

2546 octets, 46 lignes → 46 lignes

```
<?xml version="1.0" encoding="UTF-8"?>
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 64 64" width="64" height="64">
<!-- ⚠ CE COMMENTAIRE EST *DANS* <svg>, ET IL DOIT Y RESTER.
     gdk-pixbuf ne RENIFLE le type d un fichier que sur ses ~256 premiers
     octets : il y cherche la balise <svg. Un commentaire d en-tete un peu
     long la repousse au-dela, et le chargeur repond alors
         Couldn't recognize the image file format
     sur un SVG parfaitement valide - `file` le reconnait, un parseur XML
```

```
     l accepte, et l icone se remplace quand meme par la page blanche
     generique sur le bureau. Constate le 31/08 sur les trois icones.
     Tout texte d explication va donc APRES la balise ouvrante. -->
<!-- osmo-multi.svg - icone du lancement multi-operateur (start-multi.sh).
     Meme facture que osmo-launch.svg / osmo-tutorial.svg : pastille + glyphe
     blanc, pour que les trois se lisent comme une famille sur le bureau.
     Antenne bleue : c est le SS7 entre noeuds, pas un appel. -->
  <defs>
    <linearGradient id="corpsM" x1="0" y1="0" x2="0" y2="1">
      <stop offset="0%"   stop-color="#2f9fd6"/>
      <stop offset="100%" stop-color="#0d5f8f"/>
    </linearGradient>
    <linearGradient id="brillanceM" x1="0" y1="0" x2="0" y2="1">
      <stop offset="0%"   stop-color="#ffffff" stop-opacity="0.30"/>
      <stop offset="55%"  stop-color="#ffffff" stop-opacity="0.05"/>
      <stop offset="100%" stop-color="#ffffff" stop-opacity="0"/>
    </linearGradient>
  </defs>

  <rect x="3" y="3" width="58" height="58" rx="14" ry="14" fill="url(#corpsM)"/>
  <rect x="3" y="3" width="58" height="58" rx="14" ry="14" fill="url(#brillanceM)"/>
  <rect x="3.75" y="3.75" width="56.5" height="56.5" rx="13.5" ry="13.5"
        fill="none" stroke="#08405f" stroke-width="1.5" opacity="0.55"/>

  <g fill="none" stroke="#ffffff" stroke-linecap="round">
    <!-- Ondes, deux paires : l antenne rayonne des DEUX cotes, c est ce qui
         la distingue d un simple pylone et dit le lien entre noeuds. -->
    <path d="M 22.5 20.5 a 13 13 0 0 0 0 17"  stroke-width="3"/>
    <path d="M 41.5 20.5 a 13 13 0 0 1 0 17"  stroke-width="3"/>
    <path d="M 16.5 15.5 a 21 21 0 0 0 0 27"  stroke-width="3" opacity="0.55"/>
    <path d="M 47.5 15.5 a 21 21 0 0 1 0 27"  stroke-width="3" opacity="0.55"/>
  </g>

  <!-- Mat + embase -->
  <circle cx="32" cy="26" r="4.2" fill="#ffffff"/>
  <path fill="#ffffff" d="M 30.4 30.2 h 3.2 l 3.0 20.4 h -4.0 l -0.6 -5.6 h -0.0
                          l -0.6 5.6 h -4.0 z"/>
</svg>
```

3.130 /opt/GSM/osmo-operator/data/osmo-paint.svg

448 octets, 9 lignes → 9 lignes

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 64 64" width="64" height="64">
  <rect x="5" y="5" width="54" height="54" rx="13" fill="#1b1d22"/>
  <path d="M13 47 C24 25, 31 53, 42 29" fill="none" stroke="#00d1ff"
        stroke-width="5" stroke-linecap="round"/>
  <g transform="rotate(45 45 43)">
    <rect x="40" y="18" width="10" height="24" rx="2" fill="#ff9500"/>
    <polygon points="40,42 50,42 45,53" fill="#ffd199"/>
  </g>
</svg>
```

3.131 /opt/GSM/osmo-operator/data/osmo-tutorial.svg

2545 octets, 56 lignes → 56 lignes

```
<?xml version="1.0" encoding="UTF-8"?>
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 64 64" width="64" height="64">
<!-- ⚠ CE COMMENTAIRE EST *DANS* <svg>, ET IL DOIT Y RESTER.
     gdk-pixbuf ne RENIFLE le type d un fichier que sur ses ~256 premiers
     octets : il y cherche la balise <svg. Un commentaire d en-tete un peu
```

```
     long la repousse au-dela, et le chargeur repond alors
         Couldn't recognize the image file format
     sur un SVG parfaitement valide - `file` le reconnait, un parseur XML
     l accepte, et l icone se remplace quand meme par la page blanche
     generique sur le bureau. Constate le 31/08 sur les trois icones.
     Tout texte d explication va donc APRES la balise ouvrante. -->
<!-- osmo-tutorial.svg - icone du tutoriel. Meme facture que osmo-launch.svg
     (pastille + glyphe blanc) pour que les deux se lisent comme une paire sur
     le bureau ; teinte bleue pour qu on les distingue d un coup d oeil. -->
  <defs>
    <linearGradient id="corpsL" x1="0" y1="0" x2="0" y2="1">
      <stop offset="0%"   stop-color="#3f7fd0"/>
      <stop offset="100%" stop-color="#1b4788"/>
    </linearGradient>
    <linearGradient id="brillanceL" x1="0" y1="0" x2="0" y2="1">
      <stop offset="0%"   stop-color="#ffffff" stop-opacity="0.30"/>
      <stop offset="55%"  stop-color="#ffffff" stop-opacity="0.05"/>
      <stop offset="100%" stop-color="#ffffff" stop-opacity="0"/>
    </linearGradient>
  </defs>

  <rect x="3" y="3" width="58" height="58" rx="14" ry="14" fill="url(#corpsL)"/>
  <rect x="3" y="3" width="58" height="58" rx="14" ry="14" fill="url(#brillanceL)"/>
  <rect x="3.75" y="3.75" width="56.5" height="56.5" rx="13.5" ry="13.5"
        fill="none" stroke="#12305d" stroke-width="1.5" opacity="0.55"/>

  <!-- Livre ouvert : deux pages et la reliure centrale. -->
  <g>
    <path fill="#ffffff" d="
      M 31.0 20.4
      c -3.6 -2.6 -8.0 -3.9 -13.1 -3.9
      c -1.6 0 -3.1 0.1 -4.6 0.4
      c -0.8 0.1 -1.3 0.8 -1.3 1.6
      v 24.6
      c 0 1.0 0.9 1.8 1.9 1.6
      c 1.3 -0.2 2.7 -0.3 4.0 -0.3
      c 5.0 0 9.2 1.3 12.6 3.8
      c 0.2 0.2 0.4 0.2 0.5 0.2 z"/>
    <path fill="#ffffff" d="
      M 33.0 20.4
      c 3.6 -2.6 8.0 -3.9 13.1 -3.9
      c 1.6 0 3.1 0.1 4.6 0.4
      c 0.8 0.1 1.3 0.8 1.3 1.6
      v 24.6
      c 0 1.0 -0.9 1.8 -1.9 1.6
      c -1.3 -0.2 -2.7 -0.3 -4.0 -0.3
      c -5.0 0 -9.2 1.3 -12.6 3.8
      c -0.2 0.2 -0.4 0.2 -0.5 0.2 z"/>
    <rect x="31.1" y="20.0" width="1.8" height="28.6" rx="0.9" fill="#ffffff" opacity="0.55"/>
  </g>
</svg>
```

3.132 /opt/GSM/osmo-operator/data/tutorial.html

5530 octets, 107 lignes → 107 lignes

```
<!doctype html>
<html lang="fr">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>osmo-operator — Démarrage rapide</title>
<style>
  :root{
```

```
    --fond:#f7f6f4; --carte:#ffffff; --texte:#1c1b19; --doux:#5f5c57;
    --trait:#e0ddd7; --rouge:#b3160b; --bleu:#1b4788; --code:#f0eee9;
  }
  @media (prefers-color-scheme: dark){
    :root:not([data-theme="light"]){
      --fond:#161513; --carte:#201f1c; --texte:#eceae6; --doux:#a5a09a;
      --trait:#35332e; --rouge:#f0654e; --bleu:#7aa8e8; --code:#2a2825;
    }
  }
  :root[data-theme="dark"]{
    --fond:#161513; --carte:#201f1c; --texte:#eceae6; --doux:#a5a09a;
    --trait:#35332e; --rouge:#f0654e; --bleu:#7aa8e8; --code:#2a2825;
  }
  *{box-sizing:border-box}
  body{margin:0;background:var(--fond);color:var(--texte);
       font:16px/1.6 system-ui,-apple-system,"Segoe UI",Roboto,sans-serif}
  .page{max-width:760px;margin:0 auto;padding:40px 24px 64px}
  h1{font-size:28px;margin:0 0 4px;letter-spacing:-.01em}
  .sous{color:var(--doux);margin:0 0 32px}
  h2{font-size:17px;margin:32px 0 10px;padding-top:16px;border-top:1px solid var(--trait)}
  h2:first-of-type{border-top:0;padding-top:0}
  p{margin:0 0 12px}
  ol,ul{margin:0 0 12px;padding-left:22px}
  li{margin:4px 0}
  code{background:var(--code);padding:1px 6px;border-radius:4px;
       font:14px/1.5 ui-monospace,"SF Mono",Menlo,Consolas,monospace}
  table{width:100%;border-collapse:collapse;margin:0 0 12px;font-size:15px}
  th,td{text-align:left;padding:7px 10px;border-bottom:1px solid var(--trait)}
  th{color:var(--doux);font-weight:600;font-size:13px;text-transform:uppercase;letter-spacing:.04em}
  td:first-child{white-space:nowrap;font:14px ui-monospace,Menlo,Consolas,monospace}
  .grille{display:grid;grid-template-columns:1fr 1fr;gap:1px;background:var(--trait);
          border:1px solid var(--trait);border-radius:8px;overflow:hidden;margin:0 0 12px}
  .case{background:var(--carte);padding:14px 16px}
  .case b{display:block;font-size:14px;margin-bottom:2px}
  .case span{color:var(--doux);font-size:14px}
  .note{background:var(--carte);border-left:3px solid var(--rouge);
        padding:12px 16px;border-radius:0 6px 6px 0;margin:0 0 12px}
  .note b{color:var(--rouge)}
</style>
</head>
<body>
<div class="page">

<h1>osmo-operator — démarrage rapide</h1>
<p class="sous">Réseau GSM/EGPRS complet sur une machine : BTS, cœur de réseau, Asterisk et softphones.</p>

<h2>1. Lancer</h2>
<p>Double-cliquez le <b>téléphone rouge</b> sur le bureau. Quatre fenêtres s'ouvrent, une par quart d'écran :</p>
<div class="grille">
  <div class="case"><b>Terminal</b><span>le banc démarre (<code>start-native.sh</code>)</span></div>
  <div class="case"><b>Firefox</b><span>console web sur <code>127.0.0.1:8080</code></span></div>
  <div class="case"><b>Wireshark</b><span>GSMTAP, <code>udp/4729</code></span></div>
  <div class="case"><b>Linphone</b><span>postes 100 et 200, déjà enregistrés</span></div>
</div>
<p>Comptez une à deux minutes avant que la radio soit stable.</p>

<h2>2. Passer un appel</h2>
<p>Dans Linphone, composez un numéro et décrochez :</p>
<table>
  <tr><th>Numéro</th><th>Destination</th></tr>
  <tr><td>100</td><td>Linphone A</td></tr>
  <tr><td>200</td><td>Linphone B</td></tr>
  <tr><td>600</td><td>test d'écho — votre voix vous revient</td></tr>
  <tr><td>500 / 700</td><td>annonces enregistrées</td></tr>
  <tr><td>1001<i>XX</i></td><td>un mobile GSM du réseau local</td></tr>
</table>
```

```
<p><b>Depuis un mobile GSM</b>, tout passe par le <b>VTY de la console web</b>, pas par un clavier : dans la barre
 ↪  en haut, choisissez l'opérateur (<code>OP1</code>), puis le mobile (<code>BB MS1&amp;nbsp;☎&amp;nbsp;…</code>), et
 ↪  cliquez <b>Go</b>. Vous obtenez son terminal, d'où l'appel se compose :</p>
<table>
  <tr><th>Commande VTY</th><th>Effet</th></tr>
  <tr><td>call 1 100</td><td>le mobile appelle Linphone A</td></tr>
  <tr><td>call 1 hangup</td><td>raccrocher</td></tr>
  <tr><td>call 1 answer</td><td>décrocher un appel entrant</td></tr>
</table>
<p>Le <code>1</code> est le nom du mobile, celui de <code>mobile.cfg</code>. En cas de doute, <code>list</code>
 ↪  dans le VTY affiche les commandes réellement disponibles.</p>
<p><b>Commencez par le 600.</b> S'il vous répond, la chaîne SIP et l'audio sont bonnes — inutile de chercher plus
 ↪  loin côté téléphone.</p>

<h2>3. Vérifier</h2>
<ul>
  <li>Les postes sont enregistrés : <code>asterisk -rx "pjsip show endpoints"</code> → <code>Avail</code></li>
  <li>Le mobile est attaché : la console web, page <i>Abonnés</i></li>
  <li>La radio parle : Wireshark défile en continu sur <code>udp/4729</code></li>
</ul>

<h2>4. Si ça coince</h2>
<div class="note">
  <p><b>Linphone ne s'enregistre pas.</b> Il ne relit jamais sa configuration à chaud : il faut le
   ↪  <b>redémarrer</b>. Un compte modifié pendant qu'il tourne n'est pas pris en compte, et il réécrit le fichier
   ↪  en quittant.</p>
</div>
<ul>
  <li><b>Appel muet.</b> Le micro passe par la carte son, pas par le navigateur : <code>pactl list short modules |
   ↪  grep loopback</code> doit montrer deux lignes.</li>
  <li><b>Écho.</b> Les enceintes rentrent dans le micro — mettez un casque.</li>
  <li><b>Pas de console web.</b> <code>systemctl status osmo-egprs-web</code>, puis <code>journalctl -u
   ↪  osmo-egprs-web</code>.</li>
</ul>

<h2>5. Arrêter</h2>
<p><code>sudo /opt/osmo-launch.sh stop</code>, ou <code>Ctrl+C</code> dans le terminal du banc.</p>

</div>
</body>
</html>
```

3.133 /opt/GSM/osmo-operator/data/desktop/claude.desktop

342 octets, 11 lignes → 11 lignes

```
[Desktop Entry]
Type=Application
Name=Claude
Name[fr]=Claude
Comment=Claude Code - l assistant IA en terminal
Comment[fr]=Claude Code - l assistant IA en terminal
Exec=/usr/local/bin/osmo-claude-anim
Icon=/usr/share/osmo-operator/icons/claude.svg
Terminal=false
Categories=Development;Utility;
Keywords=claude;ia;ai;assistant;code;anthropic;
```

3.134 /opt/GSM/osmo-operator/data/desktop/osmo-addition.desktop

366 octets, 11 lignes → 11 lignes

```
[Desktop Entry]
Type=Application
Name=Supplements
Name[fr]=Supplements
Comment=Docker container and SS7 multioperator - installe via build.sh
Comment[fr]=Docker container and SS7 multioperator - installe via build.sh
Exec=/usr/local/bin/osmo-addition-anim
Icon=system-software-install
Terminal=false
Categories=System;
Keywords=docker;supplement;multi;ss7;addition;
```

3.135 /opt/GSM/osmo-operator/data/desktop/osmo-dino.desktop

365 octets, 11 lignes → 11 lignes

```
[Desktop Entry]
Type=Application
Name=Dino (Coureur d'ombres)
Name[fr]=Dino (Coureur d'ombres)
Comment=Le jeu du dino : Soleil, Lune et ombres calculees
Comment[fr]=Le jeu du dino : Soleil, Lune et ombres calculees
Exec=/opt/GSM/osmo-operator/tools/osmo-dino.py --app
Icon=osmo-dino
Terminal=false
Categories=Game;ArcadeGame;
Keywords=dino;jeu;game;coureur;ombres;
```

3.136 /opt/GSM/osmo-operator/data/desktop/osmo-drivers.desktop

664 octets, 13 lignes → 13 lignes

```
[Desktop Entry]
Type=Application
Name=Pilotes graphiques
Name[fr]=Pilotes graphiques
Comment=Pilote NVIDIA : etat, installation, mise a jour (ubuntu-drivers)
Comment[fr]=Pilote NVIDIA : etat, installation, mise a jour (ubuntu-drivers)
Exec=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py --progress --root --title "Pilotes graphiques" --
 ↪  /opt/GSM/osmo-operator/tools/osmo-drivers.sh
Icon=preferences-desktop-display
# [2026-10-01] Le menu whiptail vit dans un terminal VTE DANS une fenetre GTK
# du banc (osmo-gtk-run --tty) : aucun favori n ouvre de terminal.
Terminal=false
Categories=System;Settings;HardwareSettings;
Keywords=nvidia;pilote;driver;ubuntu-drivers;
```

3.137 /opt/GSM/osmo-operator/data/desktop/osmo-info.desktop

261 octets, 8 lignes → 8 lignes

```
[Desktop Entry]
Type=Application
Name=Help - how to run the bench
Comment=Every bench module: icon, launcher bar and CLI, with the code behind it
Exec=/usr/local/bin/osmo-info
Icon=/usr/share/osmo-operator/icons/osmo-info.svg
Terminal=false
Categories=Utility;
```

3.138 /opt/GSM/osmo-operator/data/desktop/osmo-install.desktop

324 octets, 11 lignes → 11 lignes

```
[Desktop Entry]
Type=Application
Name=Installer osmo-operator
Name[fr]=Installer osmo-operator
Comment=Installer le banc GSM/EGPRS sur le disque
Comment[fr]=Installer le banc GSM/EGPRS sur le disque
Exec=/usr/local/bin/osmo-install
Icon=drive-harddisk
Terminal=false
Categories=System;
Keywords=install;installer;calamares;
```

3.139 /opt/GSM/osmo-operator/data/desktop/osmo-launch.desktop

2772 octets, 64 lignes → 64 lignes

```
[Desktop Entry]
Type=Application
Name=Lancer le banc GSM
Name[fr]=Lancer le banc GSM
Comment=Demarre le banc, Wireshark, Linphone et la console web
Comment[fr]=Demarre le banc, Wireshark, Linphone et la console web
# [2026-10-23] Le clic simple : la BTS classique (osmo-bts-trx <-> pont_dsp.py) et le mobile osmocom-bb.
Exec=/opt/GSM/osmo-operator/launch.sh --bts-classique --ms
Icon=/usr/share/osmo-operator/icons/osmo-launch.svg
Terminal=false
Categories=System;Network;
Keywords=gsm;osmocom;banc;lancer;

# ── ACTIONS DU CLIC DROIT ──────────────────────────────────────
# GNOME/DING lit `Actions=` et ajoute une entree par action au menu contextuel
# de l icone (bureau ET dock). launch.sh passe tout son "$@" a start.sh :
# les deux actions ci-dessous sont donc exactement les deux lignes de commande,
# sans duplication de logique.
#
# Les noms des groupes doivent correspondre EXACTEMENT aux jetons de `Actions=`,
# a la casse pres inclus : une entree qui ne correspond a rien est ignoree en
# silence, et le clic droit affiche simplement une action de moins.
Actions=Service;Dashboard;Console;Vty;Grgsm;Stop;

# « Comme un service » : systemctl, pas de terminal, et le banc survit a la
# fermeture de la session. C est launch.sh --service qui parle a systemd -
# l icone ne connait toujours qu une seule ligne de commande.
[Desktop Action Service]
Name=Demarrer le banc en service
Name[fr]=Demarrer le banc en service (sans terminal)

Exec=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py --progress --root --title "Banc GSM 2G" --icon
 ↪  /usr/share/osmo-operator/icons/osmo-launch.svg -- /opt/GSM/osmo-operator/launch.sh --service-suivi
 ↪  --bts-classique --ms

[Desktop Action Dashboard]
Name=Tableau de bord (127.0.0.1:8080)
Name[fr]=Tableau de bord (127.0.0.1:8080)

Exec=/opt/GSM/osmo-operator/launch.sh --dashboard

# La seule action qui ouvre un terminal, et pour cause : une session tmux est
# interactive, elle n a nulle part ailleurs ou s afficher.
```

```
[Desktop Action Console]
Name=Console du banc (tmux attach)
Name[fr]=Console du banc (tmux attach)

Exec=/opt/GSM/osmo-operator/launch.sh --console

[Desktop Action Vty]
Name=Console du mobile (VTY 4247)
Name[fr]=Console du mobile (VTY 4247)

Exec=/opt/GSM/osmo-operator/launch.sh --vty

# [2026-10-01] Le DSP (C54x) est le defaut de start.sh : un clic simple
# le lance. L action du clic droit est donc l AUTRE chaine, gr-gsm.
[Desktop Action Grgsm]
Name=Lancer en mode gr-gsm (sans DSP)
Name[fr]=Lancer en mode gr-gsm (sans DSP)
Exec=/opt/GSM/osmo-operator/launch.sh --grgsm

[Desktop Action Stop]
Name=Arreter le banc
Name[fr]=Arreter le banc
Exec=/opt/GSM/osmo-operator/launch.sh --stop
```

3.140 /opt/GSM/osmo-operator/data/desktop/osmo-lte.desktop

3256 octets, 57 lignes → 57 lignes

```
[Desktop Entry]
Type=Application
Name=4G du banc (srsRAN + Open5GS)
Name[fr]=4G du banc (srsRAN + Open5GS)
Comment=Un clic demarre la 4G (coeur Open5GS, eNB, UE), un second l arrete - osmo-lte toggle
Comment[fr]=Un clic demarre la 4G (coeur Open5GS, eNB, UE), un second l arrete - osmo-lte toggle
# [2026-09-08] LA 4G PAR L ICONE. Le service est services/osmo-lte.service ;
# l icone ne connait qu une commande, « osmo-lte toggle » (tools/osmo-lte.sh),
# qui demande root par pkexec et repond en notification. Pas de terminal :
# les Actions Etat et Journal en ouvrent un quand il faut lire.
# [2026-10-12] Tout passe par la fenetre de progression du banc (osmo-gtk-run.py
# --progress) : barre [n/4], etape, console repliee derriere un bouton. Zero shell.
Exec=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py --progress --title "4G du banc" --icon
 ↪  /usr/share/osmo-operator/icons/osmo-lte.svg -- /usr/local/bin/osmo-lte toggle
Icon=/usr/share/osmo-operator/icons/osmo-lte.svg
Terminal=false
Categories=System;Network;
Keywords=4g;lte;srsran;open5gs;epc;enb;ue;

Actions=Start;StartGui;Traces;Stop;Status;Journal;

[Desktop Action Start]
Name=Demarrer la 4G
Name[fr]=Demarrer la 4G
Exec=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py --progress --title "4G du banc" --icon
 ↪  /usr/share/osmo-operator/icons/osmo-lte.svg -- /usr/local/bin/osmo-lte up

# [2026-09-10] LE SECOND MODE : LA 4G AVEC SES TRACES. srsRAN est compile avec
# srsGUI ; « osmo-lte start-gui » ajoute --gui.enable sur l eNB et l UE et leur
# porte un DISPLAY (pkexec env DISPLAY=... XAUTHORITY=...). Ce n est PAS le
# service : une unite systeme n a pas d ecran. Dans un terminal, parce que le
# lancement parle pendant trente secondes (attente de l attach) et qu une
# fenetre Qt qui ne s ouvre pas doit pouvoir se lire.
[Desktop Action StartGui]
Name=Demarrer la 4G avec les traces (srsGUI)
Name[fr]=Demarrer la 4G avec les traces (srsGUI)
```

```
Exec=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py --progress --title "4G du banc (srsGUI)" --
 ↪  /usr/local/bin/osmo-lte start-gui

# [2026-10-12] Les traces SEULES : srsGUI vit dans srsenb/srsue (--gui.enable),
# donc relancer la radio est inevitable - mais le coeur Open5GS, lui, reste.
[Desktop Action Traces]
Name=Traces srsGUI seules (relance eNB et UE, garde le coeur)
Name[fr]=Traces srsGUI seules (relance eNB et UE, garde le coeur)
Exec=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py --progress --title "Traces srsGUI" -- /usr/local/bin/osmo-lte
 ↪  traces

[Desktop Action Stop]
Name=Arreter la 4G
Name[fr]=Arreter la 4G
Exec=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py --progress --close-on-exit --title "Arret de la 4G" --icon
 ↪  /usr/share/osmo-operator/icons/osmo-lte.svg -- /usr/local/bin/osmo-lte down

[Desktop Action Status]
Name=Etat (osmo-lte status, osmo-epc status)
Name[fr]=Etat (osmo-lte status, osmo-epc status)
Exec=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py --root --keep-open --title "Etat 4G" -- bash -c
 ↪  "/usr/local/bin/osmo-lte status; /usr/local/bin/osmo-epc status"

[Desktop Action Journal]
Name=Voir le journal (journalctl -u osmo-lte)
Name[fr]=Voir le journal (journalctl -u osmo-lte)
Exec=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py --root --title "journal osmo-lte" -- journalctl -u osmo-lte -f
 ↪  -n 200 --no-pager -o cat
```

3.141 /opt/GSM/osmo-operator/data/desktop/osmo-multi.desktop

2740 octets, 48 lignes → 48 lignes

```
[Desktop Entry]
Type=Application
Name=multi-operator
Name[fr]=multi-operator
Comment=SS7 entre l operateur natif, deux operateurs docker et l inter-STP
Comment[fr]=SS7 entre l operateur natif, deux operateurs docker et l inter-STP
# [2026-09-07] PAR launch.sh --multi, ET PLUS PAR UN systemctl NU.
# `pkexec systemctl start osmo-multi.service` rendait 0 SANS RIEN DEMARRER
# quand une ConditionPathExists de l unite manquait (/etc/osmocom/osmo-multi.conf,
# ecrit par addition.sh) : systemd saute l unite, ne la marque pas en echec, et
# le clic restait sans effet ni message. launch.sh --multi teste les conditions
# lui-meme, nomme ce qui manque, demarre l unite et annonce le resultat par
# notification (cf. le bloc « LE MULTI-OPERATEUR » dans launch.sh).
#
# [2026-09-04] PAR LE SERVICE, COMME LE MONO-OPERATEUR.
# L icone lancait start-multi.sh en direct (pkexec) : le banc multi partait
# alors hors de systemd, sa sortie vivait dans un terminal que personne ne
# relit, et quand un operateur ne demarrait pas il n en restait aucune trace -
# ni `systemctl status`, ni `journalctl -u`, juste des conteneurs manquants.
# Le mono-operateur, lui, est une unite (osmo-banc.service) depuis le debut.
# Les deux se lancent donc pareil : une unite, un journal, un etat, un arret
# propre. osmo-multi.service porte deja l environnement du banc (HOME, venv,
# OSMO_NO_TERM) et son ExecStop n arrete que les conteneurs et le hub - le
# natif reste a osmo-banc.
# [2026-10-12] En fenetre de progression (barre, etape, console repliee) : le
# journal de osmo-multi.service defile dedans (launch.sh --multi-suivi).
Exec=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py --progress --root --title "Multi-operateur" --icon
 ↪  /usr/share/osmo-operator/icons/osmo-multi.svg -- /opt/GSM/osmo-operator/launch.sh --multi-suivi
Icon=/usr/share/osmo-operator/icons/osmo-multi.svg
Terminal=false
Categories=System;Network;
```

```
Keywords=ss7;m3ua;multi;operateur;interstp;

Actions=Stop;Journal;

[Desktop Action Stop]
Name=Arreter le multi-operateur
Name[fr]=Arreter le multi-operateur
Exec=/opt/GSM/osmo-operator/launch.sh --multi-stop

# Le lancement peut durer plusieurs minutes (images, conteneurs, HLR) : sans un
# moyen de REGARDER, l icone parait ne rien faire. C est le pendant direct du
# passage en service - le journal existe, autant l ouvrir d un clic.
# [2026-10-01] Dans une fenetre GTK du banc (tools/osmo-gtk-run.py), plus dans
# un gnome-terminal : aucun favori n ouvre de terminal.
[Desktop Action Journal]
Name=Voir le journal (journalctl -u osmo-multi)
Name[fr]=Voir le journal (journalctl -u osmo-multi)
Exec=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py --root --title "journal osmo-multi" -- journalctl -u osmo-multi
 ↪  -f -n 200 --no-pager -o cat
```

3.142 /opt/GSM/osmo-operator/data/desktop/osmo-paint.desktop

400 octets, 11 lignes → 11 lignes

```
[Desktop Entry]
Type=Application
Name=Peinture (overlay)
Name[fr]=Peinture (overlay)
Comment=Dessin au premier plan par-dessus le bureau : feutre, formes, gomme
Comment[fr]=Dessin au premier plan par-dessus le bureau : feutre, formes, gomme
Exec=/opt/GSM/osmo-operator/tools/overlay-draw.py
Icon=osmo-paint
Terminal=false
Categories=Graphics;Utility;
Keywords=peinture;dessin;overlay;feutre;annoter;
```

3.143 /opt/GSM/osmo-operator/data/desktop/osmo-tutorial.desktop

380 octets, 11 lignes → 11 lignes

```
[Desktop Entry]
Type=Application
Name=Tutoriel - demarrage rapide
Name[fr]=Tutoriel - demarrage rapide
Comment=Prise en main du banc en cinq minutes
Comment[fr]=Prise en main du banc en cinq minutes
Exec=/usr/local/bin/osmo-tutorial
Icon=/usr/share/osmo-operator/icons/osmo-tutorial.svg
Terminal=false
Categories=System;Documentation;
Keywords=tutoriel;aide;demarrage;quickstart;
```

3.144 /opt/GSM/osmo-operator/data/desktop/osmo-update.desktop

321 octets, 11 lignes → 11 lignes

```
[Desktop Entry]
Type=Application
Name=Update osmo-operator
Name[fr]=Mise a jour osmo-operator
Comment=Rejouer update.sh du depot osmo-operator
Comment[fr]=Rejouer update.sh du depot osmo-operator
Exec=/usr/local/bin/osmo-update-anim
Icon=system-software-update
Terminal=false
Categories=System;
Keywords=update;sms;osmo;
```

3.145 /opt/GSM/osmo-operator/environment/load.env

949 octets, 23 lignes → 23 lignes

```
# ============================================================================
#  environment/load.env — chargeur de configuration
# ============================================================================
#  Point d'entree unique. Source par run.sh avec `set -a`, donc tout est exporte.
#
#  ORDRE — il compte. Tous les fichiers utilisent `:=` (« si pas deja defini »),
#  donc LE PREMIER QUI POSE UNE VALEUR GAGNE :
#      1. la ligne de commande     VAR=x ./run.sh     <- gagne toujours
#      2. modes.env                le profil choisi
#      3. paths.env                ou sont les dependances
#      4. les fichiers par domaine
# ----------------------------------------------------------------------------

_ENVDIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"

. "$_ENVDIR/modes.env"
. "$_ENVDIR/paths.env"

for _f in coeur radio reseau debug; do
    [ -f "$_ENVDIR/$_f.env" ] && . "$_ENVDIR/$_f.env"
done

unset _f _ENVDIR
```

3.146 /opt/GSM/osmo-operator/environment/modes.env

1617 octets, 44 lignes → 44 lignes

```
# ============================================================================
#  environment/modes.env — profils prets a l'emploi
# ============================================================================
#  Reprend les modes historiques de start-native.sh, convertis en profils du
#  moteur. Choisissez-en UN :
#      CALYPSO_PROFILE=faketrx ./run.sh
#  ou, par compatibilite avec l'ancienne interface :
#      MODE=faketrx ./start-native.sh
# ----------------------------------------------------------------------------

# MODE (historique) a la priorite s'il est pose ; sinon CALYPSO_PROFILE.
: "${CALYPSO_PROFILE:=${MODE:-core}}"

case "$CALYPSO_PROFILE" in

  # Coeur seul : STP + HLR + MSC + MGW + BSC (+ Asterisk). Aucune radio.
  core|noproc)      CALYPSO_PROFILE=core ;;
```

```
  # Coeur + osmo-bts + fake_trx + trxcon + mobile. Pas de QEMU.
  faketrx)          CALYPSO_PROFILE=faketrx ;;

  # Coeur + osmo-bts-virtual + virtphy + mobile.
  virtphy)          CALYPSO_PROFILE=virtphy ;;

  # Coeur en no-process + l'emulation Calypso (deleguee a OQC_ROOT).
  qemu|calypso)     CALYPSO_PROFILE=calypso ;;

  # Coeur + fake_trx vivant + Calypso attache.
  faketrx-qemu|hybrid) CALYPSO_PROFILE=hybrid ;;

  # SDR physique.
  hw)               CALYPSO_PROFILE=hw ;;

  # Multi-operateurs SS7 par ip netns.
  virtual)          CALYPSO_PROFILE=virtual ;;

  *)
      echo "[environment/modes.env] profil inconnu : '$CALYPSO_PROFILE'" >&2
      echo "   core | faketrx | virtphy | calypso | hybrid | hw | virtual" >&2
      ;;
esac

# Nombre d'operateurs (profil « virtual » uniquement).
: "${N_OPERATORS:=1}"
```

3.147 /opt/GSM/osmo-operator/environment/paths.env

8019 octets, 150 lignes → 150 lignes

```
# ============================================================================
#  environment/paths.env — ou sont les dependances (configuration d'INSTALLATION)
# ============================================================================
#  Repond a une seule question : OU sont les choses sur CETTE machine.
#  Surchargeable depuis l'environnement (idiome `:=`) :
#      GSM_ROOT=/home/moi/gsm ./run.sh
#  Verifier :   ./run.sh --check-paths
# ----------------------------------------------------------------------------

# --- Racine de l'installation -----------------------------------------------
: "${GSM_ROOT:=/opt/GSM}"

# --- Ce depot ---------------------------------------------------------------
: "${NITB_TREE:=$(cd "$(dirname "${BASH_SOURCE[0]}")/.." && pwd)}"
: "${NITB_SCRIPTS:=$NITB_TREE/scripts}"
: "${NITB_CONFIGS:=$NITB_TREE/configs}"
: "${NITB_HELPERS:=$NITB_TREE/helpers}"

# --- L'emulation Calypso (depot voisin) -------------------------------------
# Le mode « qemu » lui delegue. [2026-09-25] Un seul arbre QEMU, /opt/GSM/qosmo
# (run.sh, run_modules, cfgs, environnement repris des anciens forks
# qosmo-grgsm/qosmo-dsp), et deux couches 1 :
#
#   grgsm_exe    DEFAUT. Couche 1 gr-gsm dans QEMU (qosmo --enable-l1-grgsm) et
#                pont grgsm_exe. C'est la pile qui va de bout en bout : le mobile
#                campe, fait sa LU, s'authentifie en COMP128v1, chiffre en A5/1,
#                echange des SMS et tient un appel voix.
#   c54x_exe     Le DSP TMS320C54x tourne HORS de QEMU (/opt/GSM/c54x_exe) sur la
#                vraie mask-ROM TI (ROM : /opt/GSM/calypso_dsp.*.bin, copie dans
#                c54x_exe/rom) ; QEMU tourne avec CALYPSO_DSP_EXTERN=1.
#
# Choisir :  ./start-native.sh --dsp        (c54x_exe ; --grgsm pour le defaut)
# Ou pointer directement un arbre :  OQC_ROOT=/chemin/vers/qosmo ./start-native.sh
```

```
: "${CALYPSO_FORK:=qosmo}"

# Cherche a cote (depot voisin), puis a l'emplacement d'installation.
if [ -x "$NITB_TREE/../$CALYPSO_FORK/run.sh" ]; then
    : "${OQC_ROOT:=$(cd "$NITB_TREE/../$CALYPSO_FORK" && pwd)}"
else
    : "${OQC_ROOT:=$GSM_ROOT/$CALYPSO_FORK}"
fi

# --- Binaires Osmocom ------------------------------------------------------
# Vides = cherches dans le PATH, ce qui est le cas d'une installation par paquets.
: "${OSMO_STP:=osmo-stp}"
: "${OSMO_HLR:=osmo-hlr}"
: "${OSMO_MSC:=osmo-msc}"
: "${OSMO_MGW:=osmo-mgw}"
: "${OSMO_BSC:=osmo-bsc}"
: "${OSMO_SGSN:=osmo-sgsn}"
: "${OSMO_GGSN:=osmo-ggsn}"
: "${OSMO_PCU:=osmo-pcu}"
: "${OSMO_BTS_TRX:=osmo-bts-trx}"
: "${OSMO_BTS_VIRTUAL:=osmo-bts-virtual}"
: "${OSMO_TRX_IPC:=osmo-trx-ipc}"
: "${ASTERISK_BIN:=asterisk}"

# --- osmocom-bb (mobile, trxcon, virtphy, fake_trx) ---------------------------
: "${OSMOCOM_BB:=$GSM_ROOT/osmocom-bb}"
: "${MOBILE_BIN:=$OSMOCOM_BB/src/host/layer23/src/mobile/mobile}"
: "${TRXCON_BIN:=$OSMOCOM_BB/src/host/trxcon/trxcon}"
: "${VIRTPHY_BIN:=$OSMOCOM_BB/src/host/virt_phy/src/virtphy}"
: "${FAKE_TRX:=$OSMOCOM_BB/src/target/trx_toolkit/fake_trx.py}"

# --- Configuration Osmocom -------------------------------------------------
: "${OSMOCOM_CFG:=/etc/osmocom}"
: "${OSMOCOM_HOME:=$HOME/.osmocom}"

# --- Sorties ---------------------------------------------------------------
# Un tiers n'a aucune raison d'ecrire dans /root : RUN_DIR (socket du moniteur,
# fichiers PID — quelques octets, et plusieurs outils les cherchent la) reste
# sous /tmp.
#
# [2026-08-31] RUN_DIR NE DOIT PAS DEPENDRE D'UNE VARIABLE QUE sudo EFFACE.
# L'ecriture precedente, `${XDG_RUNTIME_DIR:-/tmp}/osmo-nitb`, a l'air anodine.
# En pratique elle donne DEUX identites de banc selon la facon de lancer :
#
#   sudo -i puis ./start-native.sh    XDG_RUNTIME_DIR=/run/user/0
#                                     -> RUN_DIR=/run/user/0/osmo-nitb
#   sudo ./launch/start-multi.sh  (l'icone)  sudo/env_reset EFFACE XDG_RUNTIME_DIR
#                                     -> RUN_DIR=/tmp/osmo-nitb
#
# Or RUN_DIR est l'endroit ou vivent les *.pid, donc la SEULE memoire qu'a le
# banc de ce qui tourne deja. Un second lancement qui ne regarde pas le meme
# repertoire ne VOIT PAS la pile en place, la croit absente, et en empile une
# seconde par-dessus. Les deux se disputent alors les ressources GLOBALES, qui
# elles ne bougent pas avec RUN_DIR : /tmp/osmocom_l2, /tmp/ms2_l2, l'UDP
# 5720/6720, le gdb 1234. Mesure sur le banc, quatre generations vivantes en
# meme temps (18:38, 18:40, 18:43, 18:50) et deux serveurs en LISTEN sur
# /tmp/osmocom_l2 - de la venaient les "Layer2 socket failed" en boucle.
#
# On resout donc le repertoire depuis l'UID REEL, qui lui survit a sudo, et on
# ne retombe sur /tmp que si /run/user/<uid> n'existe pas. Le `:=` d'origine est
# conserve : un RUN_DIR impose par l'appelant reste prioritaire.
if [ -z "${RUN_DIR:-}" ]; then
    _oq_rt="${XDG_RUNTIME_DIR:-}"
    { [ -n "$_oq_rt" ] && [ -d "$_oq_rt" ]; } || _oq_rt="/run/user/$(id -u)"
    [ -d "$_oq_rt" ] || _oq_rt="/tmp"
    RUN_DIR="${_oq_rt}/osmo-nitb"
```

```
    unset _oq_rt
fi

# [2026-08-03] LES JOURNAUX RESTENT SUR LE TMPFS. Arbitrage entre DEUX pannes
# réelles, pas une préférence — les deux ont été vécues le même jour.
#
#   Panne A — /tmp saturé (30/07 deux fois, puis 03/08). Le tmpfs faisait 512 Mo,
#   partagés avec le /tmp système. Plein, il fait échouer les compilations
#   (« No space left on device » sur le .s du compilateur) et perdre des journaux.
#   CORRIGÉE À LA SOURCE : start-docker.sh crée désormais le conteneur avec un /tmp de
#   2 Go (--tmpfs /tmp:…size=2g), au lieu de déplacer les journaux.
#
#   Panne B — défilement tmux saccadé. Provoquée par la tentative de contournement
#   de la panne A : mettre LOG_DIR sous /root. Identifiée par bissect (commit
#   1d241a0) puis confirmée par retour arrière. Mécanisme probable : / est un
#   overlay à 16 couches, où la propagation inotify vers la couche supérieure est
#   défaillante ; `tail -F` retombe alors sur du sondage à 1 s et le flux
#   (~490 ko/s) arrive en paquets d'une seconde. tmux avale un gros bloc d'un
#   coup, à intervalle régulier — d'où la sensation de rame.
#
# MORALE : les journaux vivent sur le tmpfs, parce que tmux les suit en direct et
# que le tmpfs est le seul système de fichiers ici où inotify fonctionne. Si le
# volume redevient un problème, on tarit la SOURCE, on ne déplace pas la cible :
#   CALYPSO_INTM_TRANS=0   retire ~56 % du flux qemu.log
#   CALYPSO_MAILBOX=0      retire mailbox.log (~230 ko/s, soit ~800 Mo/h — c'est
#                          lui qui remplissait les 512 Mo à lui seul)
#
# Surchargeable comme tout le reste : LOG_DIR=/ailleurs ./start-native.sh
: "${LOG_DIR:=$RUN_DIR/logs}"

# --- Pont de nommage avec le depot Calypso -----------------------------------
# Les modules partages entre les deux depots designent l emulation Calypso par
# QEMU_TREE (son nom vu de l interieur). Ici, ce depot est le VOISIN : OQC_ROOT.
# On expose les deux noms pour que les memes modules fonctionnent des deux cotes
# sans etre dupliques.
: "${QEMU_TREE:=$OQC_ROOT}"
: "${QEMU_BIN:=$OQC_ROOT/build/qemu-system-arm}"
# Lanceur C du fork (tools/qosmo-launch -> /usr/local/bin/<fork>). C'est lui que
# 40-qemu.sh appelle a la place de qemu-system-arm ; il retombe sur QEMU_BIN
# s'il est absent. Voir start-native.sh « Lanceur QEMU » et `qosmo --help`.
: "${QOSMO_LAUNCHER:=/usr/local/bin/$CALYPSO_FORK}"

# --- Chemins de l emulation Calypso (firmware, ROM DSP, osmocon...) ----------
# Les modules du mode « qemu » sont partages avec le depot voisin et exigent
# FIRMWARE_ELF, DSP_PROM0..3, OSMOCON, QEMU_CFGS... Ces chemins sont decrits par
# le paths.env de ce voisin : on le source plutot que de les redupliquer ici.
# En DERNIER, a dessein : tout ce qui precede a deja pose sa valeur, et l idiome
# `:=` fait que le premier gagne — RUN_DIR/LOG_DIR restent donc ceux du reseau.
if [ -r "$OQC_ROOT/environnement/paths.env" ]; then
    . "$OQC_ROOT/environnement/paths.env"
fi
```

3.148 /opt/GSM/osmo-operator/fft-web/fft_web.py

11021 octets, 206 lignes → 206 lignes

```
#!/usr/bin/env python3
# fft_web.py — 2 FFT (MS + BTS) sur UNE page, serveur web autonome (port 8081).
#
# Reprend la logique de tools/fft.sh / tools/fft2.sh (PSD Welch sur l'I/Q fc32) mais :
#   - NATIF : lit directement les sources de l'hôte, pas de docker exec.
#   - HEADLESS : pas de matplotlib/X. numpy calcule la PSD → JSON → le navigateur dessine (canvas).
#   - LIVE FIFO : le spectre BTS lit /tmp/iq_fft.fifo EN CONTINU (le tap FFT
#     dédié du relay DL osmo-trx) au lieu du ring record.cfile (128 Mo) qui
```

```
#     FIGEAIT une fois plein. La FIFO est ouverte O_RDWR|O_NONBLOCK (comme
#     grgsm_fft_live.py) : pas d'EOF, le write non-bloquant du relay trouve
#     toujours un lecteur, et on ne perturbe pas la boucle relay / le camping.
#
#   - MS  = iq_fft_ms.fifo (montant, ecrit par pont.py ; LIVE)
#   - BTS = iq_fft.fifo    (DL relay osmo-trx, LIVE)
#   Chaque source accepte indifféremment un .cfile (lecture tail) OU un .fifo
#   (lecture live) : auto-détecté. Override par env CFILE_MS / CFILE_BTS.
#
# Lancement :  python3 /opt/GSM/osmo-nitb-for-calypso/fft-web/fft_web.py
#   FFT_WEB_PORT=8081 RATE=1083333 NSAMP=262144 NFFT=4096 ...  (overridables par env)
import os, json, stat, threading
from http.server import BaseHTTPRequestHandler, ThreadingHTTPServer
from urllib.parse import urlparse, parse_qs
import numpy as np

# --- racine de l'installation, resolue sans chemin en dur --------------------
# GSM_ROOT si l'environnement le pose (c'est le cas quand on passe par run.sh),
# sinon le parent du depot : les deux depots vivent cote a cote.
import os as _os
GSM_ROOT = _os.environ.get(
    "GSM_ROOT",
    _os.path.dirname(_os.path.dirname(_os.path.abspath(__file__))))


PORT  = int(os.environ.get('FFT_WEB_PORT', '8081'))
RATE  = float(os.environ.get('RATE', '1083333'))   # 4 SPS natif = 26e6/24 (= Fs du relay/FIFO)
NSAMP = int(os.environ.get('NSAMP', '262144'))     # complex samples gardés (tail / fenêtre live)
NFFT  = int(os.environ.get('NFFT', '4096'))        # résolution FFT (segment Welch)
# Cadence d'affichage : 40 ms = 25 img/s. Le navigateur poll /psd à cet intervalle
# (2 fetch ms+bts par tick → ~50 req/s). On borne le nb de segments Welch par
# requête (NSEG_MAX) pour que le calcul tienne la cadence (NFFT*NSEG_MAX FFT/req).
REFRESH_MS = int(os.environ.get('REFRESH_MS', '40'))
NSEG_MAX   = int(os.environ.get('NSEG_MAX', '16'))

SRC = {
    # [2026-08-29] Le montant vient de pont.py (PONT_AIRREC_UL_FIFO), pas du
    # cfile DSP : en mode pont, /dev/shm/dsp_iq.cfile n'a aucun producteur et
    # le spectre MS restait vide - une seule FFT vivante sur les deux.
    'ms':  {'path': os.environ.get('CFILE_MS',  '/tmp/iq_fft_ms.fifo'),
            'arfcn': os.environ.get('ARFCN_MS', '514'),  'label': 'MS — montant (iq_fft_ms.fifo)'},
    'bts': {'path': os.environ.get('CFILE_BTS', '/tmp/iq_fft.fifo'),
            'arfcn': os.environ.get('ARFCN_BTS', '514'), 'label': 'BTS — DL relay LIVE (iq_fft.fifo)'},
}

# ── État live par source FIFO : fd O_RDWR|O_NONBLOCK + buffer roulant ────────
_state = {}                                        # src -> {'fd':int, 'buf':bytearray}
_locks = {k: threading.Lock() for k in SRC}
MAXB   = NSAMP * 8                                 # octets gardés (complex64 = 8 o/échantillon)

def _is_fifo(path):
    try:
        return stat.S_ISFIFO(os.stat(path).st_mode)
    except OSError:
        return path.endswith('.fifo')              # pas encore créé : on se fie au suffixe

def read_live_iq(src, path):
    """Draine la FIFO en non-bloquant et renvoie les NSAMP derniers échantillons
    (le plus frais). O_RDWR => jamais d'EOF, jamais de blocage du relay."""
    st = _state.get(src)
    if st is None:
        if not os.path.exists(path):
            os.mkfifo(path, 0o666)
        fd = os.open(path, os.O_RDWR | os.O_NONBLOCK)
        st = {'fd': fd, 'buf': bytearray()}
        _state[src] = st
```

```
    fd, buf = st['fd'], st['buf']
    while True:
        try:
            c = os.read(fd, 1 << 16)
        except BlockingIOError:
            break
        if not c:
            break
        buf += c
        if len(buf) > MAXB:
            del buf[:-MAXB]                          # borne la RAM, garde le frais
    if len(buf) > MAXB:
        del buf[:-MAXB]
    n = (len(buf) // 8) * 8
    if n < NFFT * 8:
        return np.array([], dtype=np.complex64)
    return np.frombuffer(bytes(buf[-n:]), dtype=np.complex64)

def read_tail_iq(path, nsamp):
    sz = os.path.getsize(path)
    nbytes = min(sz - (sz % 8), nsamp * 8)         # complex64 = 8 octets/échantillon
    with open(path, 'rb') as f:
        if sz > nbytes:
            f.seek(sz - nbytes)
        raw = f.read(nbytes)
    return np.frombuffer(raw, dtype=np.complex64)

def welch_psd(iq, nfft, rate):
    if iq.size < nfft:
        return None, None
    win = np.hanning(nfft).astype(np.float32)
    nseg = min(iq.size // nfft, NSEG_MAX)          # borne le coût pour tenir 25 img/s
    iq = iq[-nseg*nfft:]                            # garde les segments les plus frais
    acc = np.zeros(nfft, dtype=np.float64)
    for i in range(nseg):
        seg = iq[i*nfft:(i+1)*nfft] * win
        acc += np.abs(np.fft.fftshift(np.fft.fft(seg)))**2
    acc /= nseg
    psd_db = 10.0*np.log10(acc + 1e-12)
    freqs = np.fft.fftshift(np.fft.fftfreq(nfft, 1.0/rate)) / 1e3   # kHz
    return freqs, psd_db

def psd_json(src):
    s = SRC[src]
    path = s['path']
    try:
        with _locks[src]:
            iq = read_live_iq(src, path) if _is_fifo(path) else read_tail_iq(path, NSAMP)
        f, p = welch_psd(iq, NFFT, RATE)
        if f is None:
            return {'error': "flux pas encore prêt (pas assez d'échantillons)", 'label': s['label']}
        step = max(1, len(f)//1024)                # sous-échantillonne pour le transport
        return {'label': s['label'], 'arfcn': s['arfcn'], 'rate': RATE,
                'freqs': [round(x, 1) for x in f[::step].tolist()],
                'psd':   [round(x, 2) for x in p[::step].tolist()]}
    except FileNotFoundError:
        return {'error': 'source absente (%s) — lance la stack' % path, 'label': s['label']}
    except Exception as e:
        return {'error': str(e), 'label': s['label']}

PAGE = """<!doctype html><html><head><meta charset=utf-8><title>Calypso FFT — MS & BTS</title>
<style>body{background:#0b0f14;color:#cdd6e0;font-family:monospace;margin:0;padding:12px}
h1{font-size:15px;color:#2aa198;margin:0 0 10px} .row{display:flex;gap:12px;flex-wrap:wrap}
.card{flex:1;min-width:380px;background:#0e151c;border:1px solid #1d2a33;border-radius:8px;padding:8px}
.t{font-size:12px;color:#b58900;margin-bottom:4px}
 ↪  canvas{width:100%;background:#060a0e;border-radius:4px;display:block}
```

```
.psd{height:200px} .wf{height:150px;margin-top:6px;image-rendering:pixelated}
.err{color:#dc322f;font-size:12px;min-height:14px}</style></head><body>
<h1>\U0001F4E1 Calypso I/Q FFT — MS vs BTS <span id=st style="color:#586e75;font-size:11px"></span></h1>
<div class=row>
 <div class=card><div class=t id=t-ms>MS</div><canvas id=c-ms class=psd></canvas><canvas id=w-ms
 ↪  class=wf></canvas><div class=err id=e-ms></div></div>
 <div class=card><div class=t id=t-bts>BTS</div><canvas id=c-bts class=psd></canvas><canvas id=w-bts
 ↪  class=wf></canvas><div class=err id=e-bts></div></div>
</div>
<script>
function cmap(v){v=Math.max(0,Math.min(1,v));var h=(1-v)*240,l=0.12+0.5*v,a=1*Math.min(l,1-l);
 var f=function(n){var k=(n+h/30)%12;return Math.round(255*(l-a*Math.max(-1,Math.min(k-3,9-k,1))));};
 return [f(0),f(8),f(4)];}
function drawPsd(src,d){var c=document.getElementById('c-'+src),x=c.getContext('2d');
 var W=c.width=c.clientWidth*2,H=c.height=c.clientHeight*2;x.clearRect(0,0,W,H);
 var p=d.psd,n=p.length;if(!n)return;var mn=Math.min.apply(null,p),mx=Math.max.apply(null,p),rg=(mx-mn)||1;
 x.strokeStyle='#16222b';x.lineWidth=1;for(var g=0;g<=4;g++){var
 ↪  yy=H*g/4;x.beginPath();x.moveTo(0,yy);x.lineTo(W,yy);x.stroke();}
 x.beginPath();x.moveTo(W/2,0);x.lineTo(W/2,H);x.strokeStyle='#243845';x.stroke();
 x.strokeStyle='#2aa198';x.lineWidth=2;x.beginPath();
 for(var i=0;i<n;i++){var xx=W*i/(n-1),yy=H-(p[i]-mn)/rg*H*0.9-H*0.05;i?x.lineTo(xx,yy):x.moveTo(xx,yy);}
 x.stroke();}
function drawWf(src,d){var c=document.getElementById('w-'+src),x=c.getContext('2d');
 var p=d.psd,n=p.length;if(!n)return;
 if(c.width!==n){c.width=n;c.height=160;x.fillStyle='#060a0e';x.fillRect(0,0,n,c.height);}
 // [2026-08-09] SENS DE DEFILEMENT REMIS A L'ENDROIT. C'etait drawImage(c,0,-1)
 // + putImageData(...,c.height-1) : le plus recent en BAS, l'historique qui
 // monte. Toutes les cascades SDR (gqrx, GNU Radio, grgsm) font l'inverse — le
 // plus recent EN HAUT, l'historique qui descend — et l'oeil lit le temps de
 // haut en bas. On decale donc l'image vers le BAS et on ecrit la ligne neuve
 // sur la PREMIERE ligne.
 x.drawImage(c,0,1);                        // scroll vers le bas d'1px
 var mn=Math.min.apply(null,p),mx=Math.max.apply(null,p),rg=(mx-mn)||1;
 var row=x.createImageData(n,1);
 for(var i=0;i<n;i++){var
 ↪  col=cmap((p[i]-mn)/rg),o=i*4;row.data[o]=col[0];row.data[o+1]=col[1];row.data[o+2]=col[2];row.data[o+3]=255;}
 x.putImageData(row,0,0);}                  // nouvelle ligne en haut
function draw(src,d){var e=document.getElementById('e-'+src),tt=document.getElementById('t-'+src);
 if(d.error){e.textContent=d.error;tt.textContent=d.label||'';return;} e.textContent='';
 tt.textContent=d.label+'  •  ARFCN '+d.arfcn+'  •  Fs '+(d.rate/1e6).toFixed(3)+' MHz';
 drawPsd(src,d);drawWf(src,d);}
function tick(){['ms','bts'].forEach(function(s){
  fetch('/psd?src='+s+'&t='+Date.now()).then(function(r){return
 ↪  r.json();}).then(function(d){draw(s,d);}).catch(function(){});});
 document.getElementById('st').textContent='— '+new Date().toLocaleTimeString();}
tick();setInterval(tick,__REFRESH_MS__);
</script></body></html>"""

class Handler(BaseHTTPRequestHandler):
    def log_message(self, *a):
        pass
    def do_GET(self):
        u = urlparse(self.path)
        if u.path == '/psd':
            q = parse_qs(u.query); src = q.get('src', ['ms'])[0]
            if src not in SRC:
                src = 'ms'
            body = json.dumps(psd_json(src)).encode()
            self.send_response(200)
            self.send_header('Content-Type', 'application/json')
            self.send_header('Cache-Control', 'no-store')
            self.send_header('Content-Length', str(len(body)))
            self.end_headers(); self.wfile.write(body); return
        body = PAGE.replace('__REFRESH_MS__', str(REFRESH_MS)).encode('utf-8')
        self.send_response(200)
        self.send_header('Content-Type', 'text/html; charset=utf-8')
```

```
        self.send_header('Content-Length', str(len(body)))
        self.end_headers(); self.wfile.write(body)

if __name__ == '__main__':
    print('FFT web on :%d  MS=%s  BTS=%s' % (PORT, SRC['ms']['path'], SRC['bts']['path']))
    ThreadingHTTPServer(('0.0.0.0', PORT), Handler).serve_forever()
```

3.149 /opt/GSM/osmo-operator/helpers/create_interop.sh

7249 octets, 199 lignes → 199 lignes

```
#!/bin/bash
# create_interop.sh - Genere la configuration inter-STP (hub SS7 central)
#
# Parametres :
#   $1  n_operators   Nombre d'operateurs (1..24)
#   $2  outfile       Chemin fichier config de sortie (defaut: osmo-stp-interop.cfg)
#
# Point codes ITU 3-8-3 :
#   Inter-STP : 0.0.0
#   Op N STP  : 1.N.2    (cluster = N, 8 bits → max 255, limite a 24)
#   Op N MSC  : 1.N.1
#   Op N BSC  : 1.N.3
#
# Routes statiques avec masque exact 7.255.7 (14 bits) :
#   1.N.1 → as-opN  (MSC)
#   1.N.3 → as-opN  (BSC)

set -e

# ── Mode WAN ──────────────────────────────────────────────────────────
#   create_interop.sh --wan <n_noeuds> <ops_par_noeud> <outfile>
#   WAN_NODE_IDS="1 3" : les numeros des noeuds (defaut 1..n), table non contigue
#
# Le hub d'origine ne dessert qu'UNE machine : ses point codes 1.<op>.<role> se
# recalculent a l'identique sur chaque noeud d'un WAN, donc trois noeuds relies
# au meme hub presenteraient trois fois 1.1.2. Un point code est une ADRESSE :
# la collision n'est pas un detail de nommage, elle rend le routage SS7 faux.
#
# Le plan WAN encode le noeud DANS le point code :
#   PC = 1.<noeud><op>.<role>     noeud 1 op 1 → 1.11.2 ; noeud 3 op 2 → 1.32.2
#   RCTX = noeud*1000 + op*100 + 50
# Lisible a l'oeil (le premier chiffre est le noeud), unique, et le champ
# central du 3-8-3 tient jusqu'a 9 noeuds × 9 operateurs.
# ── Adresse d'ecoute du hub ──────────────────────────────────────────
# Ce fichier est REGENERE a chaque demarrage du hub (start-interstp.sh). Tant
# que l'adresse etait ecrite 0.0.0.0 en dur ici, toute correction posee ailleurs
# - set_stp_ip.sh en particulier - etait effacee au redemarrage suivant, sans
# que rien ne le signale : le hub repartait en multi-homing.
#
# Et 0.0.0.0 n'est pas neutre : SCTP annonce alors TOUTES les adresses de la
# machine dans son INIT (NAT 10.0.2.x, alias 172.20.x, host-only 192.168.56.x).
# Le pair essaie des chemins qui, vus de lui, ne menent nulle part ; l'ASP monte
# puis retombe, en boucle. Une adresse unique et juste vaut mieux qu'un
# multi-homing dont trois branches sur quatre sont mortes.
#
# HUB_LISTEN_IP (ou --listen-ip) impose l'adresse. Vide = 0.0.0.0, l'ancien
# comportement, conserve pour les montages d'une seule machine.
HUB_LISTEN_IP="${HUB_LISTEN_IP:-0.0.0.0}"
_ci_args=()
while [ $# -gt 0 ]; do
    case "$1" in
        --listen-ip)   HUB_LISTEN_IP="${2:-0.0.0.0}"; shift 2 ;;
        --listen-ip=*) HUB_LISTEN_IP="${1#*=}"; shift ;;
```

```
        *)              _ci_args+=("$1"); shift ;;
    esac
done
[ "${#_ci_args[@]}" -gt 0 ] && set -- "${_ci_args[@]}"
[ -n "$HUB_LISTEN_IP" ] || HUB_LISTEN_IP="0.0.0.0"

WAN_MODE=0
if [ "${1:-}" = "--wan" ]; then
    WAN_MODE=1
    n_nodes="${2:-3}"
    ops_per_node="${3:-1}"
    outfile="${4:-osmo-stp-interop.cfg}"
    if ! [[ "$n_nodes" =~ ^[1-9]$ ]] || ! [[ "$ops_per_node" =~ ^[1-9]$ ]]; then
        echo "Erreur : --wan <1-9 noeuds> <1-9 operateurs>" >&2; exit 1
    fi
else
    n_operators="${1:-2}"
    outfile="${2:-osmo-stp-interop.cfg}"
    if ! [[ "$n_operators" =~ ^[0-9]+$ ]] || [ "$n_operators" -lt 1 ] || [ "$n_operators" -gt 24 ]; then
        echo "Erreur : n_operators doit etre 1..24" >&2
        exit 1
    fi
fi

cat > "$outfile" <<'EOFCONFIG'
!
! osmo-stp-interop.cfg - Configuration inter-STP centrale
!
! PC 0.0.0 : hub de routage SS7
!

log stderr
 logging filter all 1
 logging color 1
 logging print category-hex 0
 logging print category 1
 logging print extended-timestamp 1
 logging print level 1
 logging print file basename
 logging level lss7 info
 logging level lsccp info
 logging level lm3ua info
 logging level linp notice
!
stats interval 5
!
line vty
 no login
!
cs7 instance 0
 network-indicator international
 point-code 0.0.0
!
 xua rkm routing-key-allocation dynamic-permitted

 listen m3ua 2908
  accept-asp-connections dynamic-permitted
  local-ip __LISTEN_IP__
!
EOFCONFIG

# Heredoc quote : le gabarit sort tel quel, on substitue ensuite. Le marqueur
# n'existe que dans l'entete, la boucle WAN qui suit n'en pose pas.
sed -i "s|__LISTEN_IP__|${HUB_LISTEN_IP}|" "$outfile"

if [ "$WAN_MODE" = "1" ]; then
```

```
    for n in ${WAN_NODE_IDS:-$(seq 1 "$n_nodes")}; do
        for o in $(seq 1 "$ops_per_node"); do
            rctx_inter=$(( n * 1000 + o * 100 + 50 ))
            pc_stp="1.${n}${o}.2"
            cat >> "$outfile" <<EOF

 as as-n${n}op${o} m3ua
  routing-key ${rctx_inter} ${pc_stp}
  traffic-mode override
EOF
        done
    done
else
# [2026-08-31] DE 1 AU RANG LE PLUS HAUT, pas "1 a n_operators".
# n_operators compte les CONTENEURS ; il ne dit rien du natif, qui est un
# operateur du banc au meme titre. Avec OP_ID_BASE=2 - conteneurs en 2 et 3,
# natif en 1 - la boucle produisait as-op1 et as-op2 pour DEUX conteneurs
# qui sont en realite op2 et op3 : le hub declarait donc un as-op2 en 1.2.2
# que personne ne venait prendre, et AUCUN as-op3. D ou, cote natif,
#     1.2.2/14 INACC ... PROHIB   1.3.2/14 INACC ... PROHIB
# et un banc ou les operateurs sont attaches mais ne se routent pas.
for i in $(seq 1 "$(( ${OP_ID_BASE:-1} + n_operators - 1 ))"); do
    rctx_inter=$(( i * 100 + 50 ))
    pc_stp="1.${i}.2"

    cat >> "$outfile" <<EOF

 as as-op${i} m3ua
  routing-key ${rctx_inter} ${pc_stp}
  traffic-mode override
EOF
done
fi

cat >> "$outfile" <<'EOFROUTES'

 route-table system
EOFROUTES

if [ "$WAN_MODE" = "1" ]; then
    for n in ${WAN_NODE_IDS:-$(seq 1 "$n_nodes")}; do
        for o in $(seq 1 "$ops_per_node"); do
            cat >> "$outfile" <<EOF
  update route 1.${n}${o}.1 7.255.7 linkset as-n${n}op${o}
  update route 1.${n}${o}.3 7.255.7 linkset as-n${n}op${o}
EOF
        done
    done
else
# [2026-08-31] DE 1 AU RANG LE PLUS HAUT, pas "1 a n_operators".
# n_operators compte les CONTENEURS ; il ne dit rien du natif, qui est un
# operateur du banc au meme titre. Avec OP_ID_BASE=2 - conteneurs en 2 et 3,
# natif en 1 - la boucle produisait as-op1 et as-op2 pour DEUX conteneurs
# qui sont en realite op2 et op3 : le hub declarait donc un as-op2 en 1.2.2
# que personne ne venait prendre, et AUCUN as-op3. D ou, cote natif,
#     1.2.2/14 INACC ... PROHIB   1.3.2/14 INACC ... PROHIB
# et un banc ou les operateurs sont attaches mais ne se routent pas.
for i in $(seq 1 "$(( ${OP_ID_BASE:-1} + n_operators - 1 ))"); do
    cat >> "$outfile" <<EOF
  update route 1.${i}.1 7.255.7 linkset as-op${i}
  update route 1.${i}.3 7.255.7 linkset as-op${i}
EOF
done
fi

cat >> "$outfile" <<'EOF'
```

```
!
EOF

if [ "$WAN_MODE" = "1" ]; then
    echo "✓ Config inter-STP WAN generee : $outfile" >&2
    echo "  PC hub  : 0.0.0   Noeuds : $n_nodes × $ops_per_node operateur(s)" >&2
    echo "  PC noeuds : 1.<noeud><op>.<role>   RCTX : noeud*1000 + op*100 + 50" >&2
    echo "  Routes  : $(( n_nodes * ops_per_node * 2 )) (masque 7.255.7)" >&2
else
    echo "✓ Config inter-STP generee : $outfile" >&2
    echo "  PC hub     : 0.0.0   Operateurs : $n_operators" >&2
    echo "  Routes     : $((n_operators * 2)) (masque 7.255.7)" >&2
fi
```

3.150 /opt/GSM/osmo-operator/helpers/prepare_host.sh

4614 octets, 122 lignes → 122 lignes

```
#!/usr/bin/env bash
# force_stack.sh
# A lancer en root (sudo).
# But :
#  - FORCER (kill + relance) Linphone en user
#  - FORCER (kill + relance) Wireshark en root sur UDP/4729 (GSMTAP)
#  - Restart du service docker
# Affichage:
#   [....] action en cours  -> se transforme en -> [ OK ] action finie (meme ligne)

set -euo pipefail

GSMTAP_PORT="${GSMTAP_PORT:-4729}"

# ---------- helpers ----------
step()    { printf "[....] %s" "$1"; }
step_ok() { printf "\r[ OK ] %s\n" "$1"; }
fail()    { printf "\r[FAIL] %s\n" "$1" >&2; exit 1; }

need() { command -v "$1" >/dev/null 2>&1 || fail "Commande manquante: $1"; }

kill_name_as_user() {
  local user="$1" name="$2"
  pkill -u "$user" -x "$name" 2>/dev/null || true
}

kill_name_root() {
  local name="$1"
  pkill -x "$name" 2>/dev/null || true
}

# ---------- preflight ----------
[[ "${EUID}" -eq 0 ]] || fail "Lance en root: sudo $0"

need sudo
need docker
need ss
need systemctl

# ---------- QUI EST L UTILISATEUR DE LA SESSION ----------
# Ce bloc exigeait SUDO_USER non vide ET different de root, et sortait en [FAIL]
# sinon. Deux situations parfaitement legitimes tombaient dedans :
#
#   1. `sudo -i` puis ./launch/start-docker.sh - le shell est deja root, sudo n a plus rien a
#      transmettre et SUDO_USER est VIDE. C est le mode de travail courant sur
```

```
#      ce banc.
#   2. L ISO --desktop : la session graphique EST root. build-iso.sh ecrit
#      AutomaticLogin=root dans /etc/gdm3/custom.conf et decommente le
#      pam_succeed_if qui interdit la connexion de root. "root" est donc ICI la
#      BONNE reponse - la rejeter revenait a refuser la configuration nominale
#      de l image qu on livre.
#
# Le bon critere n est pas le NOM du compte mais la PRESENCE de sa session,
# c est-a-dire de son socket pulse. Meme logique que session_pulse_user() dans
# start-docker.sh, a laquelle ce fichier ne faisait qu ajouter une divergence.
resolve_session_user() {
  local u uid sock
  for u in "${HOST_PULSE_USER:-}" "${SUDO_USER:-}" "$(logname 2>/dev/null || true)"; do
    [[ -n "$u" ]] || continue
    uid="$(id -u "$u" 2>/dev/null)" || continue
    [[ -S "/run/user/${uid}/pulse/native" ]] && { echo "$u"; return 0; }
  done
  # Proprietaire du premier socket pulse trouve : couvre le `sudo -i` et les
  # sessions sans logname (cron, service, terminal detache).
  for sock in /run/user/*/pulse/native; do
    [[ -S "$sock" ]] || continue
    uid="${sock#/run/user/}"; uid="${uid%%/*}"
    u="$(getent passwd "$uid" 2>/dev/null | cut -d: -f1)"
    [[ -n "$u" ]] && { echo "$u"; return 0; }
  done
  # Aucune session pulse vivante : on retombe sur l invocateur, root compris.
  echo "${SUDO_USER:-$(id -un)}"
}
TARGET_USER="$(resolve_session_user)"

id -u "${TARGET_USER}" >/dev/null 2>&1 || fail "User invalide: ${TARGET_USER}"

# ---------- 1) Linphone : kill + relance en user ----------
step "Linphone: kill (user=${TARGET_USER})"
kill_name_as_user "${TARGET_USER}" "linphone"
sleep 0.2
step_ok "Linphone: kill (user=${TARGET_USER})"

step "Linphone: relance (user=${TARGET_USER})"
    # Ne PAS re-resoudre TARGET_USER ici : la ligne qui s y trouvait
    # (${SUDO_USER:-$(logname ...)}) ecrasait le resultat de
    # resolve_session_user() par une valeur moins fiable, juste avant de s en
    # servir. Une seule resolution, en tete de script.
    TARGET_UID=$(id -u "$TARGET_USER")
    # HOME par getent, pas /home/$TARGET_USER en dur : root habite /root. Avec
    # le chemin fige, une session sous root pointait XAUTHORITY sur
    # /home/root/.Xauthority - inexistant - et linphone ne s ouvrait pas.
    TARGET_HOME="$(getent passwd "$TARGET_USER" 2>/dev/null | cut -d: -f6)"
    DISPLAY="${DISPLAY:-:0}"
    XAUTHORITY="${XAUTHORITY:-${TARGET_HOME:-/home/$TARGET_USER}/.Xauthority}"

    setsid sudo -u "$TARGET_USER" \
        env DISPLAY="$DISPLAY" XAUTHORITY="$XAUTHORITY" \
            DBUS_SESSION_BUS_ADDRESS="unix:path=/run/user/${TARGET_UID}/bus" \
        linphone </dev/null >/dev/null 2>&1 &

step_ok "Linphone: relance (user=${TARGET_USER})"

# ---------- 3) Xterm : kill----------
step "Xterm kill"
kill_name_root "xterm"
sleep 0.2
step_ok "Xterm: kill (root)"

# ---------- 2) Wireshark : kill + relance en root sur GSMTAP/4729 ----------
step "Wireshark: kill (root)"
```

```
kill_name_root "wireshark"
sleep 0.2
step_ok "Wireshark: kill (root)"

# ---------- 5) Mini-check (optionnel) ----------
step "Check: listeners UDP/${GSMTAP_PORT}"
ss -lunp | awk -v p=":${GSMTAP_PORT}" '$0 ~ p {print}' >/dev/null 2>&1 || true
step_ok "Check: listeners UDP/${GSMTAP_PORT}"

step "Stack forcee"
step_ok "Stack forcee"
```

3.151 /opt/GSM/osmo-operator/install_modules/00-prereqs.sh

852 octets, 17 lignes → 17 lignes

```
# 00-prereqs - la machine peut-elle accueillir cette installation ?
INST_REGISTER prereqs "Prerequis du systeme"
INST_ROOT[prereqs]=0

inst_prereqs_run() {
    have_cmd apt-get || { inst_hint "cette installation vise Debian/Ubuntu ; sur une autre distribution, utilisez
     ↪  Docker : ./tools/make-docker-image.sh"
                          inst_fail "apt-get introuvable"; return $INST_RC_FAIL; }
    have_cmd git || { inst_fail "git absent"; return $INST_RC_FAIL; }
    local free; free=$(df -Pm "${GSM_ROOT%/*}" 2>/dev/null | awk 'NR==2{print $4}')
    if [ -n "$free" ] && [ "$free" -lt 4096 ]; then
        inst_hint "les sources et les objets de compilation occupent ~4 Go"
        inst_fail "espace disque insuffisant : ${free} Mo libres sur $(dirname "$GSM_ROOT")"
        return $INST_RC_FAIL
    fi
    inst_ok
}
inst_prereqs_verify() { inst_ok; }
```

3.152 /opt/GSM/osmo-operator/install_modules/10-deps.sh

4901 octets, 97 lignes → 97 lignes

```
# 10-deps - paquets systeme.
# La liste fait autorite dans le Dockerfile : on l'en EXTRAIT au lieu de la
# recopier, pour que les deux ne divergent jamais.
INST_REGISTER deps "Dependances systeme (apt)"
INST_DEPS[deps]="prereqs"

# [2026-08-25] L'extraction reconnaissait "RUN apt-get update && apt-get
# install" EN DUR. Le Dockerfile installe desormais via apt-fast : le motif ne
# capturait plus que le bloc d'amorcage (aria2, curl, et les bouts de la
# commande qui installe apt-fast lui-meme), et l'installation native repartait
# avec 29 jetons de charabia au lieu des paquets. Elle echouait sur des noms
# comme "&&" ou une URL, sans que rien ne designe le Dockerfile comme cause.
#
# Deux durcissements plutot qu'un simple elargissement du motif :
#   1. apt-get ET apt-fast sont acceptes, dans n'importe quel ordre update/install ;
#   2. on ne garde que ce qui RESSEMBLE a un nom de paquet. Un bloc apt peut
#      contenir des `&&`, des URL, des accolades - les laisser passer, c'est
#      transformer une erreur de format en erreur d'apt, illisible.
# Le filtre est la vraie protection : il survit au prochain changement de forme.
_deps_list() {
    # Un paquet, c'est ce qui suit "install -y --no-install-recommends" et
    # PRECEDE le premier operateur shell. Filtrer sur la forme des mots ne
    # suffit pas : le bloc d'amorcage contient "chmod", "printf", "rm",
```

```
    # qui ressemblent tous a des noms de paquets. On coupe donc a `&&`, `||`,
    # `{` ou une URL, et on ne garde que ce qu'il y a avant.
    awk '
        /^RUN( --mount=[^ ]+)? *\\?$/ { pending = 1; next }
        /^RUN( --mount=[^ ]+)? +apt-(get|fast) update && apt-(get|fast) install/ || (pending && /^ *apt-(get|fast)
↪   update && apt-(get|fast) install/) {
            pending = 0
            inblk = 1
            line = $0
            # Le RUN peut porter un --mount=... (cache apt BuildKit) et se
            # continuer a la ligne suivante : on coupe tout ce qui precede la
            # fin de la commande apt, que la ligne commence par RUN ou non.
            sub(/^.*--no-install-recommends/, "", line)
            emit(line)
            next
        }
        pending && !/^ *apt-(get|fast) update/ { pending = 0 }
        inblk { emit($0) }

        function emit(l,   n, a, i, cont) {
            # Le commentaire en PREMIER. Les lignes "# ..." au milieu de la
            # liste nont pas de barre de continuation : Docker les retire avant
            # le shell. Les traiter comme une fin de bloc coupait la liste au
            # premier commentaire - soit trois paquets sur soixante-dix.
            # (Pas dapostrophe dans ce bloc : il vit entre quotes simples.)
            sub(/#.*/, "", l)
            if (l ~ /^[ \t]*$/) return          # commentaire pur : transparent

            cont = (l ~ /\\[ \t]*$/)
            if (l ~ /&&|\|\||[{}]|https?:/) {   # on sort de la liste de paquets
                sub(/(&&|\|\||\{).*/, "", l)
                inblk = 0
            } else if (!cont) {
                inblk = 0                        # derniere ligne du bloc
            }
            sub(/\\[ \t]*$/, "", l)
            n = split(l, a, /[ \t]+/)
            for (i = 1; i <= n; i++)
                if (a[i] ~ /^[a-z0-9][a-z0-9.+*-]*$/) print a[i]
        }
    ' "$INST_TREE/Dockerfile" 2>/dev/null | sort -u | tr '\n' ' '
}

inst_deps_check() {
    have_file "$INST_TREE/Dockerfile" || { inst_fail "Dockerfile introuvable - c'est lui qui porte la liste des
↪   paquets"; return $INST_RC_FAIL; }
    inst_ok
}
# [2026-10-18] « Deja installe » se jugeait sur trois paquets temoins : un
# systeme partiellement equipe (les binaires sans les -dev) passait pour complet,
# l etape etait sautee, et osmo-hlr echouait trois etapes plus loin sur
# « Package requirements (sqlite3) were not met ». On verifie donc TOUTE la
# liste. Un motif (asterisk-moh*) compte comme present si l un des paquets qu il
# designe est installe.
_deps_missing() {
    local p
    for p in $(_deps_list); do
        dpkg-query -W -f='${db:Status-Status}\n' "$p" 2>/dev/null | grep -qx installed || printf '%s\n' "$p"
    done
}
inst_deps_done() {
    [ -n "$(_deps_list)" ] && [ -z "$(_deps_missing)" ]
}
inst_deps_run() {
    local pkgs; pkgs="$(_deps_list)"
```

```
    [ -z "${pkgs// }" ] && { inst_fail "liste de paquets vide - le format du Dockerfile a change"; return
    ↪  $INST_RC_FAIL; }
    inst_say "paquets : $(echo $pkgs | wc -w), dont manquants : $(_deps_missing | wc -l)"
    # La liste COMPLETE, pas seulement les manquants : apt-get install est
    # idempotent, et un motif comme asterisk-moh* ne se resout bien que par apt.
    apt-get update -qq && apt-get install -y --no-install-recommends $pkgs
}
inst_deps_verify() {
    local missing; missing="$(_deps_missing | tr '\n' ' ')"
    [ -n "${missing// }" ] && { inst_fail "paquets absents apres installation : $missing"; return $INST_RC_FAIL; }
    inst_ok
}
```

3.153 /opt/GSM/osmo-operator/install_modules/20-user.sh

597 octets, 11 lignes → 11 lignes

```
# 20-user - compte de service, tel que le Dockerfile le cree.
INST_REGISTER user "Utilisateur et groupe osmocom"
INST_DEPS[user]="prereqs"

inst_user_done() { getent passwd osmocom >/dev/null && getent group osmocom >/dev/null; }
inst_user_run() {
    getent group  osmocom >/dev/null || groupadd osmocom || return $INST_RC_FAIL
    getent passwd osmocom >/dev/null || useradd -r -g osmocom -s /sbin/nologin -d /var/lib/osmocom osmocom ||
    ↪  return $INST_RC_FAIL
    inst_ok
}
inst_user_verify() { inst_user_done && inst_ok || { inst_fail "le compte osmocom n'existe pas"; return
↪  $INST_RC_FAIL; }; }
```

3.154 /opt/GSM/osmo-operator/install_modules/30-sources.sh

2026 octets, 49 lignes → 49 lignes

```
# 30-sources - recuperation des depots. Idempotent : un depot deja clone est laisse tel quel
# (on ne tire pas : ce serait modifier le travail en cours de l'utilisateur sans le lui demander).
INST_REGISTER sources "Recuperation des sources Osmocom"
INST_DEPS[sources]="prereqs"

_SRC_REPOS="osmocom-bb|https://gitea.osmocom.org/phone-side/osmocom-bb|
osmo-gapk|https://gitea.osmocom.org/osmocom/gapk|
gsup-smsc-proto|https://gitea.osmocom.org/themwi/gsup-smsc-proto|
libosmo-dsp|https://gitea.osmocom.org/sdr/libosmo-dsp.git|"

inst_sources_done() {
    local l; while IFS='|' read -r dir url br; do
        [ -z "$dir" ] && continue
        have_repo "$GSM_ROOT/$dir" || return 1
    done <<< "$_SRC_REPOS"
    return 0
}
inst_sources_run() {
    mkdir -p "$GSM_ROOT" || { inst_fail "impossible de creer $GSM_ROOT"; return $INST_RC_FAIL; }
    local dir url br n=0
    while IFS='|' read -r dir url br; do
        [ -z "$dir" ] && continue
        if have_repo "$GSM_ROOT/$dir"; then
            if [ "${REINSTALL:-0}" = 1 ]; then
                inst_say "$dir : deja present, mise a jour (git pull --ff-only)"
                git -C "$GSM_ROOT/$dir" pull --ff-only \
                    || inst_say "$dir : mise a jour impossible (modifications locales ?), laisse tel quel"
```

```
                n=$((n+1))
            else
                inst_say "$dir : deja present, laisse tel quel"
            fi
            continue
        fi
        inst_say "clonage $dir"
        git clone ${br:+--branch "$br"} "$url" "$GSM_ROOT/$dir" || { inst_fail "echec du clonage de $dir ($url)";
        ↪  return $INST_RC_FAIL; }
        n=$((n+1))
    done <<< "$_SRC_REPOS"
    [ $n -eq 0 ] && return $INST_RC_DONE
    inst_ok
}
inst_sources_verify() {
    local missing="" dir url br
    while IFS='|' read -r dir url br; do
        [ -z "$dir" ] && continue
        have_repo "$GSM_ROOT/$dir" || missing="$missing $dir"
    done <<< "$_SRC_REPOS"
    [ -n "$missing" ] && { inst_fail "depots absents :$missing"; return $INST_RC_FAIL; }
    inst_ok
}
```

3.155 /opt/GSM/osmo-operator/install_modules/40-patches.sh

1478 octets, 35 lignes → 35 lignes

```
# 40-patches - correctifs locaux. `git apply --check` d'abord : un correctif deja
# applique ne doit pas faire echouer l'installation.
INST_REGISTER patches "Application des correctifs"
INST_DEPS[patches]="sources"
INST_REQUIRED[patches]=0

_patch_target() {
    case "$(basename "$1")" in
        osmo-bts-*)   printf '%s\n' "$GSM_ROOT/osmo-bts" ;;
        osmo-msc-*)   printf '%s\n' "$GSM_ROOT/osmo-msc" ;;
        osmo-hlr-*)   printf '%s\n' "$GSM_ROOT/osmo-hlr" ;;
        osmo-trx-*)   printf '%s\n' "$GSM_ROOT/osmo-trx" ;;
        # [2026-10-19] avant osmocom-bb-* : l arbre jolly/testing (appli transceiver)
        osmocom-bb-transceiver-*) printf '%s\n' "$GSM_ROOT/osmocom-bb-transceiver" ;;
        osmocom-bb-*) printf '%s\n' "$GSM_ROOT/osmocom-bb" ;;
        grgsm-*)      printf '%s\n' "$GSM_ROOT/gr-gsm" ;;
        *)            printf '' ;;
    esac
}
inst_patches_run() {
    local p t n=0 skipped=0
    for p in "$INST_TREE"/patches/*.patch; do
        [ -f "$p" ] || continue
        t="$(_patch_target "$p")"
        [ -z "$t" ] || [ ! -d "$t" ] && { skipped=$((skipped+1)); continue; }
        if git -C "$t" apply --check "$p" 2>/dev/null; then
            git -C "$t" apply "$p" && { inst_say "applique : $(basename "$p")"; n=$((n+1)); }
        else
            inst_say "deja applique ou sans objet : $(basename "$p")"; skipped=$((skipped+1))
        fi
    done
    [ $n -eq 0 ] && return $INST_RC_DONE
    inst_ok
}
inst_patches_verify() { inst_ok; }
```

3.156 /opt/GSM/osmo-operator/install_modules/50-build.sh

2973 octets, 62 lignes → 62 lignes

```
# 50-build - compilation. L'etape longue : d'ou le timeout large et le journal detaille.
INST_REGISTER build "Compilation des sources"
INST_DEPS[build]="deps sources"
INST_TIMEOUT[build]=5400

_BUILD_ORDER="libosmo-dsp osmocom-bb osmo-gapk"

inst_build_done() { have_file "$GSM_ROOT/osmocom-bb/src/host/layer23/src/mobile/mobile"; }
inst_build_run() {
    local d
    for d in $_BUILD_ORDER; do
        [ -d "$GSM_ROOT/$d" ] || { inst_say "$d absent, ignore"; continue; }
        inst_say "=== $d ==="
        ( cd "$GSM_ROOT/$d" || exit 1
          [ -x ./configure ] || [ -f ./configure.ac ] && { autoreconf -fi || true; }
          [ -x ./configure ] && ./configure
          make -j"$(nproc)" ) || { inst_hint "detail dans le journal de cette etape"
                                   inst_fail "echec de compilation : $d"; return $INST_RC_FAIL; }
    done

    # toast : codec GSM 06.10 (quut.com). libgsm1 fournit la lib, pas le binaire.
    # Sources sous $GSM_ROOT, binaire dans /usr/local/bin. Optionnel : un echec
    # (reseau, quut.com injoignable) n arrete pas l installation.
    inst_say "=== toast (codec GSM 06.10) ==="
    if command -v toast >/dev/null 2>&1; then
        inst_say "toast deja present ($(command -v toast))"
    else
        local gver=gsm-1.0.24
        local gdir=gsm-1.0-pl24   # le tarball se decompresse SOUS ce nom
        local gurl="https://www.quut.com/gsm/${gver}.tar.gz"
        if ( cd "$GSM_ROOT" \
             && wget -qO "${gver}.tar.gz" "$gurl" \
             && tar xzf "${gver}.tar.gz" \
             && cd "$gdir" \
             && { make >/dev/null 2>&1 || true; } \
             && { [ -x bin/toast ] || make toast >/dev/null 2>&1 || true; } \
             && [ -x bin/toast ] ); then
            for b in toast untoast tcat; do
                [ -e "$GSM_ROOT/$gdir/bin/$b" ] \
                    && install -m755 "$GSM_ROOT/$gdir/bin/$b" "/usr/local/bin/$b"
            done
            inst_say "toast installe dans /usr/local/bin (sources : $GSM_ROOT/$gdir)"
        else
            inst_hint "toast non compile (reseau ou quut.com injoignable) - suite sans lui"
        fi
    fi
    # ── La 4G : srsRAN (ZeroMQ) et Open5GS dans /opt/LTE ──────────────────
    # [2026-09-08] Le meme script que le Dockerfile et addition.sh
    # (tools/osmo-lte-install.sh) : .deb du cache s ils sont la, compilation
    # sinon. Non fatal : la 2G n en depend pas.
    inst_say "=== la 4G (srsRAN ZeroMQ + Open5GS) ==="
    if [ -f "$INST_TREE/tools/osmo-lte-install.sh" ]; then
        OSMO_REPO="$INST_TREE" bash "$INST_TREE/tools/osmo-lte-install.sh" --all \
            || inst_hint "4G non installee (voir ci-dessus) - « osmo-lte-install --build » plus tard"
    fi
    inst_ok
}
inst_build_verify() {
    have_file "$GSM_ROOT/osmocom-bb/src/host/layer23/src/mobile/mobile" \
        || { inst_fail "le binaire mobile n'a pas ete produit"; return $INST_RC_FAIL; }
    inst_ok
}
```

3.157 /opt/GSM/osmo-operator/install_modules/60-binaires.sh

1174 octets, 30 lignes → 30 lignes

```
# 60-binaires - depot des executables dans /usr/local/bin, comme le fait le Dockerfile.
INST_REGISTER binaires "Installation des binaires"
INST_DEPS[binaires]="build"

_BINS="trxcon:src/host/trxcon/src/trxcon
mobile:src/host/layer23/src/mobile/mobile
virtphy:src/host/virt_phy/src/virtphy
ccch_scan:src/host/layer23/src/misc/ccch_scan"

inst_binaires_done() { have_cmd mobile && have_cmd trxcon; }
inst_binaires_run() {
    local line name path n=0
    while IFS=: read -r name path; do
        [ -z "$name" ] && continue
        if [ -x "$GSM_ROOT/osmocom-bb/$path" ]; then
            install -m755 "$GSM_ROOT/osmocom-bb/$path" "/usr/local/bin/$name" && n=$((n+1))
        else
            inst_say "absent, ignore : $path"
        fi
    done <<< "$_BINS"
    [ $n -eq 0 ] && { inst_fail "aucun binaire trouve a installer - la compilation a-t-elle abouti ?"; return
     ↪  $INST_RC_FAIL; }
    inst_say "$n binaires installes"
    inst_ok
}
inst_binaires_verify() {
    local missing="" b
    for b in mobile trxcon; do have_cmd "$b" || missing="$missing $b"; done
    [ -n "$missing" ] && { inst_fail "binaires absents du PATH :$missing"; return $INST_RC_FAIL; }
    inst_ok
}
```

3.158 /opt/GSM/osmo-operator/install_modules/70-configs.sh

1520 octets, 28 lignes → 28 lignes

```
# 70-configs - configuration Osmocom. `cp -n` : on n'ecrase JAMAIS une configuration
# existante, elle a pu etre adaptee a la machine.
INST_REGISTER configs "Configuration (/etc/osmocom)"
INST_DEPS[configs]="prereqs"

inst_configs_done() { have_file /etc/osmocom/osmo-bsc.cfg; }
inst_configs_run() {
    mkdir -p /etc/osmocom /var/log/osmocom /var/run/smsc "${OSMOCOM_HOME:-$HOME/.osmocom}/bb" \
        || { inst_fail "impossible de creer les repertoires de configuration"; return $INST_RC_FAIL; }
    if [ "${REINSTALL:-0}" = 1 ]; then
        # On ecrase, mais jamais sans copie : /etc/osmocom.<date> garde l etat
        # d avant, comme le fait --regen dans start-native.sh.
        local sav="/etc/osmocom.$(date +%Y%m%d-%H%M%S)"
        cp -a /etc/osmocom "$sav" 2>/dev/null && inst_say "sauvegarde : $sav"
        cp -f "$INST_TREE"/configs/*.cfg /etc/osmocom/ 2>/dev/null
        [ -f "$INST_TREE/configs/mobile.cfg" ] && cp -f "$INST_TREE/configs/mobile.cfg"
         ↪  "${OSMOCOM_HOME:-$HOME/.osmocom}/bb/mobile.cfg"
        inst_say "configurations reecrites (--reinstall)"
    else
        cp -n "$INST_TREE"/configs/*.cfg /etc/osmocom/ 2>/dev/null
        [ -f "$INST_TREE/configs/mobile.cfg" ] && cp -n "$INST_TREE/configs/mobile.cfg"
         ↪  "${OSMOCOM_HOME:-$HOME/.osmocom}/bb/mobile.cfg"
        inst_say "configurations deposees (les fichiers existants ont ete conserves)"
```

```
    fi
    inst_ok
}
inst_configs_verify() {
    have_dir /etc/osmocom || { inst_fail "/etc/osmocom absent"; return $INST_RC_FAIL; }
    inst_ok
}
```

3.159 /opt/GSM/osmo-operator/install_modules/80-bureau.sh

10780 octets, 188 lignes → 188 lignes

```
# ============================================================================
#  80-bureau - icones et raccourcis du bureau (install native)
# ============================================================================
#  Pose EXACTEMENT ce que l ISO pose (build-iso.sh, etape 8d) a partir des
#  memes fichiers du depot :
#     data/desktop/osmo-launch.desktop   "Lancer le banc GSM"  -> launch.sh
#     data/*.svg                         les icones
#
#  osmo-multi (antenne, multi-operator) N EST PLUS POSEE ICI : son lanceur
#  start-multi.sh suppose le supplement SS7 (docker + image + topologie), qui
#  n existe qu apres addition.sh. C est addition.sh qui pose l antenne, a la fin
#  d une install SS7 reussie - pas avant, ou elle ne ferait rien au clic.
#  et le lien /usr/local/bin/osmo-start-direct -> start-native.sh.
#
#  Icon= est reecrit en CHEMIN ABSOLU (/usr/share/osmo-operator/icons/*.svg) :
#  un nom de theme passe par icon-theme.cache, qui ne connait pas une icone
#  arrivee apres lui, et GNOME affiche alors une page blanche sans rien dire
#  (voir update.sh, osmo_reposer_icones). Le cache est quand meme reconstruit
#  pour le menu des applications.
#
#  Les raccourcis sont copies sur le bureau de root et de l utilisateur qui a
#  lance sudo (SUDO_USER), en Bureau/ et Desktop/. L attribut metadata::trusted
#  ne peut pas etre pose ici (il vit dans les metadonnees gvfs de la session) :
#  update.sh le pose au login, et un double-clic + "Autoriser" suffit sinon.
#
#  [2026-10-12] LE FOND D ECRAN AUSSI. L ISO pose configs/gsm-lab-fond.jpg
#  comme fond GNOME par defaut (iso_modules/80-chroot.sh, override de schema) ;
#  l install native ne le faisait pas, et le bureau restait sur le numbat
#  d Ubuntu. On pose le meme fichier (/usr/share/backgrounds/gsm-lab-fond.jpg)
#  et le meme override (99-osmo-fond.gschema.override) : un DEFAUT de schema,
#  pas un reglage d utilisateur - qui a choisi un autre fond le garde.
#
#  Ni osmo-tutorial, ni osmo-addition, ni osmo-update : leurs lanceurs sont
#  des enrobages ecrits par build-iso.sh pour l image (terminal + pkexec) et
#  dependent du bureau GNOME de l ISO. Sur une machine deja installee, launch.sh
#  et start-multi.sh se suffisent.
# ----------------------------------------------------------------------------
INST_REGISTER bureau "Bureau : icones et raccourcis"
INST_DEPS[bureau]="configs"
INST_REQUIRED[bureau]=0

_BUREAU_ICONS=/usr/share/osmo-operator/icons
_BUREAU_ENTRIES="osmo-launch osmo-dino"
# [2026-10-12] Le fond au repos est la ville SANS antenne (gsm-lab-ville.jpg) ;
# gsm-lab-antenne.jpg (la meme ville, le pylone dresse) est posee en fondu par
# tools/osmo-fond.py quand une 2G/4G tourne. Les deux sont copiees ; le
# paysage low-poly (gsm-lab-fond.jpg) aussi, sans etre le defaut.
_BUREAU_FOND=/usr/share/backgrounds/gsm-lab-ville.jpg
_BUREAU_FONDS="gsm-lab-antenne.jpg gsm-lab-fond.jpg"
_BUREAU_SCHEMA=/usr/share/glib-2.0/schemas/99-osmo-fond.gschema.override

# Le fond par defaut : meme fichier et memes trois sections que l ISO
```

```
# (org.gnome.desktop.background, :ubuntu, :GNOME-Greeter - une session
# ubuntu:GNOME lit les secondes de preference). Sans glib-compile-schemas
# (pas de GNOME ici), on pose le fichier et on s arrete la.
_bureau_fond() {
    local src="$INST_TREE/configs/gsm-lab-ville.jpg" sec f
    for f in $_BUREAU_FONDS; do
        [ -f "$INST_TREE/configs/$f" ] && install -Dm644 "$INST_TREE/configs/$f" "/usr/share/backgrounds/$f"
    done
    ln -sf "$INST_TREE/tools/osmo-fond.py" /usr/local/bin/osmo-fond
    [ -f "$src" ] || { inst_hint "configs/gsm-lab-ville.jpg absent : fond GNOME inchange"; return 0; }
    install -Dm644 "$src" "$_BUREAU_FOND" || { inst_hint "fond non pose : $_BUREAU_FOND"; return 0; }
    [ -d /usr/share/glib-2.0/schemas ] || { inst_say "fond : $_BUREAU_FOND (pas de schemas GNOME ici)"; return 0; }
    : > "$_BUREAU_SCHEMA"
    for sec in "org.gnome.desktop.background" "org.gnome.desktop.background:ubuntu"
     ↪  "org.gnome.desktop.background:GNOME-Greeter"; do
        printf "\n[%s]\npicture-uri='file://%s'\npicture-uri-dark='file://%s'\npicture-options='zoom'\nprimary-
         ↪  color='#0d1b2a'\n"
         ↪  \
            "$sec" "$_BUREAU_FOND" "$_BUREAU_FOND" >> "$_BUREAU_SCHEMA"
    done
    # [2026-10-24] MODE CLAIR PAR DEFAUT, comme l ISO (iso_modules/80-chroot.sh) :
    # Yaru-blue, color-scheme default. 99-osmo-live (pose par l ISO sur une
    # machine installee depuis elle) disait prefer-dark : il est remis au clair.
    for sec in "org.gnome.desktop.interface" "org.gnome.desktop.interface:ubuntu"
     ↪  "org.gnome.desktop.interface:GNOME-Greeter"; do
        printf "\n[%s]\ncolor-scheme='default'\ngtk-theme='Yaru-blue'\nicon-theme='Yaru-blue'\ncursor-
         ↪  theme='Yaru'\n"
         ↪  \
            "$sec" >> "$_BUREAU_SCHEMA"
    done
    [ -f /usr/share/glib-2.0/schemas/99-osmo-live.gschema.override ] \
        && sed -i -e "s/^color-scheme='prefer-dark'/color-scheme='default'/" -e
         ↪  "s/^gtk-theme='Yaru-blue-dark'/gtk-theme='Yaru-blue'/" \
            /usr/share/glib-2.0/schemas/99-osmo-live.gschema.override
    if command -v glib-compile-schemas >/dev/null 2>&1; then
        glib-compile-schemas /usr/share/glib-2.0/schemas 2>/dev/null \
            && inst_say "fond GNOME par defaut : $_BUREAU_FOND" \
            || inst_hint "glib-compile-schemas a echoue : fond pose, defaut non compile"
    else
        inst_say "fond : $_BUREAU_FOND (glib-compile-schemas absent, defaut non compile)"
    fi
}

_bureau_homes() {
    local h
    printf '%s\n' /root
    if [ -n "${SUDO_USER:-}" ] && [ "$SUDO_USER" != root ]; then
        h="$(getent passwd "$SUDO_USER" | cut -d: -f6)"
        [ -n "$h" ] && [ -d "$h" ] && printf '%s\n' "$h"
    fi
}

inst_bureau_check() {
    [ -d "$INST_TREE/data/desktop" ] || { inst_fail "data/desktop/ absent du depot"; return $INST_RC_FAIL; }
    [ -x "$INST_TREE/launch.sh" ] || inst_hint "launch.sh n est pas executable : chmod +x $INST_TREE/launch.sh"
    inst_ok
}

inst_bureau_done() {
    local e
    for e in $_BUREAU_ENTRIES; do
        have_file "/usr/share/applications/$e.desktop" || return 1
        have_file "$_BUREAU_ICONS/$e.svg" || return 1
    done
    [ -L /usr/local/bin/osmo-start-direct ] || return 1
    [ -L /usr/local/bin/c123 ] || return 1
```

```
    [ -L /usr/local/bin/banc-rssi ] || return 1
    return 0
}

inst_bureau_run() {
    local e h d f
    install -d "$_BUREAU_ICONS" /usr/share/icons/hicolor/scalable/apps /usr/share/applications \
        || { inst_fail "impossible de creer les repertoires du bureau"; return $INST_RC_FAIL; }
    for e in $_BUREAU_ENTRIES; do
        [ -f "$INST_TREE/data/$e.svg" ] && {
            install -m644 "$INST_TREE/data/$e.svg" "$_BUREAU_ICONS/$e.svg"
            install -m644 "$INST_TREE/data/$e.svg" "/usr/share/icons/hicolor/scalable/apps/$e.svg"
        }
        install -m644 "$INST_TREE/data/desktop/$e.desktop" "/usr/share/applications/$e.desktop" \
            || { inst_fail "copie impossible : $e.desktop"; return $INST_RC_FAIL; }
        # Exec= et Icon= suivent l arbre INSTALLE, pas /opt/GSM en dur : ce
        # depot peut vivre ailleurs (GSM_ROOT).
        sed -i -e "s|^Exec=/opt/GSM/osmo-operator/|Exec=$INST_TREE/|" \
               -e "s|^Icon=.*|Icon=$_BUREAU_ICONS/$e.svg|" "/usr/share/applications/$e.desktop"
        inst_say "menu : /usr/share/applications/$e.desktop"
    done
    # osmo-start-native : le nom du lanceur depuis le 2026-10-18 ; osmo-start-direct
    # garde l ancien nom pour les habitudes et les scripts deja ecrits.
    ln -sf "$INST_TREE/start-native.sh" /usr/local/bin/osmo-start-native
    ln -sf "$INST_TREE/start-native.sh" /usr/local/bin/osmo-start-direct
    # [2026-10-24] c123 : la fenetre du C123 emule (tools/osmo-c123-gtk.py) dans le
    # PATH, pas seulement en alias de shell de connexion. Elle tourne sous le
    # python3 SYSTEME : PyGObject + GTK3 + cairo (Gdk.cairo_set_source_pixbuf
    # veut python3-gi-cairo) viennent d apt, pas de pip.
    ln -sf "$INST_TREE/tools/osmo-c123-gtk.py" /usr/local/bin/c123
    ln -sf "$INST_TREE/tools/osmo-c123-gtk.py" /usr/local/bin/osmo-c123
    # [2026-10-24] banc-rssi, banc-trx, c123-1... : profile.d ET vraies commandes (tools/osmo-banc-cmd.sh)
    install -m644 "$INST_TREE/scripts/osmo-banc-aliases.sh" /etc/profile.d/98-osmo-banc-aliases.sh
    OSMO_BANC_TREE="$INST_TREE" bash "$INST_TREE/tools/osmo-banc-cmd.sh" --install /usr/local/bin >/dev/null
    install -m644 "$INST_TREE/scripts/osmo-gnuarm-profile.sh" /etc/profile.d/97-osmo-gnuarm.sh   # gnuarm dans le
     ↪  PATH, s il est pose
    # [2026-10-24] gcc-9 et gcc-11 a cote de gcc-13, et leurs alternatives, comme l ISO
    # (iso_modules/80-chroot.sh) : jolly/testing et burst_ind d osmocom-bb veulent gcc-9.
    apt-get install -y --no-install-recommends gcc-9 g++-9 gcc-11 g++-11 >/dev/null \
        || inst_hint "gcc-9 / gcc-11 non installes : alternatives limitees aux compilateurs presents"
    bash "$INST_TREE/tools/osmo-gcc-alternatives.sh"
    # [2026-10-24] WireGuard du WAN : « wg 1 » / « wg N ... » (tools/osmo-wg.sh), comme l ISO
    install -m644 "$INST_TREE/scripts/osmo-wg-profile.sh" /etc/profile.d/97-osmo-wg.sh
    ln -sf "$INST_TREE/tools/osmo-wg.sh" /usr/local/bin/osmo-wg
    command -v wg-quick >/dev/null 2>&1 \
        || apt-get install -y --no-install-recommends wireguard-tools >/dev/null \
        || inst_hint "wireguard-tools non installe : « wg N » ne montera pas de tunnel"
    /usr/bin/python3 -c 'import gi, cairo; gi.require_version("Gtk", "3.0"); from gi.repository import Gtk'
     ↪  2>/dev/null \
        || apt-get install -y --no-install-recommends python3-gi python3-gi-cairo python3-cairo \
               gir1.2-gtk-3.0 gir1.2-gdkpixbuf-2.0 >/dev/null \
        || inst_hint "c123 : paquets GTK non installes (python3-gi python3-gi-cairo gir1.2-gtk-3.0)"

    while read -r h; do
        for d in Bureau Desktop; do
            [ -d "$h/$d" ] || continue
            for e in $_BUREAU_ENTRIES; do
                f="$h/$d/$e.desktop"
                cp -f "/usr/share/applications/$e.desktop" "$f" && chmod +x "$f"
                inst_say "bureau : $f"
            done
            [ "$h" != /root ] && chown "$SUDO_USER" "$h/$d"/osmo-*.desktop 2>/dev/null
        done
    done < <(_bureau_homes)
```

```
    command -v gtk-update-icon-cache >/dev/null 2>&1 && gtk-update-icon-cache -f -q /usr/share/icons/hicolor
     ↪  2>/dev/null
    command -v update-desktop-database >/dev/null 2>&1 && update-desktop-database /usr/share/applications
     ↪  2>/dev/null
    _bureau_fond
    inst_ok
}

inst_bureau_verify() {
    local e
    for e in $_BUREAU_ENTRIES; do
        grep -q "^Exec=$INST_TREE/" "/usr/share/applications/$e.desktop" \
            || { inst_fail "$e.desktop ne pointe pas sur $INST_TREE"; return $INST_RC_FAIL; }
    done
    inst_ok
}
```

3.160 /opt/GSM/osmo-operator/install_modules/90-verify.sh

734 octets, 17 lignes → 17 lignes

```
# 90-verify - controle d'ensemble. Distinct des `verify` de chaque etape :
# ici on regarde si la machine est utilisable, pas si une etape a reussi.
INST_REGISTER verify "Verification de l'installation"
INST_DEPS[verify]="binaires configs"
INST_ROOT[verify]=0

inst_verify_run() {
    local missing="" b
    for b in osmo-stp osmo-hlr osmo-msc osmo-mgw osmo-bsc; do have_cmd "$b" || missing="$missing $b"; done
    if [ -n "$missing" ]; then
        inst_hint "les demons Osmocom ne sont pas dans les depots Ubuntu par defaut : ajoutez le depot Osmocom, ou
         ↪  utilisez Docker (./tools/make-docker-image.sh)"
        inst_fail "demons absents :$missing"
        return $INST_RC_FAIL
    fi
    inst_ok
}
inst_verify_verify() { inst_ok; }
```

3.161 /opt/GSM/osmo-operator/install_modules/_lib/inst.sh

2527 octets, 60 lignes → 60 lignes

```
# ============================================================================
#  install_modules/_lib/inst.sh - contrat des modules d'installation
# ============================================================================
#
#  Meme principe que run_modules/_lib/mod.sh, mais pour l'installation. La
#  difference tient a la nature des etapes : on n'y "demarre" rien, on
#  transforme la machine - donc on veut savoir, pour chacune, si elle est DEJA
#  faite (idempotence) et si elle a REUSSI (verification).
#
#  Quatre fonctions par module, `run` seule obligatoire :
#     inst_<p>_check    prerequis, LECTURE SEULE          0 ok · 1 manque · 4 sans objet
#     inst_<p>_done     deja installe ?                   0 oui · 1 non
#     inst_<p>_run      fait le travail                   0 ok · 1 echec · 3 deja fait
#     inst_<p>_verify   controle APRES coup               0 ok · 1 echec
#
#  `verify` est le pendant de la barriere `wait` cote execution : installer
#  n'est pas avoir installe. Un `apt-get` qui rend 0 mais laisse un binaire
#  absent doit etre vu ici, pas trois etapes plus loin.
```

```
# ------------------------------------------------------------------------

readonly INST_RC_OK=0
readonly INST_RC_FAIL=1
readonly INST_RC_DONE=3
readonly INST_RC_NA=4

declare -a INST_ORDER=()
declare -A INST_DESC=()      # description affichee
declare -A INST_DEPS=()      # etapes prerequises
declare -A INST_REQUIRED=()  # 1 = un echec arrete tout
declare -A INST_ROOT=()      # 1 = exige les droits root
declare -A INST_TIMEOUT=()   # secondes, pour les etapes longues

INST_REGISTER() {
    local slug="$1" desc="$2"
    INST_ORDER+=("$slug")
    INST_DESC[$slug]="$desc"
    INST_DEPS[$slug]=""
    INST_REQUIRED[$slug]=1
    INST_ROOT[$slug]=1
    INST_TIMEOUT[$slug]=1800
}

inst_prefix() { printf 'inst_%s' "${1//-/_}"; }

# --- protocole de message (identique a run_modules) ---------------------------
_INST_REASON=""
_INST_HINT=""
inst_ok()   { _INST_REASON=""   ; return $INST_RC_OK; }
inst_fail() { _INST_REASON="$*" ; return $INST_RC_FAIL; }
inst_done() { _INST_REASON="$*" ; return $INST_RC_DONE; }
inst_na()   { _INST_REASON="$*" ; return $INST_RC_NA; }
inst_hint() { _INST_HINT="$*"; }
inst_say()  { printf '%s\n' "$*"; }

# --- sondes -------------------------------------------------------------------
have_cmd()  { command -v "$1" >/dev/null 2>&1; }
have_dir()  { [ -d "$1" ]; }
have_file() { [ -f "$1" ]; }
have_pkg()  { dpkg -s "$1" >/dev/null 2>&1; }
have_repo() { [ -d "$1/.git" ]; }
```

3.162 /opt/GSM/osmo-operator/installer/calamares/settings.conf

5054 octets, 127 lignes → 127 lignes

```
# ============================================================================
#  Calamares - l'installeur de l'ISO osmo-operator
# ============================================================================
#  POURQUOI CALAMARES ET PAS UBIQUITY. Ubiquity est l'installeur d'Ubuntu, il
#  est dans jammy (22.04.15) et il aurait l'air du bon choix. Il ne l'est pas :
#  il lit l'etat du systeme live par CASPER, l'initramfs d'Ubuntu. Cette ISO
#  demarre avec LIVE-BOOT, celui de Debian (build-iso.sh installe live-boot et
#  live-boot-initramfs-tools, pas casper). Ubiquity ne trouverait donc ni le
#  squashfs, ni le point de montage du medium, et echouerait avant la premiere
#  question. Calamares, lui, ne suppose rien : on lui DIT ou est le squashfs
#  (modules/unpackfs.conf). C'est l'installeur des distributions qui ne sont
#  pas Ubuntu tout en etant construites sur Debian - exactement notre cas.
#
#  L'ORDRE DES MODULES N'EST PAS DECORATIF. "exec" s'execute de haut en bas et
#  chaque etape suppose la precedente : partition cree les systemes de fichiers,
#  mount les monte, unpackfs y deverse le squashfs, et TOUT ce qui suit
#  (machineid, fstab, locale, users, grub) ecrit DANS cette cible montee. Placer
```

```
#  users avant unpackfs, c'est ecrire dans un repertoire vide qui sera ensuite
#  recouvert - le compte disparait sans un message.
# ------------------------------------------------------------------------------
---
modules-search: [ local, /usr/lib/x86_64-linux-gnu/calamares/modules ]

# Une INSTANCE nommee du module shellprocess : elle porte nos garanties propres
# (root par defaut, osmocom en second) et tourne apres grub, dans la cible.
instances:
- id:     osmo
  module: shellprocess
  config: shellprocess-osmo.conf
# Le pilote NVIDIA : la page de choix, puis l installation conditionnelle.
- id:     nvidia
  module: packagechooser
  config: packagechooser-nvidia.conf
- id:     nvidia
  module: contextualprocess
  config: contextualprocess-nvidia.conf
# Le miroir Ubuntu du systeme installe : la page de choix (mesuree par
# osmo-install au lancement, le plus rapide en defaut), puis l ecriture.
- id:     mirror
  module: packagechooser
  config: packagechooser-mirror.conf
- id:     mirror
  module: contextualprocess
  config: contextualprocess-mirror.conf
# Installe grub2-common dans la cible AVANT bootloader, sans quoi grub-install
# n'existe pas la-bas et le module sort en « code d'erreur 127 » sur un disque
# deja rempli. Les .deb arrivent par le bind /run/osmo-debs de mount.conf.
- id:     grub
  module: shellprocess
  config: shellprocess-grub.conf
# [2026-10-24] Les supplements d addition.sh : cases a cocher, puis l origine de l image
# operateur ; le job osmosupplements les ecrit dans la cible pour le premier demarrage.
- id:     supplements
  module: packagechooser
  config: packagechooser-supplements.conf
- id:     image
  module: packagechooser
  config: packagechooser-image.conf

sequence:
- show:
  - welcome
  - locale
  - keyboard
  - partition
  - users
  - packagechooser@nvidia
  - packagechooser@mirror
  - packagechooser@supplements
  - packagechooser@image
  - summary
- exec:
  - partition
  - mount
  - unpackfs
  # Le miroir choisi, AVANT tout ce qui pourrait lancer apt dans la cible.
  - contextualprocess@mirror
  - machineid
  - fstab
  - locale
  - keyboard
  - localecfg
  # LUKS (voir partition.conf) : seule /home est chiffree, la racine porte
```

```
  # noEncrypt. luksbootkeyfile et initramfscfg ne trouvent donc aucune racine
  # chiffree et ne font rien ; initramfs regenere l initrd, ce que
  # shellprocess@osmo refait de toute facon. Ils restent pour un
  # partitionnement MANUEL ou l operateur chiffrerait lui-meme la racine.
  # La ligne crypttab de /home, elle, est reprise par crypthome-postinstall
  # (shellprocess@osmo) : "noauto", c est unlock-home qui pose la question.
  - luksbootkeyfile
  - initramfscfg
  - initramfs
  - users
  - displaymanager
  - networkcfg
  - hwclock
  - services-systemd
  - shellprocess@grub
  - grubcfg
  - bootloader
  - shellprocess@osmo
  - osmosupplements
  - contextualprocess@nvidia
  - umount
- show:
  - finished

branding: osmo

# On demande confirmation avant d'ecrire : cet installeur tourne sur un banc, a
# cote de disques qui portent des captures et des bases HLR.
prompt-install: true
dont-chroot: false
oem-setup: false
disable-cancel: false
disable-cancel-during-exec: true

# Ces deux cles sont REQUISES par calamares : sans elles, il journalise
# "Required settings.conf key ... is missing" a chaque lancement et retombe sur
# un defaut non ecrit. On les pose explicitement.
#   quit-at-end: false               -> l ecran "termine" reste, l operateur
#                                       lit le resultat avant de fermer.
#   hide-back-and-next-during-exec   -> pendant la copie (exec), on masque
#                                       Precedent/Suivant : rien a naviguer
#                                       tant que l installation tourne.
quit-at-end: false
hide-back-and-next-during-exec: true
```

3.163 /opt/GSM/osmo-operator/installer/calamares/branding/osmo/branding.desc

1917 octets, 45 lignes → 45 lignes

```
---
componentName: osmo

# Les chaines vides sont VOLONTAIRES : elles retirent les liens (support, notes
# de version) que l'ecran d'accueil afficherait sinon vers des pages qui
# n'existent pas pour ce projet.
strings:
    productName:         osmo-operator
    shortProductName:    osmo-operator
    version:             0.1-9
    shortVersion:        0.1-9
    versionedName:       osmo-operator v0.1-9 - banc GSM/EGPRS
    shortVersionedName:  osmo-operator v0.1-9
    bootloaderEntryName: osmo-operator
    productUrl:          ""
```

```
    supportUrl:          ""
    knownIssuesUrl:      ""
    releaseNotesUrl:     ""

images:
    productLogo:   "logo.png"
    productIcon:   "logo.png"
    productWelcome: "welcome.jpg"

slideshow: "show.qml"
# show.qml est ecrit pour l API 1 du diaporama (un Timer qui tourne du debut a
# la fin). Sans cette cle, Calamares journalise "Invalid or missing
# *slideshowAPI*" et devine.
slideshowAPI: 1

# ── LES COULEURS DE LA BARRE LATERALE ──────────────────────────────
# [2026-09-04] CES QUATRE NOMS ONT CHANGE EN CALAMARES 3.3. Les anciens
# (sidebarBackground, sidebarText, sidebarTextSelect, sidebarTextHighlight)
# etaient refuses un par un - "Unknown branding *style* entry" dans
# ~/.cache/calamares/session.log - et la barre retombait sur la palette Qt du
# bureau sombre : noir sur noir, les etapes INVISIBLES, seule l etape courante
# se voyait. Les seuls noms que 3.3 accepte sont ceux-ci, majuscule initiale
# (lus dans libcalamaresui : SidebarBackground, SidebarText,
# SidebarTextCurrent, SidebarBackgroundCurrent). "Current" = l etape en cours,
# ce que les anciens noms appelaient "Select"/"Highlight".
style:
    SidebarBackground:         "#0d1b2a"
    SidebarText:               "#ffffff"
    SidebarBackgroundCurrent:  "#1b98e0"
    SidebarTextCurrent:        "#13293d"
```

3.164 /opt/GSM/osmo-operator/installer/calamares/branding/osmo/show.qml

6963 octets, 161 lignes → 161 lignes

```
import QtQuick 2.0;
import calamares.slideshow 1.0;

// Le diaporama pendant la copie. Deux captures du banc en marche
// (slide-banc.png : le bureau, le tableau de bord et tmux ; slide-calypso-gdb.png :
// gdb attache au firmware Calypso emule) entre des diapos de texte qui
// reprennent la fiche de l image (osmo-operator-desktop.iso v0.1-9) : ce que
// la machine est en train de recevoir, poste par poste.
// [2026-09-03] Plus de compte osmocom sur le disque : la diapo "comptes" dit
// ce que l installation fait vraiment (voir modules/shellprocess-osmo.conf).
// [2026-10-01] Fiche v0.1-6 : base, taille, 2G, baseband, 4G, telephone,
// bureau, audio, installation. A refaire a chaque tag.
// [2026-10-24] Fiche v0.1-9 (.github/release-notes.md) : BTS classique par
// defaut, Magnetite, Calypso en radio de BTS, banc-rssi, vocodeur, rien de
// telecom au boot, start.sh. Taille et fichiers non remesures : retires.
Presentation {
    id: presentation
    Timer {
        interval: 9000
        running: true
        repeat: true
        onTriggered: presentation.goToNextSlide()
    }
    // Le fond du bureau live, copie en welcome.jpg par iso_modules/85-installeur-bureau.sh.
    Slide {
        Image {
            anchors.fill: parent
            anchors.margins: 12
            source: "welcome.jpg"
```

```
        fillMode: Image.PreserveAspectFit
        smooth: true
    }
}
Slide {
    Text {
        anchors.centerIn: parent
        horizontalAlignment: Text.AlignHCenter
        font.pixelSize: 20
        color: "#13293d"
        text: "osmo-operator-desktop.iso  v0.1-9\n\n" +
              "Un operateur GSM + LTE complet, sans SDR ni telephone :\n" +
              "la radio est emulee de bout en bout, jusqu'a la bande de base.\n" +
              "L'ARM du Calypso tourne sous QEMU (qosmo),\n" +
              "son DSP C54x execute la vraie ROM masque (c54x_exe).\n\n" +
              "Base Ubuntu 24.04, noyau 6.8 generic\n" +
              "ISO hybride (dd sur cle), Secure Boot shim > grub > noyau, option toram\n" +
              "8 Go de RAM et 4 coeurs minimum"
    }
}
Slide {
    Image {
        anchors.fill: parent
        anchors.margins: 12
        source: "slide-banc.png"
        fillMode: Image.PreserveAspectFit
        smooth: true
    }
    Text {
        anchors.bottom: parent.bottom
        anchors.horizontalCenter: parent.horizontalCenter
        font.pixelSize: 14
        color: "#13293d"
        text: "Le banc en marche : tableau de bord, mobile emule, chiffrement A5/1 au VTY"
    }
}
Slide {
    Text {
        anchors.centerIn: parent
        horizontalAlignment: Text.AlignHCenter
        font.pixelSize: 19
        color: "#13293d"
        text: "2G\n\n" +
              "osmo-stp, bsc, msc, hlr, sgsn, ggsn, pcu, mgw ; 2 cellules en LAC 1,\n" +
              "GPRS/EDGE, voix par osmo-sip-connector + Asterisk, SMS GSUP/SMPP.\n\n" +
              "Nouveau en 0.1-9\n\n" +
              "BTS classique par defaut : osmo-bts-trx + pont_dsp.py, layer1 + mobile sur la ROM DSP.\n" +
              "--ms-magnetite : Motorola C139 sous FreeCalypso Magnetite (firmware TI).\n" +
              "--bts : le Calypso devient la radio de la BTS ; --ts 1 | 2 | 2/1.\n" +
              "banc-rssi : l'appli rssi d'osmocom-bb sur un quatrieme Calypso.\n" +
              "Voix : --vocoder-rom (FR de la ROM, defaut) ou --gapk (hote)."
    }
}
Slide {
    Image {
        anchors.fill: parent
        anchors.margins: 12
        source: "slide-calypso-gdb.png"
        fillMode: Image.PreserveAspectFit
        smooth: true
    }
    Text {
        anchors.bottom: parent.bottom
        anchors.horizontalCenter: parent.horizontalCenter
        font.pixelSize: 14
        color: "#13293d"
```

```
            text: "gdb attache au firmware Calypso (layer1) qui tourne dans QEMU"
        }
    }
    Slide {
        Text {
            anchors.centerIn: parent
            horizontalAlignment: Text.AlignHCenter
            font.pixelSize: 19
            color: "#13293d"
            text: "4G\n\n" +
                  "Open5GS + srsRAN sur ZMQ, PLMN 001/01, chiffrement NAS nul pour l'analyse,\n" +
                  "CSFB par SGs entre la MME et osmo-msc.\n\n" +
                  "Telephone\n\n" +
                  "VM postmarketOS : data en 4G, voix et SMS en 2G sous une seule identite,\n" +
                  "oFono sur un modem AT virtuel - code de deverrouillage 147147."
        }
    }
    Slide {
        Text {
            anchors.centerIn: parent
            horizontalAlignment: Text.AlignHCenter
            font.pixelSize: 19
            color: "#13293d"
            text: "Bureau\n\n" +
                  "GNOME, Firefox (deb Mozilla), Wireshark, Linphone (poste 100), Conky.\n" +
                  "Le ? du dock ouvre l'aide : modes, modules, notebooks, depots.\n" +
                  "Reglages de toute la pile : http://127.0.0.1:8090\n" +
                  "Nouvel ecran de demarrage Plymouth : une antenne et ses cercles.\n\n" +
                  "Audio\n\n" +
                  "PulseAudio systeme, PipeWire pour le screencast seul,\n" +
                  "PCM gsm_out / gsm_in.\n\n" +
                  "Retire : avahi, Chromium ; sssd masque."
        }
    }
    Slide {
        Text {
            anchors.centerIn: parent
            horizontalAlignment: Text.AlignHCenter
            font.pixelSize: 19
            color: "#13293d"
            text: "Vos comptes\n\n" +
                  "L'utilisateur que vous venez de creer ouvre la session\n" +
                  "(sudoer : il pilote le banc avec sudo).\n" +
                  "root - le compte de travail, deverrouille,\n" +
                  "avec le mot de passe demande a l'installation.\n" +
                  "Le compte osmocom de la cle live n'est pas installe.\n\n" +
                  "/home chiffre (LUKS) si vous l'avez coche : au demarrage,\n" +
                  "« decrypt /home -passphrase- or Enter » - Entree a vide demarre sans.\n" +
                  "Rien de telecom ne demarre au boot (preset 80-osmo-operator) :\n" +
                  "banc, coeur Osmocom, BTS, Asterisk, oFono, Open5GS et srsRAN vous attendent."
        }
    }
    Slide {
        Text {
            anchors.centerIn: parent
            horizontalAlignment: Text.AlignHCenter
            font.pixelSize: 20
            color: "#13293d"
            text: "Pour demarrer le banc\n\n" +
                  "    cd /opt/GSM/osmo-operator\n" +
                  "    sudo ./start.sh --bts-classique --ms\n\n" +
                  "(ou le telephone rouge du dock ; sudo ./start.sh --help pour les autres modes)\n" +
                  "Le tableau de bord web ecoute deja sur cette machine."
        }
    }
}
```

3.165 /opt/GSM/osmo-operator/installer/calamares/jobs/osmosupplements/main.py

2881 octets, 70 lignes → 70 lignes

```
#!/usr/bin/env python3
# osmosupplements - les choix des supplements, de Calamares au premier demarrage.
#
# [2026-10-24] packagechooser@supplements (optionalmultiple, methode legacy) range les cles
# cochees dans packagechooser_supplements, separees par des virgules ; packagechooser@image
# range ghcr ou hub dans packagechooser_image. On les traduit en drapeaux d addition.sh,
# ecrits dans /etc/osmo-supplements de la cible, et on active osmo-supplements.service :
# tools/osmo-supplements-firstboot.sh les rejoue au premier demarrage, reseau monte.
import os

import libcalamares

DRAPEAUX = {
    "multi": "--multi",
    "opencl": "--opencl",
    "claude": "--claude",
    "firmwares": "--firmwares",
    "extras": "--extras",
}
IMAGES = {
    "ghcr": "ghcr.io/bbaranoff/osmo-operator/osmocom-nitb:latest",
    "hub": "bastienbaranoff/norf_gsm",
}
UNITE = "osmo-supplements.service"


def pretty_name():
    return "Supplements (premier demarrage)"


def _liste(cle):
    v = libcalamares.globalstorage.value(cle) or ""
    if isinstance(v, (list, tuple)):
        return [str(x).strip() for x in v if str(x).strip()]
    return [x.strip() for x in str(v).split(",") if x.strip()]


def run():
    root = libcalamares.globalstorage.value("rootMountPoint")
    if not root:
        return ("Supplements", "point de montage de la cible inconnu")
    choix = [c for c in _liste("packagechooser_supplements") if c in DRAPEAUX]
    conf = os.path.join(root, "etc/osmo-supplements")
    lien = os.path.join(root, "etc/systemd/system/multi-user.target.wants", UNITE)
    if not choix:
        libcalamares.utils.debug("osmosupplements : rien de coche")
        for f in (conf, lien):
            if os.path.lexists(f):
                os.remove(f)
        return None
    image = (_liste("packagechooser_image") or ["ghcr"])[0]
    with open(conf, "w") as f:
        f.write("# ecrit par l installeur (Calamares) - lu par tools/osmo-supplements-firstboot.sh\n")
        f.write('OSMO_SUPPLEMENTS="%s"\n' % " ".join(DRAPEAUX[c] for c in choix))
        if "multi" in choix:
            f.write('HUB_IMAGE="%s"\n' % IMAGES.get(image, IMAGES["ghcr"]))
        # le compte cree (page Utilisateurs) : au premier demarrage personne n est encore
        # connecte, addition.sh ne le trouverait pas (groupe docker, icones)
        compte = libcalamares.globalstorage.value("username")
        if compte:
            f.write('OSMO_COMPTE="%s"\n' % compte)
    if os.path.exists(os.path.join(root, "etc/systemd/system", UNITE)):
```

```
        os.makedirs(os.path.dirname(lien), exist_ok=True)
        if os.path.lexists(lien):
            os.remove(lien)
        os.symlink("/etc/systemd/system/" + UNITE, lien)
    else:
        libcalamares.utils.warning("osmosupplements : %s absente de la cible" % UNITE)
    libcalamares.utils.debug("osmosupplements : %s" % ", ".join(choix))
    return None
```

3.166 /opt/GSM/osmo-operator/installer/calamares/jobs/osmosupplements/module.desc

346 octets, 8 lignes → 8 lignes

```
---
# [2026-10-24] Ecrit les choix des pages « Supplements » et « Image operateur » dans
# /etc/osmo-supplements de la cible et active osmo-supplements.service (premier demarrage).
# Pose par iso_modules/85-installeur-bureau.sh dans le repertoire des modules systeme.
type: "job"
name: "osmosupplements"
interface: "python"
script: "main.py"
```

3.167 /opt/GSM/osmo-operator/installer/calamares/modules/bootloader.conf

815 octets, 18 lignes → 18 lignes

```
# GRUB, en EFI comme en BIOS.
#
# efiBootloaderId : le repertoire cree sous /boot/efi/EFI/. On garde "ubuntu"
# et PAS "osmo-operator" - ce n'est pas de la coquetterie : les binaires GRUB
# signes par Canonical (grub-efi-amd64-signed) ont /EFI/ubuntu compile comme
# prefixe. Les poser ailleurs, c'est un GRUB signe qui ne trouve pas sa
# configuration et tombe sur son invite "grub>" - le meme piege que celui
# documente dans build-iso.sh a l'etape Secure Boot, mais sur le disque cette
# fois. Le systeme installe reste ainsi amorcable Secure Boot, comme l'ISO.
---
efiBootLoader:      grub
grubInstall:        grub-install
grubMkconfig:       grub-mkconfig
grubCfg:            /boot/grub/grub.cfg
grubProbe:          grub-probe
efiBootMgr:         efibootmgr
efiBootloaderId:    ubuntu
installEFIFallback: true
```

3.168 /opt/GSM/osmo-operator/installer/calamares/modules/contextualprocess-mirror.conf

780 octets, 15 lignes → 15 lignes

```
# ============================================================================
#  contextualprocess@mirror - ecrit le miroir choisi dans la cible
# ============================================================================
#  Lit la cle globalstorage packagechooser_mirror (page packagechooser@mirror).
#  Tourne juste apres unpackfs, dans la cible (dontChroot: false) : tout apt
#  qui suit (le pilote NVIDIA, une mise a jour au premier boot) lit deja le bon
#  miroir. "keep" ne touche a rien. Les autres cles sont ecrites par
#  osmo-install a chaque lancement (m1, m2... dans l ordre des mesures) : un
```

```
#  sed sur /etc/apt/sources.list, et sur ubuntu.sources (deb822) s il existe.
---
dontChroot: false
timeout: 60
packagechooser_mirror:
    "keep":
        - "-/bin/true"
```

3.169 /opt/GSM/osmo-operator/installer/calamares/modules/contextualprocess-nvidia.conf

2371 octets, 32 lignes → 32 lignes

```
# ========================================================================
#  contextualprocess@nvidia - installe le pilote NVIDIA si l entree est choisie
# ========================================================================
#  Lit la cle globalstorage packagechooser_nvidia (page packagechooser@nvidia).
#  Valeur "nvidia" : on installe, DANS LA CIBLE, nvidia-driver-610.
#
#  [2026-09-04] UNE VERSION FIGEE, PLUS `ubuntu-drivers`. Le choix du pilote
#  passait par `ubuntu-drivers install`, qui interroge les depots et decide
#  seul : d une machine a l autre, et d une semaine a l autre, on n installait
#  pas la meme chose - et sur une cible fraiche, sans ubuntu-drivers-common ni
#  reseau au bon moment, il ne decidait rien du tout. Le banc demande
#  nvidia-driver-610, il est dans noble-updates/multiverse, il s installe comme
#  n importe quel paquet. C est ce que fait un operateur a la main :
#      sudo apt install nvidia-driver-610
#
#  Deux gardes restent : une carte NVIDIA doit etre vue (lspci, vendor 10de) et
#  le reseau doit repondre - sinon on le dit dans le journal et l installation
#  continue : un disque sans pilote proprietaire vaut mieux qu une installation
#  echouee.
#
#  /run de la cible est un tmpfs vide (mount.conf) : /etc/resolv.conf, lien vers
#  /run/systemd/resolve/stub-resolv.conf, y pointe dans le vide. On y ecrit des
#  resolveurs publics le temps de l apt ; le tmpfs disparait au demontage.
---
dontChroot: false
timeout: 1800
packagechooser_nvidia:
    "nvidia":
        - "-/bin/bash -c \"if ! lspci -n 2>/dev/null | grep -qi ' 10de:'; then echo '[nvidia] aucune carte NVIDIA
 ↪  (10de) : pilote non installe'; exit 0; fi; mkdir -p /run/systemd/resolve; [ -s
 ↪  /run/systemd/resolve/stub-resolv.conf ] || printf 'nameserver 1.1.1.1\\\\nnameserver 8.8.8.8\\\\n' >
 ↪  /run/systemd/resolve/stub-resolv.conf; export DEBIAN_FRONTEND=noninteractive; apt-get update -qq || { echo
 ↪  '[nvidia] pas de reseau : pilote non installe'; exit 0; }; if apt-get install -y nvidia-driver-610; then echo
 ↪  '[nvidia] pilote installe :'; dpkg -l 'nvidia-driver-*' 2>/dev/null | awk '/^ii/{print \\$2, \\$3}'; else echo
 ↪  '[nvidia] nvidia-driver-610 non installable (absent du miroir ?) - le pilote libre nouveau reste en place';
 ↪  fi\""
        - "-/bin/bash -c \"update-initramfs -u >/dev/null 2>&1 || true\""
    "none":
        - "-/bin/true"
```

3.170 /opt/GSM/osmo-operator/installer/calamares/modules/displaymanager.conf

144 octets, 6 lignes → 6 lignes

```
# GDM, et lui seul : c'est ce que ubuntu-desktop-minimal installe.
---
displaymanagers:
    - gdm
```

```
basicSetup:      false
sysconfigSetup:  false
```

3.171 /opt/GSM/osmo-operator/installer/calamares/modules/finished.conf

76 octets, 3 lignes → 3 lignes

```
---
restartNowMode: user-unchecked
restartNowCommand: "systemctl -i reboot"
```

3.172 /opt/GSM/osmo-operator/installer/calamares/modules/fstab.conf

183 octets, 8 lignes → 8 lignes

```
---
mountOptions:
    default:  defaults,noatime
    btrfs:    defaults,noatime,space_cache=v2
ssdExtraMountOptions:
    ext4:     discard
    btrfs:    discard
crypttabOptions: luks
```

3.173 /opt/GSM/osmo-operator/installer/calamares/modules/grubcfg.conf

1290 octets, 22 lignes → 22 lignes

```
# ============================================================================
#  grubcfg.conf - /etc/default/grub du systeme installe
# ============================================================================
#  overwrite: false - on complete le fichier de la cible au lieu de le
#  remplacer : les valeurs que dpkg y a mises (GRUB_CMDLINE_LINUX_DEFAULT du
#  paquet grub-common) restent.
---
overwrite: false
defaults:
    GRUB_TIMEOUT: 5
    GRUB_DISABLE_OS_PROBER: false
    # ── LUKS : sans cette ligne, une installation chiffree NE DEMARRE PAS ──
    # [2026-09-04] partition.conf ne declare qu UNE partition (racine, 100%) :
    # quand l operateur coche "Chiffrer le systeme", /boot part DANS le
    # conteneur LUKS avec le reste - il n y a pas de /boot en clair. GRUB doit
    # donc ouvrir le volume lui-meme pour lire son noyau, et il ne compile
    # cryptomount/cryptodisk dans son image que si grub-mkconfig voit ce "y".
    # Sans lui : "error: disk `cryptouuid/...' not found" puis l invite grub>.
    # GRUB 2.12 (noble) sait ouvrir le LUKS2 argon2id de luksGeneration.
    # Sur une installation NON chiffree la cle ne change rien : grub-mkconfig
    # ne trouve aucun volume chiffre sur le chemin d amorcage et n emet rien.
    GRUB_ENABLE_CRYPTODISK: "y"
```

3.174 /opt/GSM/osmo-operator/installer/calamares/modules/keyboard.conf

1096 octets, 20 lignes → 20 lignes

```
# ============================================================================
#  keyboard.conf - le clavier ecrit dans la cible
# ============================================================================
#  Sans ce fichier, calamares journalisait "No config file for keyboard found"
#  et posait un clavier par defaut (us) qui ne correspond pas au banc. On lui
#  donne un modele et une disposition de repli ; l'ecran clavier laisse toujours
#  l'operateur changer, ceci n'est que le point de depart.
# ----------------------------------------------------------------------------
---
# Modele de clavier generique 105 touches : le bon defaut pour un PC de banc.
keyboardModel: pc105

# Disposition de repli si l'ecran clavier n'est pas confirme. fr : ce banc est
# opere en France ; l'operateur peut choisir autre chose a l'ecran keyboard.
xOrgConfFileName:   "00-keyboard.conf"
convertedKeymapPath: "/usr/share/X11/xkb/symbols/pc"

# Ecrit aussi la configuration console (vconsole/keymap) dans la cible, pas
# seulement X : un banc se pilote souvent en tty.
writeEtcDefaultKeyboard: true
```

3.175 /opt/GSM/osmo-operator/installer/calamares/modules/locale.conf

430 octets, 9 lignes → 9 lignes

```
---
# La zone est fixee : ce banc est monte ici, et l installeur ne doit pas
# dependre du reseau pour une question a laquelle on connait la reponse.
region:  Europe
zone:    Paris
# Pas de bloc geoip : "style: none" n est pas une valeur reconnue - Calamares
# journalisait "GeoIP style *none* does not do anything" puis "GeoIP Style
# "none" is not recognized". L absence de la cle, elle, est le vrai "aucune
# recherche geoip".
```

3.176 /opt/GSM/osmo-operator/installer/calamares/modules/mount.conf

1239 octets, 32 lignes → 32 lignes

```
---
# [2026-09-04] "options" est une LISTE pour Calamares 3.3 (noble : 3.3.5) :
# main.py fait ",".join(options). Ecrit "options: bind" (la forme 3.2), il
# montait avec -o b,i,n,d, journalisait "Cannot mount /dev" et "Cannot mount
# /run/udev", et grub-install echouait dans la cible sur
#     failed to get canonical path of `/dev/sda1'
# - l installation se terminait sans chargeur de demarrage.
extraMounts:
    - device: proc
      fs:     proc
      mountPoint: /proc
    - device: sys
      fs:     sysfs
      mountPoint: /sys
    - device: /dev
      mountPoint: /dev
      options: [ bind ]
    - device: tmpfs
```

```
      fs:     tmpfs
      mountPoint: /run
    - device: /run/udev
      mountPoint: /run/udev
      options: [ bind ]
    # [2026-09-02] Le bind /var/cache/osmo-debs -> /run/osmo-debs a ete retire :
    # rien, dans build-iso.sh, n'a jamais peuple ce repertoire. Le montage
    # echouait a chaque installation (Calamares l'avale) et shellprocess@grub
    # ne trouvait rien a faire. grub2-common et les binaires signes sont poses
    # DANS le rootfs par build-iso.sh, qui refuse de produire une ISO sans eux.
extraMountsEfi:
    - device: efivarfs
      fs:     efivarfs
      mountPoint: /sys/firmware/efi/efivars
```

3.177 /opt/GSM/osmo-operator/installer/calamares/modules/packagechooser-image.conf

646 octets, 16 lignes → 16 lignes

```
---
# [2026-10-24] L origine de l image operateur, comme la seconde fenetre (boutons radio)
# d addition.sh. Sans effet si « Docker et multi-operateur SS7 » n est pas coche.
mode: required
method: legacy
labels:
    step: "Image operateur"
    step[fr]: "Image operateur"
default: ghcr
items:
    - id: ghcr
      name: "Le depot (GHCR)"
      description: "ghcr.io/bbaranoff/osmo-operator/osmocom-nitb:latest, publiee par la CI a chaque push. Ignore
↪   sans le supplement multi-operateur."
    - id: hub
      name: "Docker Hub"
      description: "bastienbaranoff/norf_gsm, la pile publiee a la main. Ignore sans le supplement
↪   multi-operateur."
```

3.178 /opt/GSM/osmo-operator/installer/calamares/modules/packagechooser-mirror.conf

1197 octets, 23 lignes → 23 lignes

```
# ============================================================================
#  packagechooser@mirror - le miroir Ubuntu du systeme installe
# ============================================================================
#  [2026-09-04] archive.ubuntu.com repondait a 3 Ko/s depuis le banc quand les
#  miroirs francais servaient en 0,1 s. Le systeme installe garde le miroir de
#  la cle live (celui que build-iso.sh avait mesure a la fabrication) - sauf
#  si l operateur en choisit un autre ici. Calamares ne mesure rien : c est
#  osmo-install (le lanceur du live, build-iso.sh) qui REECRIT cette page a
#  chaque lancement avec packaging/apt-mirror.sh --list : une entree par miroir
#  qui repond, son temps de reponse, le plus rapide PRE-SELECTIONNE, et une
#  entree "celui de la cle" pour ne rien changer. Le remplacement lui-meme est
#  dans contextualprocess-mirror.conf, apres unpackfs, dans la cible.
---
mode: required
method: legacy
labels:
    step: "Miroir Ubuntu"
```

```
    step[fr]: "Miroir Ubuntu"
default: keep
items:
    - id: keep
      name: "Celui de la cle live"
      description: "Aucune mesure disponible : le systeme installe garde le miroir de la cle. @@MIRROR_STATE@@"
```

3.179 /opt/GSM/osmo-operator/installer/calamares/modules/packagechooser-nvidia.conf

1672 octets, 31 lignes → 31 lignes

```
# ============================================================================
#  packagechooser@nvidia - le pilote NVIDIA dans les choix d installation
# ============================================================================
#  [2026-09-03] Une page "Pilotes graphiques" avec deux entrees : aucun pilote
#  supplementaire (defaut) ou le pilote proprietaire NVIDIA. Calamares ne sait
#  pas griser une entree selon le materiel ; c est osmo-install (le lanceur du
#  live, build-iso.sh) qui REECRIT ce fichier juste avant de lancer Calamares :
#    - carte NVIDIA vue par lspci  -> l entree NVIDIA est proposee, avec le nom
#                                     de la carte, et pre-selectionnee ;
#    - pas de carte                -> l entree reste, mais marquee "aucune carte
#                                     detectee", et l installation ne fera rien.
#  L installation elle-meme est dans contextualprocess-nvidia.conf : elle ne
#  part que si l entree est choisie ET qu une carte est la (lspci dans la cible),
#  et elle a besoin du reseau (les paquets viennent des depots Ubuntu).
#
#  [2026-09-04] LA VERSION EST FIGEE : nvidia-driver-610. Plus d arbitrage par
#  ubuntu-drivers - voir contextualprocess-nvidia.conf.
---
mode: optional
method: legacy
labels:
    step: "Pilotes graphiques"
    step[fr]: "Pilotes graphiques"
default: none
items:
    - id: none
      name: "Aucun pilote supplementaire"
      description: "Le pilote libre (nouveau) du noyau, comme sur la cle live."
    - id: nvidia
      name: "Pilote NVIDIA 610 (proprietaire)"
      description: "Installe nvidia-driver-610 depuis les depots Ubuntu (reseau requis). @@NVIDIA_STATE@@"
```

3.180 /opt/GSM/osmo-operator/installer/calamares/modules/packagechooser-supplements.conf

1496 octets, 27 lignes → 27 lignes

```
---
# [2026-10-24] Les supplements d addition.sh dans l installeur : les memes cinq choix
# que sa fenetre zenity (memes cles), en cases a cocher. Ils s installent au PREMIER
# DEMARRAGE, reseau monte (osmo-supplements.service), pas pendant Calamares : l image
# operateur fait ~11 Go et les firmwares compilent longtemps. Le job osmosupplements
# (installer/calamares/jobs) ecrit les choix dans /etc/osmo-supplements de la cible.
mode: optionalmultiple
method: legacy
labels:
    step: "Supplements"
    step[fr]: "Supplements"
items:
```

```
    - id: multi
      name: "Docker et multi-operateur SS7"
      description: "docker.io, l image operateur (~11 Go, origine choisie a la page suivante) et la topologie op1
↪  natif + op2/op3 docker + inter-STP."
    - id: opencl
      name: "OpenCL (calcul GPU)"
      description: "Runtime ICD, clinfo, le pilote de la carte detectee (Intel / Mesa-AMD, pocl en repli) et les
↪  outils de calcul GPU du banc."
    - id: claude
      name: "Claude Code"
      description: "La CLI de l assistant IA (installeur natif claude.ai/install.sh), lancee ensuite avec la
↪  commande claude."
    - id: firmwares
      name: "Firmwares telephone"
      description: "gnuarm (arm-elf-gcc 4.5.2), FreeCalypso Magnetite, osmocom-bb-trx, Fernvale MT6260,
↪  documentation SIMCom. Long : plusieurs Go et des compilations."
    - id: extras
      name: "Jeux et media"
      description: "Doom, Quake III (OpenArena), Kodi, YouTube (Firefox + uBlock Origin), Wireshark root,
↪  Linphone."
```

3.181 /opt/GSM/osmo-operator/installer/calamares/modules/partition.conf

5793 octets, 109 lignes → 109 lignes

```
# ============================================================================
#  partition.conf - le decoupage du disque cible
# ============================================================================
#  Sans ce fichier, le module partition tournait sur ses defauts internes et
#  journalisait trois avertissements ("initialSwapChoice absent",
#  "defaultFileSystemType absent", "defaultPartitionTableType absent"). Pire :
#  le defaut d'un partitionneur qui ne sait rien du contexte est le mauvais
#  pour un banc - il proposait le redimensionnement d'un Windows qui n'existe
#  pas. On lui dit ce qu'on veut : effacer et installer, en ext4, sur une table
#  GPT quand la machine demarre en UEFI.
# ----------------------------------------------------------------------------
---
# efiSystemPartition : le point de montage de la partition EFI. Doit
# correspondre a ce que bootloader/grubcfg attendent (GRUB EFI y depose
# grubx64/shimx64). /boot/efi est la convention Debian/Ubuntu.
efiSystemPartition:     "/boot/efi"

# Taille de l'ESP. 300 MiB laisse la place a shim + grub + un noyau de secours
# sans etre avare.
efiSystemPartitionSize: 300MiB
efiSystemPartitionName: EFI

# Le systeme de fichiers par defaut de la racine. ext4 : ce banc n'a besoin ni
# des instantanes de btrfs ni du journal de xfs, et ext4 est celui que
# fsck/grub/live-boot connaissent le mieux ici.
defaultFileSystemType:  "ext4"

# La table de partitions quand rien n'impose l'inverse : GPT en UEFI (le cas de
# cette ISO, qui demarre par shim signe), msdos en BIOS herite.
defaultPartitionTableType: gpt

# Le swap. "none" par defaut : un banc de mesure tourne en RAM et un fichier
# d'echange sur SSD n'apporte rien ; l'operateur peut toujours choisir
# "swapfile" dans la liste ci-dessous s'il installe sur une petite machine.
initialSwapChoice: none
userSwapChoices:
    - none
    - swapfile
```

```
# On autorise l'effacement complet ET l'installation manuelle. Pas de
# "alongside" (cohabitation) : un banc GSM n'est pas un poste bureautique a
# cote d'un autre OS, et l'option ne ferait qu'ouvrir une voie a l'erreur.
allowManualPartitioning: true

# ── LUKS : CE QUI EST CHIFFRE, C EST /home - PAS LA RACINE ──────────────────
# [2026-09-25] La case "Chiffrer le systeme" apparait dans les modes
# automatiques (effacer le disque, installer a cote, remplacer). Cochee, elle ne
# chiffre PAS tout : la racine porte "noEncrypt" ci-dessous, seule la partition
# /home passe en LUKS2.
#
# LE MODELE, en une phrase : tout ce qui est systeme reste lisible, tout ce qui
# tient au trousseau est chiffre. Le banc se pilote en root (VTY, netns,
# tcpdump) et un operateur doit pouvoir sortir une trace, relire un journal ou
# reparer la machine depuis une cle live sans rien dechiffrer. Ce qui n a rien a
# faire en clair, ce sont les SECRETS : trousseau, cookies, jetons, profils -
# et ceux-la vivent tous dans /home.
#
# CE QUE CELA SIMPLIFIE. Racine en clair = GRUB lit la racine directement, pas
# de /boot separe, pas de phrase de passe dans l initramfs, pas de
# /crypto_keyfile.bin (luksbootkeyfile constate qu aucune racine n est chiffree
# et ne fait rien). Le volume /home n est ouvert qu apres, par unlock-home.
#
# ET CE QUE CELA DEMANDE. Calamares ecrirait pour /home une ligne crypttab
# ordinaire : systemd demanderait la phrase de passe au demarrage et
# ATTENDRAIT. crypthome-postinstall (shellprocess@osmo) repasse crypttab et
# fstab en "noauto" et remplit /etc/default/crypthome, pour que ce soit
# unlock-home.service qui pose la question - avec un delai, et un demarrage qui
# se termine quelle que soit la reponse.
#
# En partitionnement MANUEL, rien de tout cela n est impose : le chiffrement se
# coche partition par partition, et crypthome-postinstall raccorde ce qu il
# trouve sous /home, chiffre ou non.
enableLuksAutomatedPartitioning: true
luksGeneration: luks2
# LVM : Calamares ne le propose pas en mode automatique ; il est disponible
# dans le partitionnement MANUEL (creer un groupe de volumes, y poser la racine,
# chiffrer le volume physique en LUKS si on veut LUKS-sur-LVM). lvm2 est dans
# le rootfs (build-iso.sh, bloc desktop) : l installeur, l initrd de la cible
# (initramfscfg + initramfs) et le systeme installe le comprennent.

# ── LE DECOUPAGE ────────────────────────────────────────────────────────────
# Deux partitions, et c est ce qui rend le chiffrement de /home possible : tant
# que /home etait un repertoire de la racine, il n y avait rien a chiffrer
# separement, et cocher la case aurait chiffre le systeme entier.
#
# LES TAILLES. Moitie-moitie, avec deux garde-fous. La racine porte /opt/GSM,
# les images docker et les .deb du banc - d ou un plancher a 25 GiB, au-dessus
# des 20 GiB que welcome.conf exige deja. /home porte les profils ET les images
# disque de pmbootstrap, qui se comptent en dizaines de gigaoctets : lui laisser
# le reste (100% = ce qui n a pas ete pris) evite d avoir a deviner.
#
# Sur un disque trop petit pour les deux, la racine prend tout et il n y a pas
# de /home separe : crypthome-postinstall le CONSTATE et le dit en clair plutot
# que de laisser croire a un banc chiffre.
partitionLayout:
    - name: "osmo-egprs"
      filesystem: "ext4"
      mountPoint: "/"
      size: 50%
      minSize: 25GiB
      # Le point du dispositif : la racine reste LISIBLE, meme quand
      # l operateur coche "chiffrer". Sans cette ligne, cocher la case
      # chiffrerait le systeme et on perdrait la reparation depuis une cle live.
      noEncrypt: true
    - name: "crypthome"
```

```
      filesystem: "ext4"
      mountPoint: "/home"
      size: 100%
      minSize: 5GiB
```

3.182 /opt/GSM/osmo-operator/installer/calamares/modules/shellprocess-grub.conf

1917 octets, 32 lignes → 32 lignes

```
# ========================================================================
#  shellprocess-grub - GRUB DOIT EXISTER DANS LA CIBLE AVANT LE MODULE bootloader
# ========================================================================
#  LE SYMPTOME. L'installation va jusqu'au bout de la copie, puis :
#
#      Le chargeur de demarrage n'a pas pu etre installe. La commande
#      grub-install --target=x86_64-efi --efi-directory=/boot/efi
#      --bootloader-id=ubuntu --force a renvoye le code d'erreur 127.
#
#  127 = commande introuvable. Rien dans le message ne le dit, et le disque
#  reste partitionne, rempli, et NON AMORCABLE.
#
#  LA CAUSE. Le module bootloader tourne avec dont-chroot: false, donc DANS LE
#  SYSTEME INSTALLE. Or ce systeme sort du squashfs, et le squashfs n'avait que
#  grub-common (grub-mkconfig, grub-probe) -- PAS grub2-common, qui fournit
#  grub-install. build-iso.sh installait bien grub-efi-amd64-bin, mais sur la
#  MACHINE QUI FABRIQUE L'ISO (ISO_HOST_PKGS), pour rendre l'image amorcable :
#  deux choses differentes, et la confusion ne se voit qu'ici.
#
#  build-iso.sh pose desormais ces paquets DANS le rootfs, avec un garde-fou qui
#  refuse de produire une ISO sans eux. Ce module ne repare plus rien : le
#  mecanisme "dpkg -i /run/osmo-debs/*.deb" a ete retire le 2026-09-02, parce
#  que rien n'a jamais rempli /var/cache/osmo-debs sur la cle - c'etait un
#  chemin mort. Il reste UN garde-fou, non fatal (prefixe "-") : dire dans le
#  journal, avant le module bootloader, si grub-install manque.
---
dontChroot: false
timeout: 600
script:
    # La preuve, pas l'esperance : si grub-install manque encore, on le DIT dans
    # le journal de l'installateur, la ou quelqu'un ira lire apres l'echec.
    - "-/bin/bash -c \"command -v grub-install >/dev/null || echo 'ATTENTION: grub-install absent de la cible - le
 ↪  module bootloader va echouer (code 127)' >&2\""
```

3.183 /opt/GSM/osmo-operator/installer/calamares/modules/shellprocess-osmo.conf

9385 octets, 119 lignes → 119 lignes

```
# ========================================================================
#  Ce qui distingue un systeme INSTALLE de la cle live
# ========================================================================
#  [2026-09-03] INVERSION DE LA REGLE. Jusqu ici ce module faisait ressembler
#  le disque a la cle : session sur root, autologin root en console et dans
#  GDM, un compte osmocom au mot de passe fixe. Le disque restait FIGE au live,
#  et ce que l on tapait dans l ecran "Utilisateurs" ne comptait pour rien.
#
#  Desormais ce sont les CARACTERISTIQUES DE L INSTALLATION qui font foi :
#    - l utilisateur est celui de l ecran "Utilisateurs" (${USER}), avec son
#      mot de passe, son autologin s il l a coche, son nom de machine ;
#    - root reste un compte de travail : deverrouille, mot de passe = celui
#      demande a l installation (users.conf, setRootPassword) ;
#    - le compte osmocom de la cle DISPARAIT (sauf si c est le nom choisi) ;
#    - les autologins root de la cle (console, GDM) et la regle PAM qui les
```

```
#      permettait sont retires : GDM ouvre sur ${USER}, comme sur tout Ubuntu.
#  Le banc, lui, se pilote toujours en root (sudo -i, ou login root) : les
#  lanceurs du bureau, les unites et /opt/GSM sont en place pour ${USER}, qui
#  est sudoer (users.conf, defaultGroups).
#
#  dontChroot: false - les commandes tournent DANS la cible montee, pas sur le
#  systeme live. Sans cela on modifierait le compte root de la cle USB.
#  timeout genereux : sur un disque lent, update-initramfs prend son temps.
#  ${USER} et ${ROOT} sont substitues par Calamares (module shellprocess).
---
dontChroot: false
timeout: 600
script:
    # root est le compte de travail : compte deverrouille. Son mot de passe est
    # CELUI TAPE dans l'ecran "Utilisateurs" (users.conf, setRootPassword).
    - "-/usr/bin/passwd -u root"

    # ── Plus de compte osmocom sur le disque ──────────────────────────
    # Il n existait que pour la cle. Sauf si l installateur a justement choisi
    # ce nom : alors c est SON compte, cree par le module users, on n y touche pas.
    - "-/bin/bash -c \"[ \\\"${USER}\\\" = osmocom ] || { pkill -u osmocom 2>/dev/null; userdel -r osmocom
↪   2>/dev/null; groupdel osmocom 2>/dev/null; rm -rf /home/osmocom; }\""

    # ── /home chiffre : raccorder ce que l installeur vient de produire ──
    # A LANCER AVANT la suppression d osmocom ? NON - apres. Ce script ecrit
    # CRYPTHOME_OWNER=${USER}, et c est justement parce que osmocom disparait
    # que le proprietaire du home chiffre doit devenir le compte de session.
    # Il ne chiffre rien : Calamares l a deja fait (partition.conf, la racine
    # porte noEncrypt et seule /home passe en LUKS2). Il repasse crypttab et
    # fstab en "noauto" pour que ce soit unlock-home qui pose la question au
    # demarrage, sans bloquer le boot, et pose le trousseau de root sur le
    # volume. Si /home n est pas chiffre, il le dit et laisse tout inerte.
    # Au demarrage : "oui" ouvre GDM sur ${USER} (persistant), "non" sur root,
    # ephemere, ses ecritures dans /tmp (unlock-home, session_root). Les
    # lignes GDM ci-dessous restent justes : c est la configuration de "oui",
    # celle qu unlock-home remet a chaque demarrage.
    - "-/usr/local/sbin/crypthome-postinstall ${USER}"

    # ── Les autologins root de la cle sont retires ────────────────────
    # Console : tty1 et ttyS0 ouvraient sur root sans demander (drop-ins poses
    # par build-iso.sh). Sur le disque, le login est celui d Ubuntu.
    - "-/bin/bash -c \"rm -f /etc/systemd/system/getty@tty1.service.d/10-autologin-root.conf
↪   /etc/systemd/system/serial-getty@ttyS0.service.d/10-autologin-root.conf\""
    - "-/bin/bash -c \"rmdir /etc/systemd/system/getty@tty1.service.d
↪   /etc/systemd/system/serial-getty@ttyS0.service.d 2>/dev/null || true\""
    # GDM : le module displaymanager (juste avant, dans settings.conf) a ecrit
    # l autologin de ${USER} s il a ete coche. Si la ligne AutomaticLogin=root
    # de la cle a survecu, elle prend le nom de l utilisateur installe.
    - "-/bin/bash -c \"[ -f /etc/gdm3/custom.conf ] && sed -i 's/^AutomaticLogin=root$/AutomaticLogin=${USER}/'
↪   /etc/gdm3/custom.conf || true\""
    # La regle PAM 'user != root' avait ete commentee sur la cle pour laisser
    # GDM ouvrir une session root. On la remet : c est celle d Ubuntu.
    - "-/bin/bash -c \"sed -i 's/^#\\\\(auth.*pam_succeed_if.so user != root quiet_success\\\\)/\\\\1/'
↪   /etc/pam.d/gdm-password /etc/pam.d/gdm-autologin 2>/dev/null || true\""

    # ── Le bureau de ${USER} : les memes lanceurs que la cle ──────────
    # Le home vient de /etc/skel, que build-iso.sh a rempli (lanceurs, favoris,
    # gnome-initial-setup-done). On s assure que ce qui a ete cree avant que le
    # skel ne soit lu (module users) ait bien les lanceurs, et le bon proprietaire.
    - "-/bin/bash -c \"h=\\$(getent passwd ${USER} | cut -d: -f6); [ -n \\\"\\$h\\\" ] && for d in Desktop Bureau;
↪   do [ -d /etc/skel/\\$d ] && mkdir -p \\$h/\\$d && cp -n /etc/skel/\\$d/*.desktop \\$h/\\$d/ 2>/dev/null; done;
↪   [ -n \\\"\\$h\\\" ] && chown -R ${USER}: \\$h || true\""

    # Le dashboard web : les deux unites, et l'oneshot arme au boot. Elles sont
    # prises dans le depot, qui voyage DANS la cible (/opt/GSM/osmo-operator) -
    # rien a telecharger, et une ISO qui n'aurait pas encore les unites dans son
```

```
    # squashfs produit quand meme un disque qui les a. Le certificat TLS est
    # fige au build (Dockerfile) : le premier demarrage n a rien a telecharger.
    - "-/bin/bash -c \"cp -f /opt/GSM/osmo-operator/services/osmo-egprs-web.service /etc/systemd/system/
↪   2>/dev/null || true\""
    - "-/bin/bash -c \"mkdir -p /etc/systemd/system/multi-user.target.wants && ln -sf
↪   /etc/systemd/system/osmo-egprs-web.service
↪   /etc/systemd/system/multi-user.target.wants/osmo-egprs-web.service\""

    # ── L'ICONE « INSTALLER » N'A PLUS RIEN A INSTALLER ──────────────────
    # Elle est posee sur le bureau de root et dans /etc/skel par build-iso.sh,
    # et elle a un sens sur la cle : c'est la premiere chose qu'on y cherche.
    # Sur le disque, elle est au mieux inutile, au pire dangereuse - un
    # double-clic ouvre Calamares sur la machine qu'on vient d'installer, avec
    # son propre disque dans la liste des cibles.
    #
    # osmo-install s'en protege (il refuse de partir sans medium live), mais un
    # refus n'est pas une reponse : l'icone ne doit pas etre la. On l'efface de
    # tous les bureaux et de leurs deux noms - "Desktop" et "Bureau" existent
    # tous les deux dans l'image, c'est la locale de la session qui decide
    # lequel est affiche, et n'en effacer qu'un laisserait l'autre en place.
    #
    # /usr/share/applications aussi : sans ca l'entree reste dans le menu des
    # applications, ou elle est plus discrete mais tout aussi active. Le binaire
    # /usr/local/bin/osmo-install, lui, RESTE - il ne fait rien de mal sur un
    # disque, et il sert a installer une AUTRE machine depuis celle-ci si on
    # rebranche une cle.
    - "-/bin/bash -c \"rm -f /root/Desktop/osmo-install.desktop /root/Bureau/osmo-install.desktop
↪   /usr/share/applications/osmo-install.desktop /etc/skel/Desktop/osmo-install.desktop
↪   /etc/skel/Bureau/osmo-install.desktop\""
    - "-/bin/bash -c \"for h in /home/*; do rm -f \\$h/Desktop/osmo-install.desktop
↪   \\$h/Bureau/osmo-install.desktop; done\""
    # L'entree du paquet calamares, elle, etait seulement masquee (NoDisplay).
    # Sur le disque on la retire : elle lance /usr/bin/calamares en direct, donc
    # sans aucun des garde-fous d'osmo-install.
    - "-/bin/bash -c \"rm -f /usr/share/applications/calamares.desktop\""
    - "-/bin/bash -c \"update-desktop-database /usr/share/applications >/dev/null 2>&1 || true\""

    # ── L UI smartphone : la config pmbootstrap de ${USER} ───────────────
    # [2026-09-08] pmbootstrap tourne sous le compte de session (il refuse
    # root) et lit ~/.config/pmbootstrap_v3.cfg. 88-lte-pmos.sh a pose le
    # gabarit dans /etc/skel (pmbootstrap_v3.cfg.osmo, avec @WORK@/@APORTS@) :
    # on l instancie ici avec le home de ${USER}. osmo-pmos-build ferait de
    # meme au premier lancement ; le faire ici rend l icone utilisable tout de
    # suite. Le dossier de travail : ~/.local/var/pmbootstrap (defaut pmbootstrap).
    - "-/bin/bash -c \"h=\\$(getent passwd ${USER} | cut -d: -f6); t=/etc/skel/.config/pmbootstrap_v3.cfg.osmo; [
↪   -n \\\"\\$h\\\" ] && [ -f \\$t ] && [ ! -f \\$h/.config/pmbootstrap_v3.cfg ] && mkdir -p \\$h/.config && sed
↪   \\\"s#@WORK@#\\$h/.local/var/pmbootstrap#g; s#@APORTS@#\\$h/.local/var/pmbootstrap/cache_git/pmaports#g\\\"
↪   \\$t > \\$h/.config/pmbootstrap_v3.cfg && chown -R ${USER}: \\$h/.config || true\""
    - "-/bin/bash -c \"chmod -R a+rX /opt/user_interface 2>/dev/null || true\""

    # live-boot n'a plus rien a faire sur un disque : ses scripts d'initramfs
    # chercheraient un medium live a chaque demarrage et allongent le boot.
    - "-/bin/bash -c \"apt-get purge -y live-boot live-boot-initramfs-tools >/dev/null 2>&1 || true\""
    - "-/bin/bash -c \"update-initramfs -u >/dev/null 2>&1 || true\""
```

3.184 /opt/GSM/osmo-operator/installer/calamares/modules/unpackfs.conf

2100 octets, 37 lignes → 37 lignes

```
# Ou est le systeme a deverser sur le disque.
#
# ── POURQUOI PAS /run/live/medium/live/filesystem.squashfs ──────────────
# C'etait la valeur d'avant, et c'est le chemin que live-boot donne EN MODE
# NORMAL : le medium est monte sur /run/live/medium, et le squashfs y est a sa
```

```
# place d'origine, dans live/.
#
# Ce chemin n'existe pas dans les autres modes de demarrage, et l'ISO en offre
# deux autres dans son menu :
#
#   « en RAM » (toram=filesystem.squashfs) : live-boot ne recopie pas
#     l'arborescence du medium, il copie LE FICHIER seul a la racine d'un tmpfs
#     qu'il deplace ensuite sur /run/live/medium. Le squashfs est alors en
#     /run/live/medium/filesystem.squashfs - sans le "live/". Le systeme
#     demarre tres bien (live-boot sait ou il l'a mis) et l'installation, elle,
#     echouait : « Mauvaise configuration unpackfs », ou le refus prealable
#     d'osmo-install.
#   « persistant » : live-boot balaie les disques et peut monter la cle sous
#     /run/live/persistence/sdX, laissant /run/live/medium vide.
#
# ── CE QU'ON LIT A LA PLACE ──────────────────────────────────────────
# /run/osmo-install-src/live/filesystem.squashfs est un chemin que
# /usr/local/bin/osmo-install GARANTIT avant de lancer Calamares : il retrouve
# le squashfs reellement monte (par le loop qui porte la racine en cours
# d'execution, pas par un chemin suppose) et le bind-monte la. Le chemin ne
# depend donc plus d'aucun mode de demarrage - c'est le lanceur qui absorbe la
# difference, une fois, a un endroit.
#
# COROLLAIRE : lancer /usr/bin/calamares directement ne marche plus. C'est
# voulu et c'etait deja le cas : son entree de menu est en NoDisplay sur la cle
# et supprimee a l'installation, parce qu'elle court-circuite aussi le
# nettoyage des cibles restees montees. On passe par osmo-install.
---
unpack:
    - source: /run/osmo-install-src/live/filesystem.squashfs
      sourcefs: squashfs
      destination: ""
```

3.185 /opt/GSM/osmo-operator/installer/calamares/modules/users.conf

3058 octets, 66 lignes → 66 lignes

```
# ============================================================================
#  Le compte cree pendant l'installation
# ============================================================================
#  LE MODELE DE CETTE IMAGE : root est le compte de travail par defaut, et
#  osmocom est un SECOND compte, non privilegie, sudoer, qu'on choisit
#  explicitement. C'est vrai sur la cle live comme sur le systeme installe.
#
#  Cet ecran est donc la ou l'on "demande si on veut changer l'utilisateur par
#  defaut" : la personne qui installe peut creer le compte qu'elle veut. Quoi
#  qu'elle reponde, l'instance shellprocess@osmo qui tourne juste apres
#  RETABLIT les deux garanties de l'image - autologin sur root, et osmocom
#  present et sudoer. Un choix personnalise s'ajoute, il ne remplace pas.
#
#  setRootPassword: true - sans mot de passe root, "su" est inutilisable sur un
#  systeme dont root EST le compte de travail. doAutologin est laisse a true :
#  l'installeur pose l'autologin sur le compte cree, et shellprocess le bascule
#  ensuite sur root.
---
defaultGroups:
    - adm
    - cdrom
    - sudo
    - dip
    - plugdev
    - lpadmin
    - dialout
    - audio
```

```
    - video
    # [2026-09-04] pulse-access : le son du banc passe par un PulseAudio
    # SYSTEME (osmo-pulse.service). Son socket accepte l'anonyme
    # (auth-anonymous=1, pose par iso_modules/82-services.sh), mais ce groupe
    # est le chemin documente de PulseAudio en mode systeme : sans lui, la
    # moindre reprise du system.pa d'origine rendrait le compte installe muet.
    - pulse-access
    # [2026-09-05] docker : le Conky (tools/conky-osmo-status.sh) et l'encart
    # (tools/osmo-panel.py) tournent sous CE compte, et sondent chaque
    # operateur en conteneur par "docker exec". Sans ce groupe, chaque sonde
    # rend "permission denied while trying to connect to the docker API" - que
    # le code lit comme "operateur arrete". Mesure sur le banc : conky annoncant
    # "banc a l arret", sections Coeur GSM / Radio / Abonnes VIDES, pendant que
    # les trois operateurs tournaient. Une sonde qui confond "absent" et
    # "invisible" est une sonde qui ment (tools/osmo-op.sh le dit deja).
    # Le groupe n'existe qu'une fois docker installe (addition.sh) ; Calamares
    # ignore silencieusement un groupe absent, et addition.sh rattrape le
    # compte de la session de son cote (_docker_groupe_session).
    - docker

autologinGroup:  autologin
doAutologin:     true
sudoersGroup:    sudo
setRootPassword: true
doReusePassword: false

# Un banc n'est pas une machine de production : on n'impose pas de politique de
# mot de passe, elle ne ferait que faire choisir "azerty" plus lentement.
passwordRequirements:
    nonempty:  false
    minLength: -1

user:
    shell: /bin/bash
    # root seulement : osmocom n est plus reserve. Le compte osmocom de la cle
    # est retire du disque par shellprocess-osmo.conf, sauf si c est le nom
    # choisi ici - il devient alors le compte de l utilisateur, tout simplement.
    forbidden_names:
        - root
```

3.186 /opt/GSM/osmo-operator/installer/calamares/modules/welcome.conf

253 octets, 14 lignes → 14 lignes

```
---
showSupportUrl:      false
showKnownIssuesUrl:  false
showReleaseNotesUrl: false
requirements:
    requiredStorage:  20
    requiredRam:      2
    check:
        - storage
        - ram
        - root
    required:
        - storage
        - root
```

3.187 /opt/GSM/osmo-operator/instances/instance.env.example

5219 octets, 173 lignes → 173 lignes

```
# instances/instance.env.example — patron des reglages PAR INSTANCE Calypso (c54x_exe/run.sh).
# Genere par tools/globals-complet.py ; la page web (tools/osmo-reglages.py) ecrit instance-N.env :
#   1 = la radio de la BTS (TRX Calypso, --bts) ou le mobile sur le pont (--bts-classique)
#   2 = le mobile QEMU par l air (--bts --ms, --ms-magnetite)
#   3 = le 2e TRX Calypso, TS1 (--ts 2)
# run.sh lit instances/instance-$INSTANCE.env au demarrage, sous la forme : "${VAR:=valeur}" :
# l environnement (start-native.sh, ligne de commande) garde la priorite, la valeur du fichier vient
# ensuite, puis le defaut de run.sh. Valeur VIDE = defaut.

# ── /opt/GSM/c54x_exe/run.sh ──────────────────────────────

# defaut : dsp   (run.sh:52)
# choix : dsp | grgsm
MODE=""
# defaut : 0   (run.sh:53)
# choix : 0 | 1
PONT=""
# defaut : /tmp/osmocom_l2_$INSTANCE   (run.sh:68)
L2_SOCK=""
# defaut : /tmp/qemu-monitor-pont$INSTANCE.sock   (run.sh:69)
MONITOR=""
# defaut : /tmp/calypso_dsp$INSTANCE.sock   (run.sh:70)
DSP_SOCK=""
# defaut : $((1233 + INSTANCE))   (run.sh:71)
GDB_STUB=""
# defaut : $((44443 + INSTANCE))   (run.sh:72)
GDB_TELNET=""
# defaut : none   (run.sh:73)
PANNEAU_LOGS=""
# defaut : $((6702 + 10 * (INSTANCE - 1) + (INSTANCE >= 3 ? 30 : 0)))   (run.sh:76)
CALYPSO_BSP_PORT=""
# defaut : /tmp/calypso_keys$INSTANCE.sock   (run.sh:77)
CALYPSO_KEYS_SOCK=""
# defaut : 1   (run.sh:78)
# choix : 0 | 1
AIR=""
# defaut : /opt/GSM/osmo-operator/pont/air.py   (run.sh:81)
AIR_PY=""
# defaut : /opt/GSM/qosmo   (run.sh:82)
QOSMO=""
# defaut : $QOSMO/build/qemu-system-arm   (run.sh:83)
QEMU=""
# defaut : 0   (run.sh:90)
# choix : 0 | 1
OSMO_TRX=""
# defaut : (vide)   (run.sh:91)
FIRMWARE_APP=""
# defaut : 0   (run.sh:106)
# choix : 0 | 1
OSMO_TRX_TS1=""
# defaut : 3   (run.sh:107)
OSMO_TRX_TS1_INSTANCE=""
# defaut : 1   (run.sh:108)
OSMO_TRX_PHONE=""
# defaut : $(awk '/^ *bts 0$/{f=1} f&&/base_station_id_code/{print $2; exit}' /etc/osmocom/osmo-bsc.cfg
 ↪  2>/dev/null)   (run.sh:124)
OSMO_TRX_BSIC=""
# defaut : /opt/GSM/osmocom-bb-transceiver/src/host/layer23/src/transceiver/transceiver   (run.sh:125)
TRANSCEIVER=""
# defaut : $([ "$OSMO_TRX_TS1" = 1 ] && echo 0 || echo 1)   (run.sh:132)
# choix : 0 | 1
OSMO_TRX_TCH=""
# defaut : 0x03   (run.sh:134)
OSMO_TRX_TS_MASK=""
# defaut : 0x03   (run.sh:135)
OSMO_TRX_TX_MASK=""
```

```
# defaut : 2   (run.sh:136)
CALYPSO_TRX_TX_SLOTS=""
# defaut : --no-sync -e 3   (run.sh:146)
OSMO_TRX_ARGS=""
# defaut : 1   (run.sh:149)
# choix : 0 | 1
MONTANT_ROM_TX_CHAQUE=""
# defaut : 1   (run.sh:153)
# choix : 0 | 1
CALYPSO_BSP_TRX=""
# defaut : -11   (run.sh:157)
CALYPSO_BSP_TRX_DELTA=""
# defaut : $_FW/layer1.highram.elf   (run.sh:161)
FIRMWARE_ELF=""
# defaut : ${FIRMWARE_ELF%.elf}.bin   (run.sh:162)
FIRMWARE_BIN=""
# defaut : (vide)   (run.sh:171)
FIRMWARE_FLASH=""
# defaut : 0   (run.sh:180)
# choix : 0 | 1
CALYPSO_PACER_RATTRAPAGE=""
# defaut : 1   (run.sh:181)
# choix : 0 | 1
CALYPSO_PONT_ARM_LOCK=""
# defaut : 1   (run.sh:184)
# choix : 0 | 1
CALYPSO_BSP_SB_FIN_TRAME=""
# defaut : 1   (run.sh:185)
# choix : 0 | 1
CALYPSO_BSP_NB_MARGE=""
# defaut : (vide)   (run.sh:188)
PONT_CAN_NB_TOA=""
# defaut : 0.5   (run.sh:193)
CALYPSO_BSP_SB_DEC=""
# defaut : 1   (run.sh:194)
# choix : 0 | 1
CALYPSO_BSP_BRUIT_FOND=""
# defaut : $AIR   (run.sh:199)
CALYPSO_BSP_RATTRAPE=""
# defaut : 1   (run.sh:200)
# choix : 0 | 1
CALYPSO_RACH_VERS_AIR=""
# defaut : 0   (run.sh:205)
# choix : 0 | 1
PONT_UL_TN_DEFAUT=""
# defaut : 0   (run.sh:209)
# choix : 0 | 1
AIR_B_FN_DELTA=""
# defaut : 0   (run.sh:210)
# choix : 0 | 1
MONTANT_RACH_DIFFERE=""
# defaut : $( . /etc/osmocom/radio-plan.env 2>/dev/null; echo "${PLAN_ARFCN:-}")   (run.sh:211)
CALYPSO_FAUX_AIR=""
# defaut : ti925t   (run.sh:213)
QEMU_CPU=""
# defaut : c139   (run.sh:214)
CALYPSO_LCD=""
# defaut : /opt/GSM/fc-host-tools/rvinterf/lowlevel/rvinterf   (run.sh:215)
RVINTERF_BIN=""
# defaut : /opt/GSM/osmocom-bb/src/host/osmocon/osmocon   (run.sh:217)
OSMOCON=""
# defaut : $(command -v mobile || echo /usr/local/bin/mobile)   (run.sh:218)
MOBILE=""
# defaut : $HERE/mobile_pont.cfg   (run.sh:219)
MOBILE_CFG=""
# defaut : /opt/GSM/osmo-operator/pont/pont.py   (run.sh:224)
```

```
PONT_PY=""
# defaut : (vide)   (run.sh:239)
RT_FIFO=""
# defaut : (vide)   (run.sh:240)
RT_CPUS=""
# defaut : (vide)   (run.sh:241)
RT_RENICE_AUTRES=""
# defaut : 1   (run.sh:263)
# choix : 0 | 1
GDB=""
# defaut : $QOSMO/tools/gdb-telnet.py   (run.sh:266)
GDB_TELNET_PY=""
# defaut : 120000   (run.sh:269)
INSNS=""
# defaut : 1   (run.sh:276)
# choix : 0 | 1
LOCKSTEP=""
# defaut : -v   (run.sh:278)
VERB=""
# defaut : none   (run.sh:279)
IQ=""
# defaut : 30000   (run.sh:280)
AMP=""
# defaut : 1   (run.sh:294)
# choix : 0 | 1
PONT_AIRREC=""
# defaut : 4194304   (run.sh:385)
QEMU_LOG_BUFFERED=""
# defaut : 64   (run.sh:422)
CALYPSO_PONT_RETRY_DIV=""
# defaut : 10   (run.sh:722)
JOURNAUX_GARDES=""
# defaut : (vide)   (run.sh:763)
BRUIT_MODE=""
# defaut : (vide)   (run.sh:764)
BRUIT_SNR_DB=""
# defaut : 16702   (run.sh:765)
BRUIT_PORT_DSP=""
# defaut : 14730   (run.sh:766)
BRUIT_PORT_GRGSM=""
# defaut : $BRUIT_SNR_DB   (run.sh:811)
CALYPSO_BSP_SNR_DB=""
```

3.188 /opt/GSM/osmo-operator/iso_modules/00-options.sh

24689 octets, 391 lignes → 391 lignes

```
#!/bin/bash
# iso_modules/00-options.sh - options de la ligne de commande, suite Ubuntu, root
# Source par build-iso.sh, dans l ordre des numeros : meme shell, memes
# variables, memes fonctions. Ne s execute pas seul. `return` en tete de
# module = "rien a faire ici" (c est ainsi que --arm saute une etape).

NO_CACHE=""
# ── WAN : jamais par defaut. Avec --wan, l'ISO EST un noeud du WAN ──────────
# La table est figee dans /etc/osmo-wan.conf de l'image ; au demarrage
# start-native.sh la voit (WAN_AUTO=1) et l'applique sans qu'on repasse --wan.
#
# UNE SEULE ISO pour les N machines : chacune se reconnait a son IP dans la
# table (wan_nodes_detect_self). --wan-id ne sert qu'a forcer un noeud quand
# l'IP ne suffit pas (DHCP non fixe, NAT).
ISO_WAN=0
ISO_WAN_NODES=""
ISO_WAN_ID=""
```

```
ISO_WAN_OPS=1

# ── ROLE DE L'IMAGE ──────────────────────────────────────────────────────
# Une seule chaine de construction, deux produits :
#
#   --role operator --node N   →  osmo-operator-N.iso
#       Un noeud du WAN : coeur GSM complet, ses point codes a lui, son ASP
#       attache au hub. C'est l'image historique, plus l'identite de noeud.
#
#   --role interstp            →  interstp.iso
#       Le hub SS7 : PC 0.0.0, aucun operateur, il ne fait QUE router du M3UA
#       entre les noeuds. C'est ce qui manquait pour que le SS7 traverse le WAN.
#
# Sans --role, rien ne change : on produit l'ISO d'avant, sous son nom d'avant.
ISO_ROLE=""
ISO_ROLE_GIVEN=0
ISO_NODE=""
# ── VARIANTE LITE ────────────────────────────────────────────────────────
#   --lite  →  osmo-operator-lite.iso
# Le meme noeud, construit depuis l'image d'execution elaguee (Dockerfile.lite)
# au lieu de l'image de construction : les ateliers de compilation de /opt/GSM -
# gnuradio, gr-gsm, libosmo*, osmo-*, les objets de qemu - ne partent pas dans
# le squashfs. Ce qui TOURNE est identique, aux fichiers pres qui n'ont servi
# qu'a compiler. Les trois depots, eux, restent entiers : c'est la regle de
# cette image, pas une exception (voir Dockerfile.lite).
ISO_LITE=0
# ── VARIANTE DESKTOP ─────────────────────────────────────────────────────
#   --desktop  →  osmo-operator-desktop.iso
# La meme image, plus un bureau GNOME (ubuntu-desktop-minimal), wireshark en
# fenetre et linphone-desktop. Elle existe pour ce qui ne se pilote pas au VTY :
# lire une capture GSMTAP dans wireshark plutot qu en tshark, passer un appel
# SIP a la main sur le coeur qu on vient de demarrer, ouvrir les outils Qt de
# gr-gsm (grgsm_livemon). Le rootfs grossit d environ 2,5 Go : c est pourquoi
# c est une VARIANTE, et non un ajout aux images existantes.
ISO_DESKTOP=0
# --arm : les cartes de armbian/build-armbian.sh (--board=)
ISO_ARM_BOARD="${OSMO_ISO_BOARD:-all}"
# ── --all : TOUTES les images en une passe ──────────────────────────────
# Sans argument, le script construit deja les trois images historiques (hub,
# noeud, noeud lite). --all y ajoute la desktop : c est la seule facon de tout
# sortir d un coup, sans avoir a relancer une quatrieme fois a la main. Il ne
# change rien au comportement par defaut - la desktop pese ~2,5 Go de plus et
# n a pas a s imposer a qui ne l a pas demandee.
ISO_ALL=0
# ── LE PULL EST LE DEFAUT (amd64) ────────────────────────────────────────
# [2026-09-14] ISO_SKIP_BUILD valait 0 : toute ISO partait d un build.sh local,
# soit 1h23 de compilation pour refabriquer a l identique une image deja
# publiee. Le defaut est desormais l inverse - on TIRE l image, et build.sh ne
# tourne que si aucune image publiee ne convient (voir iso_docker_build dans
# 21-docker-build.sh, qui essaie les references dans l ordre).
#
# amd64 SEULEMENT : rien n est publie en arm64, une passe --arm compile donc
# toujours. Un --skip-build explicite reste honore sur les deux architectures.
#
# --skip-build[=REF] garde son sens : il rend le pull OBLIGATOIRE (plus de
# repli sur build.sh, un echec est une erreur) et, avec =REF, impose la
# reference au lieu de la laisser deduire de l empreinte du depot.
ISO_SKIP_BUILD="${OSMO_ISO_SKIP_BUILD:-1}"
# Donne sur la ligne de commande, ou seulement le defaut ? Cette distinction
# decide de ce qui arrive quand le pull echoue : demande, c est fatal ; par
# defaut, on retombe sur la construction locale.
ISO_SKIP_BUILD_GIVEN=0
# [2026-10-10] La PREMIERE image essayee : l image Docker Hub, taguee
# osmocom-nitb. Elle ne suit PAS le commit - iso_docker_build le dit en clair
# avant de s en servir ; GHCR (empreinte, puis :latest) ne vient qu apres.
ISO_PULL_IMAGE="${OSMO_ISO_PULL_IMAGE:-bastienbaranoff/norf_gsm:latest}"
```

```
# --build-docker : l INVERSE de --skip-build, et il gagne. Sert quand quelque
# chose en amont pousse un --skip-build qu on ne veut pas ici : la CI qui tire
# l image de base depuis GHCR, un OSMO_ISO_PULL_IMAGE qui traine dans
# l environnement, un alias. Avec lui, l image est construite localement par
# build.sh, point. Equivalent en variable : OSMO_ISO_BUILD_DOCKER=1.
ISO_FORCE_BUILD="${OSMO_ISO_BUILD_DOCKER:-0}"
# --without-debs : aucune .deb NE VOYAGE dans l image. Par defaut l etape 7c
# (71-debs-banc.sh) fabrique les paquets du banc - osmo-operator, pont,
# qemu-calypso, calypso-firmware - et les range dans /var/cache/osmo-debs du
# rootfs, pour qu une machine installee puisse refaire le banc sans docker ni
# git. C est du poids mort pour qui n en fera jamais rien, et une ISO live tient
# en RAM. Cette option les supprime, et force ISO_EMBED_DEBS=0 pour que les
# paquets du build docker (etape 5) ne restent pas non plus : ils servent
# toujours a POSER la pile dans le rootfs, ils n y sejournent plus.
# Equivalent en variable : OSMO_ISO_WITHOUT_DEBS=1.
ISO_WITH_DEBS=1
[ "${OSMO_ISO_WITHOUT_DEBS:-0}" = "1" ] && ISO_WITH_DEBS=0

# ── ARCHITECTURE : amd64 (ISO PC) ou arm64 (image SD Raspberry Pi 4) ──────────
# --arm : la MEME pile, construite pour arm64 sur une base ARMBIAN 24.04 (le
# noyau bcm2711 et le BSP rpi4b d Armbian, depot apt.armbian.com), livree en
# IMAGE SD (.img : une partition FAT pour le firmware du Pi, une ext4
# persistante pour le systeme), pas en ISO live. Tout se fabrique sur l hote x86 : l image docker par buildx
# --platform linux/arm64 (qemu-user-static emule aarch64, c est LENT : comptez
# une dizaine d heures au premier build, le cache .deb rend les suivants
# courts), le rootfs par debootstrap --foreign puis chroot sous binfmt. Ni
# bureau (--desktop), ni installateur, ni passe --all sur cette cible.
ISO_ARCH=amd64
# Defaut : le hub du banc = le conteneur osmo-inter-stp, sur le backbone docker
# (INTER_STP_IP de start-docker.sh). Les VM le joignent par la route LAN -> docker
# (network/setup-docker-lan-route.sh). Le host-only VirtualBox (192.168.56.1)
# reste possible, mais via --hub-ip.
ISO_HUB_IP="172.20.0.10"
# ── Table WAN par defaut : le banc ──────────────────────────────────────────
# [2026-09-03] Table remise a jour. Format : <noeud>:<IP>:<indicatif>
#   noeud 1  192.168.1.123  la VM (cette ISO)   indicatif 11
#   noeud 2  172.20.0.12  osmo-operator-2     indicatif 22   (conteneur)
#   noeud 3  172.20.0.13  osmo-operator-3     indicatif 33   (conteneur)
#   hub      172.20.0.10  osmo-inter-stp                    (hors table)
# Plan de numerotation : MSISDN = <noeud>00<op><ms> (100101 = noeud 1, op 1,
# MS 1 ; 200101 = noeud 2...) - voir osmo_msisdn dans generate_configs.sh.
# L indicatif (11, 22, 33) est le prefixe compose pour joindre un autre noeud.
# Elle sert quand --wan-nodes n'est pas donne. Sans defaut, une construction
# sans terminal - la CI - s'arretait a l'etape 7b sur une question que personne
# ne lisait : "pas de terminal : renseignez WAN_NODES / WAN_NODE_ID / WAN_OPS".
# [2026-09-14] Le noeud 1 passe de 192.168.1.2 a 192.168.1.123 : c'est
# l'adresse de la VM du banc sur le LAN. La MEME valeur est reprise dans
# start-interstp.sh (WAN_NODES_DEFAULT) et start-docker.sh - une table lue d'un cote
# doit ressembler a celle qu'on ecrit de l'autre, sinon les point codes ne
# correspondent plus aux AS declares. Changer l'un sans les autres se paie au
# moment ou le SS7 traverse.
ISO_WAN_NODES_DEFAULT="1:192.168.1.123:11 2:172.20.0.12:22 3:172.20.0.13:33"

# ── AUCUNE 2G NE PART AU DEMARRAGE ──────────────────────────────────────────
# [2026-09-13] osmo-multi.service avait ete ACTIVE par defaut, pour n'avoir
# plus rien a faire apres addition.sh : l'unite etant Conditionnee a la
# topologie /etc/osmocom/osmo-multi.conf, on la croyait sans consequence tant
# que le supplement multi-operateur n'etait pas installe.
#
# [2026-09-14] ON REVIENT A 0, ET C'EST LA REGLE POUR TOUTE LA 2G. Des que
# cette topologie existe - et addition.sh l'ecrit - le banc multi-operateur
# entier montait AU BOOT : la pile radio, les conteneurs, plusieurs minutes,
# avant meme que l'operateur ait vu son bureau, et sur une configuration qu'il
# n'avait pas choisie. C'est exactement ce qu'on avait deja refuse pour
# osmo-banc le 2026-09-05 (82-services.sh l.158) ; il n'y a pas de raison que
# le multi y echappe. Un banc 2G se DEMARRE : c'est le geste de l'operateur,
```

```
# pas un effet de bord du demarrage.
#   systemctl start osmo-multi          une session (ou l'icone du bureau)
#   systemctl enable --now osmo-multi   pour qu'il reparte a chaque demarrage
# --multi (ou OSMO_ISO_MULTI=1) au build pour une image qui l'active, comme
# --banc le fait pour osmo-banc. Les deux unites restent POSEES dans l'image :
# rien a installer, seulement a lancer.
OSMO_ISO_MULTI="${OSMO_ISO_MULTI:-0}"
export OSMO_ISO_MULTI

OUTPUT_SET=0

# ── --help ──────────────────────────────────────────────────────────────
# Place AVANT le test root (plus bas dans ce module) : demander l'aide ne
# demande pas d'etre root. Les valeurs entre crochets sont les defauts reels,
# lus dans les variables ci-dessus - pas une copie qui derivera.
iso_usage() {
    # ATTENTION : les couleurs du depot (BOLD='\033[1m'...) sont faites pour
    # `echo -e`. Un heredoc passe par `cat`, qui ne traduit rien : elles
    # sortaient telles quelles, "\033[1mbuild-iso.sh\033[0m" en toutes lettres.
    # $'...' (quoting ANSI-C) produit le vrai octet d'echappement des
    # l'affectation, et le heredoc n'a plus rien a traduire.
    local B=$'\033[1m' C=$'\033[36m' D=$'\033[2m' N=$'\033[0m'
    cat <<EOF
${B}build-iso.sh${N} - fabrique une ISO bootable (ou une image SD arm64) a partir
de l'image docker osmocom-nitb. Tout passe par build.sh et start-docker.sh.

  ${B}Usage${N} : sudo ./build-iso.sh [options]
  Sans option : les quatre images amd64 - interstp, operator, lite, desktop.

${B}QUELLES IMAGES${N}
  ${C}--role=operator|interstp${N}  le noeud complet, ou le hub SS7 seul   ${D}[${ISO_ROLE:-operator}]${N}
  ${C}--lite${N}                    l'ISO allegee : sans les arbres de sources de /opt/GSM
  ${C}--desktop${N}                 l'ISO avec bureau GNOME et installateur (~2,5 Go de plus)
  ${C}--all${N}                     les quatre d'un coup (operator + lite + desktop + interstp)
  ${C}--node=N${N}                  numero du noeud dans le plan de numerotation    ${D}[${ISO_NODE:-1}]${N}
  ${C}--output=FICHIER${N}          nom du fichier de sortie
 ↪  ${D}[${OUTPUT:-osmo-operator-<version>.iso}]${N}

${B}L'IMAGE DOCKER SOURCE${N}
  ${D}Par defaut, en amd64, l'image est TIREE et non construite, puis taguee
  osmocom-nitb. Dans l'ordre : l'image Docker Hub (bastienbaranoff/norf_gsm:latest),
  puis ghcr.io/<depot>/osmocom-nitb:base-<empreinte du depot>, puis le :latest
  de GHCR. Seule l'image d'empreinte correspond a l'arbre de travail ; pour les
  autres le script previent avant de s'en servir.
  Si aucune ne repond, build.sh prend le relais (1h23). En arm64 rien n'est
  publie : la compilation reste le defaut.${N}
  ${C}--skip-build[=IMAGE]${N}      le pull devient OBLIGATOIRE : pas de repli sur
                            build.sh, et =IMAGE impose la reference au lieu de
                            la deduire de l'empreinte. N'importe quel registre :
                              --skip-build=ghcr.io/<depot>/osmocom-nitb:base-<empreinte>
                            Une image deja presente localement n'est pas retiree.
                            ${D}[${ISO_PULL_IMAGE:-bastienbaranoff/norf_gsm:latest}]${N}
  ${C}--build-docker${N}            l'inverse, et il gagne sur --skip-build d'ou qu'il
                            vienne : l'image est construite ici par build.sh.
  ${C}--no-cache${N}                tout recompiler : cache docker ignore, .deb refaits.
                            Sur un runner, c'est 1h23 - voir .github/workflows/.

${B}CE QUI VOYAGE DANS L'IMAGE${N}
  ${C}--without-debs${N}            aucun .deb embarque : ni les paquets du banc
                            (osmo-operator, pont, qemu-calypso, calypso-firmware),
                            ni ceux du build docker. Ils posent toujours la pile
                            dans le rootfs, ils n'y restent plus.
  ${C}--with-debs${N}               le defaut : les paquets du banc sont dans
                            /var/cache/osmo-debs de l'image, pour un
                            \`dpkg -i\` sans docker ni git.
```

```
${B}AU DEMARRAGE DE L'IMAGE${N}
  ${C}--banc / --no-banc${N}        activer (ou non) osmo-banc.service au boot   ${D}[${OSMO_ISO_BANC:-0}]${N}
  ${C}--multi / --no-multi${N}      activer (ou non) osmo-multi.service au boot  ${D}[${OSMO_ISO_MULTI:-0}]${N}
  ${C}--kb=LANG${N}                 disposition clavier                          ${D}[${OSMO_ISO_KB:-fr}]${N}

${B}LA SIM${N}
  ${C}--softsim / --no-softsim${N}  softSIM (SIM via SAP, Ruby) dans /opt/GSM/softsim, pySim et
                            pcscd/CCID pour la carte d un lecteur HID Omnikey    ${D}[${OSMO_ISO_SOFTSIM:-1}]${N}

${B}LE LIEN ENTRE NOEUDS (WAN)${N}
  ${C}--wan${N}                     active la table WAN
  ${C}--wan-nodes=LISTE${N}         <noeud>:<IP>:<indicatif>, separes par des espaces
                            ${D}[${ISO_WAN_NODES_DEFAULT:-}]${N}
  ${C}--wan-id=N${N}                numero de CE noeud dans la table
  ${C}--wan-ops=N${N}               nombre d'operateurs sur ce noeud
  ${C}--hub-ip=IP${N}               adresse du hub SS7

${B}LA BASE${N}
  ${C}--version=22.04|24.04${N}     suite Ubuntu du rootfs                       ${D}[${ISO_UBUNTU:-24.04}]${N}
  --arm, --arch=arm64       images SD arm64 par github.com/armbian/build
                            (armbian/build-armbian.sh) : Raspberry Pi 4 et
                            toutes les Orange Pi 5. --desktop = GNOME + le
                            bureau du banc. Pas de --all ni de --lite. Le
                            premier build emule aarch64 : une dizaine d'heures.
  ${C}--board=LISTE${N}             cartes de --arm : rpi4, opi5, all, ou des noms
                            Armbian (orangepi5-max,rpi4b...)       ${D}[${ISO_ARM_BOARD:-all}]${N}
  ${C}--arch=amd64${N}              le defaut.

  ${C}-h, --help${N}                ceci.

${B}Equivalents en variables${N} (pour la CI, ou un environnement sans ligne de commande) :
  OSMO_ISO_BUILD_DOCKER=1   comme --build-docker
  OSMO_ISO_SKIP_BUILD=0     ne pas tirer d'image : construire, comme avant 09-2026
  OSMO_ISO_WITHOUT_DEBS=1   comme --without-debs
  OSMO_ISO_PULL_IMAGE=REF   l'image de --skip-build
  OSMO_ISO_BANC=1 / OSMO_ISO_MULTI=1 / OSMO_ISO_KB=fr / OSMO_ISO_VERSION=...
  OSMO_ISO_SOFTSIM=0        comme --no-softsim
  OSMO_ISO_WORK=/chemin     repertoire de travail (defaut /var/tmp, PAS /tmp qui
                            est souvent un tmpfs : le rootfs n'y tient pas)
  OSMO_DEB_CACHE=/chemin    cache des .deb de l'hote   ${D}[/var/cache/osmo-debs]${N}
EOF
}

for arg in "$@"; do case "$arg" in
    -h|--help)      iso_usage; exit 0 ;;
    --output=*)     OUTPUT="${arg#*=}"; OUTPUT_SET=1 ;;
    --no-cache)     NO_CACHE="--no-cache" ;;
    --wan)          ISO_WAN=1 ;;
    --wan-nodes=*)  ISO_WAN=1; ISO_WAN_NODES="${arg#*=}" ;;
    --wan-id=*)     ISO_WAN=1; ISO_WAN_ID="${arg#*=}" ;;
    --wan-ops=*)    ISO_WAN=1; ISO_WAN_OPS="${arg#*=}" ;;
    --role=*)       ISO_ROLE="${arg#*=}"; ISO_ROLE_GIVEN=1 ;;
    --node=*)       ISO_NODE="${arg#*=}" ;;
    --hub-ip=*)     ISO_HUB_IP="${arg#*=}" ;;
    --kb=*)         OSMO_ISO_KB="${arg#*=}" ;;
    --version=*)    ISO_UBUNTU="${arg#*=}" ;;
    # Forme en deux mots (`--version 24.04`) : la boucle est un `for` sans
    # shift, donc le drapeau memorise que la valeur arrive au tour suivant, et
    # le motif ci-dessous la ramasse. Ces valeurs ne signifient rien d autre
    # dans la ligne de commande : aucun risque de collision.
    --version)      ISO_UBUNTU_NEXT=1 ;;
    22.04|24.04|jammy|noble)
                    if [ "${ISO_UBUNTU_NEXT:-0}" = 1 ]; then
                        ISO_UBUNTU="$arg"; ISO_UBUNTU_NEXT=0
                    fi ;;
    # ── LE BANC AU BOOT : NON, SAUF DEMANDE EXPLICITE ──────────────────
```

```
    # osmo-banc.service et osmo-multi.service sont POSES dans toutes les images
    # (82-services.sh) mais plus ACTIVES : une machine fraichement installee
    # demarre sur son bureau, pas sur une pile radio qu'on n'a pas encore
    # configuree. --banc / --multi (ou OSMO_ISO_BANC=1 / OSMO_ISO_MULTI=1)
    # rendent l'activation au demarrage, image par image.
    --banc)         OSMO_ISO_BANC=1 ;;
    --no-banc)      OSMO_ISO_BANC=0 ;;
    --multi)        OSMO_ISO_MULTI=1 ;;
    --no-multi)     OSMO_ISO_MULTI=0 ;;
    --softsim)      OSMO_ISO_SOFTSIM=1 ;;
    --no-softsim)   OSMO_ISO_SOFTSIM=0 ;;
    --lite)         ISO_LITE=1 ;;
    --desktop)      ISO_DESKTOP=1 ;;
    --all)          ISO_ALL=1 ;;
    --skip-build)   ISO_SKIP_BUILD=1; ISO_SKIP_BUILD_GIVEN=1 ;;
    --skip-build=*) ISO_SKIP_BUILD=1; ISO_SKIP_BUILD_GIVEN=1; ISO_PULL_IMAGE="${arg#*=}"; ISO_PULL_IMAGE_GIVEN=1
    ↪  ;;
    --build-docker) ISO_FORCE_BUILD=1 ;;
    --without-debs) ISO_WITH_DEBS=0 ;;
    --with-debs)    ISO_WITH_DEBS=1 ;;
    --arm|--arch=arm64) ISO_ARCH=arm64 ;;
    --board=*)      ISO_ARM_BOARD="${arg#*=}" ;;
    --arch=amd64)   ISO_ARCH=amd64 ;;
    --arch=*)       echo -e "${RED}--arch : amd64 ou arm64 (recu : ${arg#*=})${NC}" >&2; exit 2 ;;
esac; done

# Arbitrage APRES la boucle, pour que l ordre des options sur la ligne de
# commande ne decide de rien : --build-docker l emporte sur --skip-build, d ou
# qu il vienne (ligne de commande ou variable).
if [ "$ISO_FORCE_BUILD" = "1" ] && [ "$ISO_SKIP_BUILD" = "1" ]; then
    # Le message ne se justifie que si un --skip-build a VRAIMENT ete demande :
    # depuis que le pull est le defaut, l annoncer a chaque --build-docker
    # ferait parler le script d une option que personne n a ecrite.
    [ "$ISO_SKIP_BUILD_GIVEN" = "1" ] && \
        echo -e "${YELLOW}--build-docker : --skip-build ignore, l image sera construite localement${NC}"
    ISO_SKIP_BUILD=0
fi
# Sans .deb embarquees, celles du build docker ne restent pas non plus dans le
# rootfs (50-injection-image.sh les efface sauf ISO_EMBED_DEBS=1).
[ "$ISO_WITH_DEBS" = "0" ] && ISO_EMBED_DEBS=0
export ISO_FORCE_BUILD ISO_WITH_DEBS

# ── BASE UBUNTU : 22.04 (jammy) ou 24.04 (noble) ──────────────────────────
# Le nom de suite etait ecrit EN DUR a quatre endroits (debootstrap, les trois
# lignes de sources.list). Une seule variable maintenant, et un seul endroit ou
# la faire varier.
#
# Defaut 24.04 (noble) : c est la base du Dockerfile depuis le 2026-09-03, et
# l ISO DOIT partir de la meme suite que l image dont elle copie /usr/local et
# le venv /root/.env (python 3.12, glibc 2.39, bibliotheques *t64). Un rootfs
# jammy sous une image noble donnerait un venv sans interpreteur et des .so
# sans leurs symboles. La verification est faite plus bas, sur l os-release de
# l image source. 22.04 reste accepte pour une image construite sur jammy.
: "${ISO_UBUNTU:=24.04}"
case "$ISO_UBUNTU" in
    22.04|jammy) ISO_UBUNTU=22.04; ISO_SUITE=jammy ;;
    24.04|noble) ISO_UBUNTU=24.04; ISO_SUITE=noble ;;
    *) echo -e "${RED}--version : 22.04 ou 24.04 (recu : $ISO_UBUNTU)${NC}" >&2; exit 2 ;;
esac
export ISO_UBUNTU ISO_SUITE

# Ce qui decoule de l architecture : le miroir (arm64 n est PAS sur
# archive.ubuntu.com mais sur ports), le suffixe des images docker (les deux
# architectures cohabitent sur le meme hote sans s ecraser), le format de
# sortie. ISO_IMG_TAG s ajoute au nom des images : osmocom-nitb:arm64.
case "$ISO_ARCH" in
```

```
    amd64) ISO_MIRROR="$(bash "$DIR/packaging/apt-mirror.sh" "$ISO_SUITE" 2>/dev/null || echo
     ↪  http://archive.ubuntu.com/ubuntu)"; ISO_IMG_TAG="" ;;
    arm64) ISO_MIRROR="http://ports.ubuntu.com/ubuntu-ports"; ISO_IMG_TAG=":arm64" ;;
esac
# [2026-09-14] LE SUFFIXE N EXISTE QUE POUR LA COHABITATION. Il est la pour
# qu une image arm64 construite sur un hote x86 n ecrase pas l image native du
# meme nom - c est exactement la regle de build.sh, qui ne pose IMG_TAG que
# lorsque « $OSMO_ARCH != $HOST_ARCH » (l.183). Sur un hote DEJA aarch64 il n y
# a rien a faire cohabiter : build.sh y tague osmocom-nitb tout court, tandis
# qu ici on cherchait osmocom-nitb:arm64 - une image que personne n avait
# construite, et 31-image-source.sh ne trouvait plus sa source. Les deux cotes
# doivent repondre pareil a la meme question.
[ "$ISO_ARCH" = "$(dpkg --print-architecture 2>/dev/null || echo amd64)" ] && ISO_IMG_TAG=""
export ISO_ARCH ISO_MIRROR ISO_IMG_TAG
[ "$(id -u)" -ne 0 ] && { echo -e "${RED}Root requis.${NC}"; exit 1; }

# [2026-10-24] --arm PASSE PAR LE FRAMEWORK ARMBIAN (github.com/armbian/build) :
# armbian/build-armbian.sh fabrique noyau, BSP, u-boot/firmware, rootfs et image
# pour chaque carte, et y pose la pile par userpatches/customize-image.sh. Les
# modules d assemblage manuel (82-arm-natif, 92-rpi-image : debootstrap + noyau
# apt.armbian.com + .img a la main, Pi 4 seulement) ne sont plus atteints : on
# remplace ce script par build-armbian.sh, ici, avant toute etape.
if [ "$ISO_ARCH" = "arm64" ]; then
    [ "$ISO_ALL" = "1" ]  && { echo -e "${RED}--arm : pas de --all (--board=all construit toutes les cartes)${NC}"
     ↪  >&2; exit 2; }
    [ "$ISO_LITE" = "1" ] && { echo -e "${RED}--arm : pas de --lite avec le framework Armbian${NC}" >&2; exit 2; }
    case "${ISO_ROLE:-operator}" in operator) ;; *) echo -e "${RED}--arm : --role=operator seulement (recu :
     ↪  $ISO_ROLE)${NC}" >&2; exit 2 ;; esac
    [ "$ISO_SUITE" = noble ] || { echo -e "${RED}--arm : noble (24.04) seulement - les .deb du build arm64 visent
     ↪  noble${NC}" >&2; exit 2; }
    [ "$OUTPUT_SET" = "1" ] && echo -e "${YELLOW}--arm : --output ignore, les images sortent dans le
     ↪  output/images/ d armbian/build${NC}"
    _ARM=("--board=${ISO_ARM_BOARD:-all}" "--release=$ISO_SUITE" --install-deps)   # 20-hote.sh, qui posait
     ↪  qemu-user-static, n est plus atteint
    [ "$ISO_DESKTOP" = "1" ]            && _ARM+=(--desktop)
    [ "${OSMO_ISO_BANC:-0}" = "1" ]     && _ARM+=(--banc)
    [ "${OSMO_ISO_MULTI:-0}" = "1" ]    && _ARM+=(--multi)
    # l image de --skip-build=REF seulement : le defaut (norf_gsm:latest) est amd64
    [ "${ISO_PULL_IMAGE_GIVEN:-0}" = "1" ] && _ARM+=("--image=$ISO_PULL_IMAGE")
    # rien n est publie en arm64 hors GHCR base-arm64 : sans --skip-build explicite,
    # l image de la pile se construit si elle manque (comme avant)
    [ "$ISO_SKIP_BUILD_GIVEN" = "1" ] || _ARM+=(--build-image)
    [ "${OSMO_ISO_ARM_DRY:-0}" = "1" ]  && _ARM+=(--dry-run)   # voir les compile.sh sans rien lancer
    echo -e "  ${CYAN}cible : arm64, framework Armbian - armbian/build-armbian.sh ${_ARM[*]}${NC}"
    exec bash "$DIR/armbian/build-armbian.sh" "${_ARM[@]}"
fi


# Fin de module : `. fichier` rend le statut de sa DERNIERE commande, et
# build-iso.sh tourne sous set -e. Un module qui finirait par un test faux
# ("[ ... ] && { ...; }") arreterait tout, sans un mot. Toujours 0 ici.
true
```

3.189 /opt/GSM/osmo-operator/iso_modules/10-clavier.sh

4425 octets, 70 lignes → 70 lignes

```
#!/bin/bash
# iso_modules/10-clavier.sh - clavier de l image (question ou --kb)
# Source par build-iso.sh, dans l ordre des numeros : meme shell, memes
# variables, memes fonctions. Ne s execute pas seul. `return` en tete de
# module = "rien a faire ici" (c est ainsi que --arm saute une etape).

# ═══════════════════════════════════════════════════════════════════
```

```
# LES QUESTIONS : toutes ici, une seule fois, avant que quoi que ce soit ne parte
# ═══════════════════════════════════════════════════════════════════════════
# Une question posee au milieu d'une construction d'une heure attend un humain
# qui, lui, est parti. Et posee dans l'IMAGE - au premier boot - elle bloque
# chaque machine qui demarre, alors que la reponse est la meme pour toutes.
#
# Tout ce qui se demande se demande donc ICI :
#   - avant la construction, pour ne jamais interrompre une passe en cours ;
#   - une seule fois, meme quand on produit les DEUX images : la reponse part
#     dans l'environnement (export), et les passes filles en heritent ;
#   - jamais en CI : sans terminal, on prend le defaut au lieu d'attendre un
#     EOF qui, sous set -e, ferait echouer la construction.
#
# --kb=XX ou OSMO_ISO_KB=XX court-circuitent la question.
ISO_KB_DEFAULT="fr"
if [ -z "${OSMO_ISO_KB:-}" ]; then
    if [ -t 0 ]; then
        echo -e "${CYAN}${BOLD}== Clavier de l'image ==${NC}"
        echo "  1) fr   2) us   3) de   4) es   5) it"
        echo "  6) pt   7) gb   8) be   9) ch   0) autre"
        read -rp "  Choix [1] : " _kb_choice || _kb_choice=""
        case "${_kb_choice:-1}" in
            1|"") OSMO_ISO_KB="fr" ;;
            2) OSMO_ISO_KB="us" ;;  3) OSMO_ISO_KB="de" ;;
            4) OSMO_ISO_KB="es" ;;  5) OSMO_ISO_KB="it" ;;
            6) OSMO_ISO_KB="pt" ;;  7) OSMO_ISO_KB="gb" ;;
            8) OSMO_ISO_KB="be" ;;  9) OSMO_ISO_KB="ch" ;;
            0) read -rp "  Layout (fr, us, ru, ar...) : " OSMO_ISO_KB || OSMO_ISO_KB=""
               OSMO_ISO_KB="${OSMO_ISO_KB:-$ISO_KB_DEFAULT}" ;;
            *) OSMO_ISO_KB="$ISO_KB_DEFAULT" ;;
        esac
    else
        OSMO_ISO_KB="$ISO_KB_DEFAULT"
        echo -e "  ${YELLOW}Pas de terminal : clavier ${OSMO_ISO_KB} (--kb=XX pour changer)${NC}"
    fi
fi
export OSMO_ISO_KB
echo -e "  ${GREEN}✓${NC} clavier de l'image : ${CYAN}${OSMO_ISO_KB}${NC}"

# ── Sans argument : LES DEUX images ──────────────────────────────────────
# Un WAN a besoin de deux choses differentes - un hub SS7 et des noeuds - et
# rien ne dit laquelle on veut quand on ne precise rien. On produit donc les
# deux, par deux passes completes.
#
# UNE SEULE image d'operateur suffit pour les neuf noeuds : le numero se choisit
# au demarrage (`start-native.sh --node N`), qui reecrit les point codes. C'est
# la raison pour laquelle on ne fabrique pas osmo-operator-1..9.
#
# ═══════════════════════════════════════════════════════════════════════════
# CE QUI NE SE FAIT QU UNE FOIS : les paquets de l hote, le build docker
# ═══════════════════════════════════════════════════════════════════════════
# [2026-09-03] Chaque passe fille refaisait son apt sur l hote et SON build
# docker (build.sh, puis Dockerfile.run, puis Dockerfile.lite) : quatre images
# = quatre fois le meme travail. Les deux fonctions ci-dessous sont appelees
# par la passe parente (--all) une seule fois, et les filles les sautent sur
# OSMO_ISO_HOST_READY / OSMO_ISO_IMAGE_READY. Une passe lancee seule les
# appelle elle-meme, une fois.


# Fin de module : `. fichier` rend le statut de sa DERNIERE commande, et
# build-iso.sh tourne sous set -e. Un module qui finirait par un test faux
# ("[ ... ] && { ...; }") arreterait tout, sans un mot. Toujours 0 ici.
true
```

3.190 /opt/GSM/osmo-operator/iso_modules/20-hote.sh

13137 octets, 219 lignes → 219 lignes

```
#!/bin/bash
# iso_modules/20-hote.sh - paquets de l hote (iso_host_packages)
# Source par build-iso.sh, dans l ordre des numeros : meme shell, memes
# variables, memes fonctions. Ne s execute pas seul. `return` en tete de
# module = "rien a faire ici" (c est ainsi que --arm saute une etape).

# ── Paquets hote requis pour fabriquer l'ISO (squashfs, grub, xorriso...) ──
# Installes ici plutot que dans le workflow CI : `sudo ./build-iso.sh` suffit
# sur une machine Debian/Ubuntu vierge, sans etape "Install host tools" externe.
#
# [2026-09-10] LA LISTE EST CELLE D UNE UBUNTU 24.04 FRAICHE. Elle a ete
# etablie non pas d apres ce qui manquait sur LA machine de developpement -
# qui a tout depuis longtemps - mais d apres ce que la chaine APPELLE SUR L
# HOTE, paquet par paquet, en partant d une noble qui n a qu ubuntu-minimal.
# Ce qui manquait vraiment :
#   syslinux-utils   isohybrid (91-secure-boot). "isolinux" NE LE FOURNIT PAS :
#                    isolinux pose isolinux.bin, isohybrid est dans
#                    syslinux-utils. La ligne d avant listait isolinux et
#                    l ISO sortait sans table de partitions hybride - une clef
#                    USB gravee en dd ne bootait pas sur les BIOS qui l exigent.
#   rsync            unpackfs de calamares, et les copies de rootfs (87-lite).
#   wget curl        firmware TAS2781 et sources gsm-1.0.24 (52-qemu), noyau
#                    pmOS (88-lte-pmos), cles de depots (80-chroot).
#   ca-certificates gnupg  ces telechargements sont en https, et les cles de
#                    depot passent par gpg --dearmor.
#   python3-pil      tools/plymouth-render.py (86-finitions) : le theme de
#                    demarrage est GENERE, il lui faut Pillow.
#   python3-yaml     controle des configs calamares (85-installeur-bureau).
#   gcc make         toast/untoast (codec GSM 06.10) compile sur l hote pour
#                    le rootfs (52-qemu). Sans eux : ISO sans toast, en silence.
#   kpartx           images disque de l installeur et de pmOS (85, 88).
#   e2fsprogs fdisk util-linux  mke2fs / sfdisk / losetup.
#   xz-utils cpio file zstd  squashfs -comp xz, initrd, identification.
#   grub2-common     grub-install ; grub-common seul n a que grub-mkrescue.
#   fonts-ubuntu     tools/grub-theme-render.py (86, 90) : polices du theme
#                    GRUB, converties par grub-mkfont (grub-common).
#   meson ninja-build pkg-config libpulse-dev libx11-dev libxext-dev
#   libxrender-dev libxcomposite-dev libgl-dev  GLava compile sur l hote pour
#                    le rootfs (tools/glava-build.sh, 85-glava).
# Ubuntu 24.04 fournit deja : tar, gzip, dpkg-deb, coreutils, mount. Pas la
# peine de les demander. aria2 et curl viennent avec apt-fast-install.sh.
#
# shim-signed / grub-efi-amd64-signed / dosfstools : la chaine Secure Boot.
# Voir "Etape 9" plus bas - sans eux l'ISO ne demarre pas sur une machine dont
# le Secure Boot est actif, et le firmware ne dit qu'une erreur de certificat.
# apt-fast partout : le meme installeur que le Dockerfile et le chroot
# (packaging/apt-fast-install.sh), donc les memes reglages apt sur l hote.
#
# COMMUN AUX DEUX ARCHITECTURES : tout ce qui n est ni grub ni xorriso.
ISO_HOST_PKGS_COMMON="debootstrap git rsync wget curl ca-certificates gnupg \
zstd xz-utils cpio file gcc make python3 python3-pil python3-yaml \
e2fsprogs fdisk util-linux dosfstools mtools kpartx"
# amd64 : l ISO elle-meme - squashfs, grub (BIOS + EFI), xorriso, isohybrid.
ISO_HOST_PKGS="$ISO_HOST_PKGS_COMMON squashfs-tools xorriso grub-common grub2-common grub-pc-bin
 ↪  grub-efi-amd64-bin shim-signed grub-efi-amd64-signed isolinux syslinux-common syslinux-utils \
fonts-ubuntu meson ninja-build pkg-config libpulse-dev libx11-dev libxext-dev libxrender-dev libxcomposite-dev
 ↪  libgl-dev"
# --arm : ni grub ni xorriso ni squashfs. Il faut l emulation (qemu-user-static
# + binfmt-support : chroot et docker executent de l aarch64 sur l hote x86),
# debootstrap, et de quoi fabriquer l image SD sans monter quoi que ce soit :
# mke2fs -d (e2fsprogs), mkfs.vfat + mcopy (dosfstools, mtools), sfdisk.
[ "${ISO_ARCH:-amd64}" = "arm64" ] \
```

```
    && ISO_HOST_PKGS="$ISO_HOST_PKGS_COMMON qemu-user-static binfmt-support docker-buildx"

# ── Docker : une dependance, pas une option ──────────────────────────
# Toute la chaine part d une image docker (build.sh, ou --skip-build qui la
# tire du Hub). build.sh pose docker.io + docker-compose-v2 + docker-buildx,
# mais il n est PAS appele en --skip-build - et le controle d outils de
# 22-wan-table, lui, exige docker dans les deux cas. On le pose donc ici.
# UNIQUEMENT SI ABSENT : sur un hote qui a docker-ce (depot Docker), poser
# docker.io le casserait. Un docker deja la, quelle que soit sa provenance,
# est laisse tel quel.
iso_host_docker() {
    if command -v docker >/dev/null 2>&1; then
        docker info >/dev/null 2>&1 || systemctl start docker >/dev/null 2>&1 || true
        return 0
    fi
    echo -e "  ${CYAN}docker absent : installation de docker.io + compose v2 + buildx${NC}"
    apt-fast install -y --no-install-recommends docker.io docker-compose-v2 docker-buildx \
        || apt-fast install -y --no-install-recommends docker.io docker-compose-v2 \
        || apt-fast install -y --no-install-recommends docker.io \
        || { echo -e "${RED}docker non installable par apt : posez-le (get.docker.com) et relancez${NC}" >&2;
        ↪  return 1; }
    systemctl enable --now docker >/dev/null 2>&1 || true
}

# ── Les outils que la chaine appelle, et le paquet qui les porte ─────
# Le controle d avant ne nommait que six binaires et disait "Manquant: X" sans
# dire quoi installer. Ici chaque outil sait d ou il vient : un hote ou apt a
# echoue (miroir coupe, paquet retire) donne une ligne par outil et LA commande
# qui repare. Les outils optionnels (isohybrid, gcc) ne font qu avertir.
iso_host_check_tools() {
    local missing="" pkgs="" t p
    local required="debootstrap git rsync wget curl mkfs.vfat mcopy sfdisk mke2fs kpartx python3"
    local optional="isohybrid gcc make"
    [ "${ISO_ARCH:-amd64}" = "arm64" ] \
        || required="$required mksquashfs unsquashfs xorriso grub-mkrescue grub-mkimage"
    for t in $required; do
        command -v "$t" >/dev/null 2>&1 && continue
        case "$t" in
            mksquashfs|unsquashfs) p=squashfs-tools ;;
            grub-mkrescue|grub-mkimage) p=grub-common ;;
            mkfs.vfat) p=dosfstools ;;  mcopy) p=mtools ;;
            sfdisk) p=fdisk ;;          mke2fs) p=e2fsprogs ;;
            *) p="$t" ;;
        esac
        missing="$missing $t"; pkgs="$pkgs $p"
    done
    for t in $optional; do
        command -v "$t" >/dev/null 2>&1 \
            || echo -e "  ${YELLOW}!${NC} $t absent (optionnel) : $([ "$t" = isohybrid ] \
                && echo "pas de table hybride sur l ISO (clef USB gravee en dd)" \
                || echo "toast/untoast ne seront pas compiles")" >&2
    done
    if [ -n "$missing" ]; then
        echo -e "${RED}Outils hote manquants :${missing}${NC}" >&2
        echo -e "${RED}  apt-get install -y$(echo "$pkgs" | tr ' ' '\n' | sort -u | tr '\n' ' ')${NC}" >&2
        return 1
    fi
    # Pillow : plymouth-render.py est appele avec le python3 du PATH, qui peut
    # etre un venv (/root/.env) ou python3-pil n est pas. On avertit, on ne
    # bloque pas : le theme de demarrage n empeche pas l ISO de sortir.
    python3 -c "import PIL" >/dev/null 2>&1 \
        || echo -e "  ${YELLOW}!${NC} Pillow absent du python3 du PATH ($(command -v python3)) - theme Plymouth
        ↪  non genere" >&2
    return 0
}
```

```
# Appelable plusieurs fois (21-docker-build en --all, puis 22-wan-table) : apt
# ne tourne qu au premier appel. Sans ce garde-fou, une passe --all refaisait
# l `apt-fast install` a chaque module qui appelle la fonction.
_ISO_HOST_PKGS_DONE=0
iso_host_packages() {
    [ "$_ISO_HOST_PKGS_DONE" = "1" ] && return 0
    _ISO_HOST_PKGS_DONE=1
    if [ "${OSMO_ISO_HOST_READY:-0}" = "1" ]; then
        echo -e "${GREEN}[0/9] Paquets hote : deja poses par la passe parente${NC}"; return 0
    fi
    if command -v apt-get &>/dev/null; then
        echo -e "${GREEN}[0/9] Installation des paquets hote (apt-fast)...${NC}"
        export DEBIAN_FRONTEND=noninteractive
        bash "$DIR/packaging/apt-fast-install.sh" >/dev/null 2>&1 \
            || echo -e "  ${YELLOW}apt-fast non installe - apt-get${NC}"
        command -v apt-fast >/dev/null 2>&1 || apt-fast() { apt-get "$@"; }
        apt-fast update -qq || true
        # UN paquet introuvable (retire d une suite, miroir partiel) faisait
        # echouer la ligne entiere, et le repli d avant reposait une liste
        # ecrite a la main, plus courte que la vraie. Repli generique : on
        # reprend paquet par paquet, et on NOMME ceux qui n ont pas voulu.
        if ! apt-fast install -y --no-install-recommends $ISO_HOST_PKGS; then
            echo -e "  ${YELLOW}installation groupee refusee - reprise paquet par paquet${NC}"
            local _ko="" _p
            for _p in $ISO_HOST_PKGS; do
                apt-fast install -y --no-install-recommends "$_p" >/dev/null 2>&1 || _ko="$_ko $_p"
            done
            [ -n "$_ko" ] && echo -e "  ${YELLOW}non installes :${_ko}${NC}" >&2
        fi
        iso_host_docker || true
    else
        echo -e "${YELLOW}apt-get absent : verification seule des outils hote.${NC}"
    fi
    iso_host_check_tools
}

# ── binfmt : l hote x86 doit executer l aarch64 de facon TRANSPARENTE ────────
# Deux consommateurs. Le chroot du rootfs : il suffit que qemu-aarch64-static
# soit enregistre dans binfmt_misc et copie DANS le rootfs. Docker (buildx
# --platform linux/arm64, docker run de l image arm64 pour la cloture ldd) :
# le conteneur n a pas l emulateur, il faut le drapeau F (fix-binary) pour que
# le noyau garde un descripteur ouvert sur l emulateur de l hote. Le paquet
# qemu-user-static l enregistre avec F depuis jammy ; si ce n est pas le cas
# ici, on passe par l image binfmt de Docker (tonistiigi/binfmt), qui fait
# exactement cet enregistrement - c est un telechargement, il est dit.
#
# [2026-09-14] SUR UN HOTE arm64, IL N Y A RIEN A EMULER. Cette fonction
# s arretait en erreur fatale sur une machine aarch64 (un Pi, un Ampere, le
# runner ubuntu-24.04-arm de GitHub) : /proc/sys/fs/binfmt_misc/qemu-aarch64
# n y existe pas - et n a aucune raison d exister -, update-binfmts n a rien a
# activer, et tonistiigi/binfmt --install arm64 n installe pas un handler pour
# l architecture native. On sortait donc en « binfmt qemu-aarch64
# indisponible » sur la seule machine qui n en a pas besoin. Meme regle que
# build.sh (l.183) et 40-rootfs.sh (l.57) : on compare a l architecture de
# l hote, et on ne parle d emulation que si les deux different.
iso_arm_binfmt() {
    [ "${ISO_ARCH:-amd64}" = "arm64" ] || return 0
    if [ "${ISO_ARCH}" = "$(dpkg --print-architecture 2>/dev/null || echo amd64)" ]; then
        echo -e "  ${GREEN}✓${NC} hote deja en ${ISO_ARCH} : ni qemu-user ni binfmt, tout est natif"
        return 0
    fi
    local f=/proc/sys/fs/binfmt_misc/qemu-aarch64
    [ -d /proc/sys/fs/binfmt_misc ] || mount -t binfmt_misc binfmt_misc /proc/sys/fs/binfmt_misc 2>/dev/null ||
     ↪  true
    [ -e "$f" ] || update-binfmts --enable qemu-aarch64 >/dev/null 2>&1 || true
    if [ -e "$f" ] && grep -q '^flags:.*F' "$f"; then
```

```
        echo -e "  ${GREEN}✓${NC} binfmt qemu-aarch64 (fix-binary) : chroot et docker savent executer l aarch64"
        return 0
    fi
    echo -e "  ${YELLOW}binfmt qemu-aarch64 sans drapeau F - enregistrement via tonistiigi/binfmt (docker)${NC}"
    docker run --privileged --rm tonistiigi/binfmt --install arm64 >/dev/null 2>&1 || true
    if [ -e "$f" ] && grep -q '^flags:.*F' "$f"; then
        echo -e "  ${GREEN}✓${NC} binfmt qemu-aarch64 enregistre (fix-binary)"
    else
        echo -e "${RED}binfmt qemu-aarch64 indisponible : apt install qemu-user-static binfmt-support, puis
        ↪  relancez${NC}" >&2
        exit 1
    fi
}


# ── ET ON LES POSE ICI, AU LANCEMENT ──────────────────────────────────────
# [2026-09-10] La fonction etait DEFINIE dans ce module mais appelee seulement
# par 21-docker-build (branche --all) et 22-wan-table. Consequence sur une
# machine fraiche : le premier module a avoir besoin d un outil le trouvait
# absent, et l erreur tombait plusieurs etapes plus loin - loin de sa cause.
# Les paquets sont maintenant poses des le module 20, avant le moindre docker
# build ; le seul module qui passe avant est 10-clavier (la question du
# clavier, qu on veut toujours en premier). Les appels de 21 et 22 restent :
# ils ne refont rien (garde _ISO_HOST_PKGS_DONE).
iso_host_packages


# Fin de module : `. fichier` rend le statut de sa DERNIERE commande, et
# build-iso.sh tourne sous set -e. Un module qui finirait par un test faux
# ("[ ... ] && { ...; }") arreterait tout, sans un mot. Toujours 0 ici.
true
```

3.191 /opt/GSM/osmo-operator/iso_modules/20-stockage-docker.sh

2756 octets, 48 lignes → 48 lignes

```
#!/bin/bash
# iso_modules/20-stockage-docker.sh - Docker ne doit pas stocker sur un overlay
# Source par build-iso.sh, dans l ordre des numeros (apres 20-hote : docker est
# deja pose). Ne s execute pas seul.
#
# Quand l hote est lui-meme un conteneur (racine en `overlay`), le snapshotter
# overlayfs de docker/containerd monte un overlay SUR un overlay, ce que le
# noyau refuse. Le build echoue des les `RUN --mount=type=cache` avec :
#   failed to solve: mount source: "overlay" ... err: invalid argument
# Remede : /var/lib/docker et /var/lib/containerd sur un ext4 en boucle
# (fichier sparse). Idempotent ; ne touche a rien si le stockage est deja sain.
# Desactiver : OSMO_ISO_NO_DOCKER_FIX=1. Taille : OSMO_ISO_DOCKER_LOOP_GB (10).
iso_docker_storage_fix() {
    command -v docker >/dev/null 2>&1 || return 0
    [ "${OSMO_ISO_NO_DOCKER_FIX:-0}" = 1 ] && return 0
    # --skip-build ne construit rien : le pull n a pas besoin d overlay imbrique.
    local fs; fs="$(findmnt -no FSTYPE -T /var/lib/containerd 2>/dev/null || stat -f -c %T /var/lib/containerd
    ↪  2>/dev/null)"
    [ "$fs" = overlay ] || return 0
    [ "$(id -u)" = 0 ] || { echo -e "${YELLOW}! /var/lib/containerd est sur overlay : le build docker echouera
    ↪  (relancer en root)${NC}" >&2; return 0; }

    local img=/var/lib/docker-loop.img base=/var/lib/dockerdata gb="${OSMO_ISO_DOCKER_LOOP_GB:-10}"
    echo -e "  ${CYAN}stockage docker sur overlay : bascule sur un ext4 en boucle (${gb} Go, $img)${NC}"
    if [ -n "$(docker ps -q 2>/dev/null)" ]; then
        echo -e "${RED}des conteneurs tournent : arretez-les avant (le correctif redemarre docker)${NC}" >&2
        return 1
    fi
```

```
    systemctl stop docker docker.socket containerd || return 1
    mkdir -p "$base"
    if [ ! -f "$img" ]; then
        truncate -s "${gb}G" "$img" && mkfs.ext4 -q -F "$img" || return 1
    fi
    mountpoint -q "$base" || mount -o loop "$img" "$base" || return 1
    mkdir -p "$base/docker" "$base/containerd"
    mountpoint -q /var/lib/docker     || mount --bind "$base/docker" /var/lib/docker || return 1
    mountpoint -q /var/lib/containerd || mount --bind "$base/containerd" /var/lib/containerd || return 1
    if ! grep -q docker-loop /etc/fstab 2>/dev/null; then
        printf '%s\n' "$img $base ext4 loop 0 0" \
            "$base/docker /var/lib/docker none bind 0 0" \
            "$base/containerd /var/lib/containerd none bind 0 0" >> /etc/fstab
    fi
    systemctl start containerd docker || return 1
    sleep 3
    echo -e "  ${GREEN}✓${NC} docker stocke sur $(findmnt -no FSTYPE -T /var/lib/containerd)"
}
iso_docker_storage_fix || { echo -e "${RED}Correctif du stockage docker echoue (voir plus haut)${NC}" >&2; exit 1;
 ↪  }

# Fin de module : toujours 0 (voir 20-hote.sh).
true
```

3.192 /opt/GSM/osmo-operator/iso_modules/21-docker-build.sh

18216 octets, 292 lignes → 292 lignes

```
#!/bin/bash
# iso_modules/21-docker-build.sh - iso_docker_build, passe parente --all, nom de sortie
# Source par build-iso.sh, dans l ordre des numeros : meme shell, memes
# variables, memes fonctions. Ne s execute pas seul. `return` en tete de
# module = "rien a faire ici" (c est ainsi que --arm saute une etape).

# ── UN SEUL build docker ──────────────────────────────────────────────
# L image source de TOUTES les ISO est osmocom-nitb (Dockerfile), construite
# par build.sh - qui la passe par docker compose et par le cache .deb. Plus de
# Dockerfile.run ni de Dockerfile.lite ici : les configs sont injectees depuis
# ce depot (etape 2b), l elagage lite se fait sur le rootfs (etape 8c).
# Seule exception : --role=interstp demande SEUL. Le hub n a besoin que
# d osmo-stp et de trois bibliotheques : Dockerfile.stp, et on ne va pas au
# bout de la pile.
# ── LES IMAGES CANDIDATES, DANS L ORDRE ──────────────────────────────
# [2026-09-14] L EMPREINTE EST CELLE DE LA CI : la meme commande que
# .github/workflows/docker.yml (job `base`) et build-iso.yml (« Empreinte du
# contexte »), aux redirections pres, qui ne changent pas le calcul. C est tout l interet - si elle donne la meme
# valeur ici que sur le runner, l image publiee sous ce nom a ete construite a
# partir EXACTEMENT de ce qui est dans l arbre de travail, et la recompiler ne
# produirait rien d autre. Toucher a cette liste de chemins sans la toucher
# dans les deux workflows fait diverger les empreintes, et plus rien ne se
# tire : les trois doivent bouger ensemble.
#
# L ordre (depuis le 2026-10-10, Docker Hub passe devant GHCR) :
#   1. la reference EXPLICITE de --skip-build=REF - elle ne se discute pas ;
#   1b. l image Docker Hub ($ISO_PULL_IMAGE, bastienbaranoff/norf_gsm:latest),
#       taguee osmocom-nitb comme les autres ;
#   2. ghcr.io/<depot>/osmocom-nitb:base-<empreinte> - celle qui correspond
#       au commit, publiee par docker.yml ;
#   3. ghcr.io/<depot>/osmocom-nitb:latest - la derniere publiee par le meme
#       workflow, sur le meme registre. Elle ne suit pas le commit, mais elle
#       est a jour et elle est PUBLIQUE : verifie le 14-09, le registre repond
#       sans jeton et porte base-89a36ab9ba1d941d, base-e68ecc7ee98f41e0,
#       cache et latest ;
# Seule la 2 garantit que l image correspond a l arbre de travail : pour les
```

```
# deux suivantes, iso_docker_build previent avant de s en servir.
iso_pull_refs() {
    if [ "${ISO_SKIP_BUILD_GIVEN:-0}" = "1" ]; then echo "$ISO_PULL_IMAGE"; return 0; fi
    local _key _repo
    # EN CONDITION DE `if`, et ce n est pas du style : build-iso.sh tourne sous
    # `set -euo pipefail`, et une affectation depuis une pipeline qui echoue
    # (hors depot git, git ls-files sort en 128) arreterait tout le script sans
    # un mot - exactement le piege qui a coute le build du 13-09 dans
    # 86-finitions.sh. En condition, set -e laisse passer et _key reste vide.
    # Le 2>/dev/null est SUR git, pas sur le `if` : dans une commande simple les
    # expansions precedent les redirections, une redirection posee au bout ne
    # couvre donc pas la substitution - « fatal: not a git repository » sortait
    # quand meme, au milieu du build.
    if ! _key="$(cd "$DIR" && git ls-files -- Dockerfile packaging configs scripts services \
                              helpers opt-gsm tools patches 2>/dev/null \
                 | sort | xargs sha256sum 2>/dev/null | sha256sum | cut -c1-16)"; then
        _key=""
    fi
    # Le depot vient de origin ; hors clone git (une archive deployee a la
    # main), on retombe sur le depot amont plutot que de ne rien proposer.
    _repo="$(cd "$DIR" && git config --get remote.origin.url 2>/dev/null || true)"
    _repo="${_repo%.git}"; _repo="${_repo##*github.com[:/]}"
    # Un `owner/nom` et rien d autre : une origin qui n est pas sur github.com
    # ressort telle quelle du decoupage ci-dessus (ssh, gitlab, un chemin
    # local), et on n en ferait qu une reference GHCR imaginaire.
    case "$_repo" in *[:@\ ]*|*/*/*) _repo="" ;; */*) ;; *) _repo="" ;; esac
    [ -n "$_repo" ] || _repo="bbaranoff/osmo-operator"
    # L empreinte est vide hors depot git : pas de reference d empreinte a
    # proposer, mais le latest du meme depot reste valable.
    # [2026-10-10] Docker Hub EN TETE : bastienbaranoff/norf_gsm:latest est la
    # pile de reference du banc ; GHCR ne sert plus que de repli.
    echo "$ISO_PULL_IMAGE"
    [ -n "$_key" ] && echo "ghcr.io/${_repo,,}/osmocom-nitb:base-${_key}"
    echo "ghcr.io/${_repo,,}/osmocom-nitb:latest"
}

iso_docker_build() {
    local role="$1"
    if [ "${OSMO_ISO_IMAGE_READY:-0}" = "1" ]; then
        echo -e "${GREEN}[1/9] Image docker : deja construite par la passe parente${NC}"; return 0
    fi
    # ── TIRER PLUTOT QUE CONSTRUIRE ──────────────────────────────────────
    # L image tiree est taguee du nom que build.sh aurait produit (osmocom-nitb,
    # ou osmocom-nitb:arm64 en --arm) pour que 31-image-source et toute la
    # suite n y voient aucune difference ; OSMO_ISO_SRC_IMAGE empeche en plus
    # une image locale du meme nom de s inviter.
    #
    # DEUX GARDES avant meme d essayer, et elles ne valent que pour le DEFAUT :
    # un --skip-build ecrit a la main passe outre les deux.
    #   arm64    rien n est publie pour cette architecture ; un pull amd64 sur
    #            une cible arm64 donnerait un rootfs inutilisable.
    #   interstp le hub seul se construit en MINUTES depuis Dockerfile.stp
    #            (osmo-stp et trois bibliotheques). Lui faire tirer les ~11 Go
    #            de l image de base pour en extraire quatre binaires serait
    #            plus long que de la compiler.
    local _try_pull=0
    if [ "${ISO_SKIP_BUILD:-0}" = "1" ] && [ "${ISO_FORCE_BUILD:-0}" != "1" ]; then
        if [ "${ISO_SKIP_BUILD_GIVEN:-0}" = "1" ]; then _try_pull=1
        elif [ "${ISO_ARCH:-amd64}" = "amd64" ] && [ "$role" != "interstp" ]; then _try_pull=1
        fi
    fi
    if [ "$_try_pull" = "1" ]; then
        local _local="osmocom-nitb${ISO_IMG_TAG}" _ref _got=""
        echo -e "${GREEN}[1/9] Image docker : on TIRE avant de construire${NC}"
        for _ref in $(iso_pull_refs); do
            # DEJA LA ? On ne tire pas. Le registre n est pas toujours joignable
```

```
        # par celui qui execute : sur un runner, docker/login-action ecrit
        # les identifiants dans le ~/.docker du compte runner, et
        # build-iso.sh tourne sous sudo - un pull GHCR y repartirait sans
        # jeton. L image ayant ete tiree en amont, il n y a rien a chercher.
        if docker image inspect "$_ref" >/dev/null 2>&1; then
            echo -e "  ${GREEN}✓${NC} ${CYAN}${_ref}${NC} deja presente localement"
            _got="$_ref"; break
        fi
        # manifest inspect AVANT le pull : il coute un aller-retour HTTP la
        # ou un pull rate peut passer des minutes a recommencer des couches.
        if docker manifest inspect "$_ref" >/dev/null 2>&1; then
            echo -e "  ${CYAN}pull${NC} ${_ref} (linux/${ISO_ARCH:-amd64})..."
            if docker pull --platform "linux/${ISO_ARCH:-amd64}" "$_ref"; then
                _got="$_ref"; break
            fi
        fi
        echo -e "  ${YELLOW}!${NC} ${_ref} : indisponible"
    done
    if [ -n "$_got" ]; then
        # TOUT CE QUI N EST PAS UNE IMAGE D EMPREINTE ne suit pas le
        # commit : on ne le laisse pas passer en silence, ni pour le latest
        # de GHCR ni pour Docker Hub. Une ISO batie dessus peut porter une
        # pile differente de l arbre de travail - ce qui s explique cinq
        # minutes apres le build, et jamais trois semaines apres. Le motif
        # `:base-` est la signature des seules images qui, elles, collent.
        # `if`, et pas « [ ... ] && [ ... ] && echo » : une chaine && dont
        # le premier test est faux rend 1, et ce depot s est deja fait
        # arreter par une ligne de cette forme sous set -e.
        # [2026-10-10] Sauf l image Docker Hub : elle passe en tete PAR
        # CHOIX, l avertir a chaque build ne dirait plus rien.
        if [ "${_got}" = "${_got#*:base-}" ] && [ "${_got}" != "${ISO_PULL_IMAGE}" ] \
           && [ "${ISO_SKIP_BUILD_GIVEN:-0}" != "1" ]; then
            echo -e "  ${YELLOW}!${NC} aucune image ne correspond a l'empreinte du depot : ${CYAN}${_got}${NC}
             ↪  ne suit pas ce commit (${CYAN}--build-docker${NC} pour compiler l'arbre de travail)"
        fi
        docker tag "$_got" "$_local"
        export OSMO_ISO_SRC_IMAGE="$_local"
       echo -e "  ${GREEN}✓${NC} image ${_local} prete ($(docker image inspect "$_local" --format '{{.Size}}'
         ↪  2>/dev/null | awk '{printf "%.0f Mo", $1/1048576}'))"
        return 0
    fi
    # Demande a la main, le pull etait la consigne : ne pas y arriver est
    # une erreur. Par defaut, ce n etait qu un raccourci - on compile.
    if [ "${ISO_SKIP_BUILD_GIVEN:-0}" = "1" ]; then
        echo -e "${RED}--skip-build : aucune image joignable (${ISO_PULL_IMAGE})${NC}" >&2; exit 1
    fi
    echo -e "  ${YELLOW}!${NC} aucune image publiee joignable - construction locale"
fi
if [ "$role" = "interstp" ]; then
   echo -e "${GREEN}[1/9] Hub seul : construction de ${CYAN}osmocom-stp${NC}${GREEN} (Dockerfile.stp)...${NC}"
    echo -e "  ${CYAN}osmo-stp + libosmocore + libosmo-netif + libosmo-sigtran. Rien d'autre.${NC}"
    if [ "${ISO_ARCH:-amd64}" = "arm64" ]; then
        # arm64 : buildx --platform, image taguee :arm64, cache .deb de l hote
        # dans le contexte comme le fait build.sh (osmo-deb y lit les paquets
        # de SON architecture, les amd64 qui y trainent ne le derangent pas).
        mkdir -p "$DIR/.deb-cache"; cp -f "${OSMO_DEB_CACHE:-/var/cache/osmo-debs}"/osmo-build-*_arm64.deb
         ↪  "$DIR/.deb-cache/" 2>/dev/null || true
        docker buildx build --platform linux/arm64 --load $NO_CACHE \
            --build-arg "OSMO_DEB_REFRESH=$([ -n "$NO_CACHE" ] && echo 1 || echo 0)" \
            -f "$DIR/Dockerfile.stp" -t "osmocom-stp${ISO_IMG_TAG}" "$DIR" \
            || { echo -e "${RED}Echec de la construction d'osmocom-stp${ISO_IMG_TAG}${NC}"; exit 1; }
    elif docker compose version >/dev/null 2>&1; then
        ( cd "$DIR" && OSMO_DEB_REFRESH="$([ -n "$NO_CACHE" ] && echo 1 || echo 0)" \
          docker compose -f "$DIR/compose.yaml" build $NO_CACHE stp ) \
            || { echo -e "${RED}Echec de la construction d'osmocom-stp${NC}"; exit 1; }
    else
```

```
            docker build $NO_CACHE -f "$DIR/Dockerfile.stp" -t osmocom-stp "$DIR" \
                || { echo -e "${RED}Echec de la construction d'osmocom-stp${NC}"; exit 1; }
        fi
        echo -e "  ${GREEN}✓${NC} osmocom-stp${ISO_IMG_TAG} construite ($(docker image inspect
        ↪  "osmocom-stp${ISO_IMG_TAG}" --format '{{.Size}}' 2>/dev/null | awk '{printf "%.0f Mo", $1/1048576}'))"
        return 0
    fi
    echo -e "${GREEN}[1/9] Execution de build.sh (image osmocom-nitb${ISO_IMG_TAG}, docker compose + cache
    ↪  .deb)...${NC}"
    if [ -f "$DIR/build.sh" ]; then
        # --arch=arm64 : build.sh passe par buildx --platform linux/arm64 et tague
        # l image osmocom-nitb:arm64. Le reste (apt-fast, docker, sctp, toast, le
        # cache .deb) est identique.
        bash "$DIR/build.sh" $NO_CACHE $([ "${ISO_ARCH:-amd64}" = "arm64" ] && echo "--arch=arm64")
    else
        echo -e "${YELLOW}build.sh introuvable, construction manuelle de l'image osmocom-nitb...${NC}"
        if [ "${ISO_ARCH:-amd64}" = "arm64" ]; then
            # --target : Dockerfile finit par l etape `debs` (cache .deb pour
            # la CI), pas par osmocom-nitb. Sans lui, image vide.
            docker buildx build --platform linux/arm64 --load $NO_CACHE \
                --target osmocom-nitb -t "osmocom-nitb${ISO_IMG_TAG}" "$DIR"
        else
            docker build $NO_CACHE --target osmocom-nitb -t osmocom-nitb "$DIR"
        fi
    fi
    echo -e "  ${GREEN}✓${NC} image osmocom-nitb${ISO_IMG_TAG} prete"
}

if [ "$ISO_ALL" = "1" ] || { [ "${ISO_ARCH:-amd64}" = "amd64" ] && [ "$ISO_ROLE_GIVEN" = "0" ] && [ "$OUTPUT_SET"
↪  = "0" ] \
   && [ -z "$ISO_NODE" ] && [ "$ISO_LITE" = "0" ] && [ "$ISO_DESKTOP" = "0" ]; }; then
    _N="QUATRE"   # [2026-08-29] --all par defaut : les QUATRE images, desktop incluse
    echo -e "${CYAN}${BOLD}== Construction des ${_N} images ==${NC}"
    echo -e "  1. ${CYAN}interstp.iso${NC}               le hub SS7 (PC 0.0.0)"
    echo -e "  2. ${CYAN}osmo-operator.iso${NC}          un noeud - son numero se choisit au demarrage :"
    echo -e "     ${CYAN}./start-native.sh --node N${NC}   (N de 1 a 9)"
    echo -e "  3. ${CYAN}osmo-operator-lite.iso${NC}     le meme noeud, sans les ateliers de compilation"
    echo -e "  4. ${CYAN}osmo-operator-desktop.iso${NC}  le meme noeud, avec GNOME, wireshark et linphone"
    echo ""

    # --all et --desktop RETIRES des arguments repasses aux passes filles.
    # Les laisser ferait rentrer chaque passe dans ce meme bloc : une recursion
    # sans fond, qui ne produirait jamais la moindre ISO.
    SUB_ARGS=()
    for _a in "$@"; do
        case "$_a" in --all|--desktop) ;; *) SUB_ARGS+=("$_a") ;; esac
    done
    set -- "${SUB_ARGS[@]+"${SUB_ARGS[@]}"}"

    # ── [2026-09-03] UNE SEULE FOIS : apt sur l hote, build docker ──────────
    iso_host_packages
    iso_docker_build operator
    export OSMO_ISO_HOST_READY=1 OSMO_ISO_IMAGE_READY=1 OSMO_ISO_ALL_RUN=1

    # ── L ORDRE : interstp, normal, lite, desktop - et le rootfs se transmet ──
    # Le hub d abord : petit, independant, un echec se voit en minutes. Puis la
    # NORMALE, construite de zero (debootstrap + apt + injection) : c est elle
    # qui coute. Les deux autres REPRENNENT SON ROOTFS au lieu de le refaire :
    #   lite     = une COPIE de la normale, dont on RETIRE les ateliers (elle
    #              enleve, elle vient donc apres la normale, jamais avant) ;
    #   desktop  = la normale elle-meme (deplacee), a laquelle on AJOUTE la
    #              difference apt : le bureau, l installeur, les snaps.
    # apt est idempotent : sur un rootfs herite, la liste commune coute quelques
    # secondes de resolution, seul le delta est telecharge.
    _KEEP="$WORK"; mkdir -p "$_KEEP"
    _all_fail() { echo -e "${RED}Echec de $1${NC}" >&2; rm -rf "$_KEEP"; exit 1; }
```

```
    "$0" --role=interstp "$@" || _all_fail interstp.iso
    OSMO_ISO_ROOTFS_KEEP="$_KEEP/rootfs-normal" \
        "$0" --role=operator --output=osmo-operator.iso "$@" || _all_fail osmo-operator.iso
    [ -d "$_KEEP/rootfs-normal" ] || _all_fail "osmo-operator.iso (rootfs non transmis)"
    OSMO_ISO_ROOTFS_FROM="$_KEEP/rootfs-normal" OSMO_ISO_ROOTFS_MODE=copy \
        "$0" --role=operator --lite --output=osmo-operator-lite.iso "$@" || _all_fail osmo-operator-lite.iso
    _ISOS=("$(pwd)/interstp.iso" "$(pwd)/osmo-operator.iso" "$(pwd)/osmo-operator-lite.iso")
    OSMO_ISO_ROOTFS_FROM="$_KEEP/rootfs-normal" OSMO_ISO_ROOTFS_MODE=move \
        "$0" --role=operator --desktop --output=osmo-operator-desktop.iso "$@" || _all_fail
        ↪ osmo-operator-desktop.iso
    _ISOS+=("$(pwd)/osmo-operator-desktop.iso")
    rm -rf "$_KEEP"
    echo -e "${GREEN}${BOLD}═══ Les ${_N} images sont pretes ═══${NC}"
    ls -lh "${_ISOS[@]}" 2>/dev/null | sed 's/^/  /'
    exit 0
fi

case "${ISO_ROLE:-operator}" in
    interstp)
        ISO_ROLE="interstp"
        [ "$OUTPUT_SET" = "1" ] || OUTPUT="interstp.iso"
        # Le hub n'a pas d'atelier a elaguer : son image (Dockerfile.stp) ne
        # porte que osmo-stp et quatre bibliotheques. --lite n'y veut rien dire,
        # et l'accepter en silence produirait une "lite" identique a l'autre.
        [ "$ISO_LITE" = "1" ] && { echo "--lite ne s'applique pas au hub (--role=interstp)" >&2; exit 2; }
        # Meme raison pour le bureau : le hub tourne sans ecran, en salle
        # machine ou en VM sans console graphique. Un GNOME dessus, c'est 2,5 Go
        # et une pile X de plus sur la seule image qui n'affiche jamais rien.
        [ "$ISO_DESKTOP" = "1" ] && { echo "--desktop ne s'applique pas au hub (--role=interstp)" >&2; exit 2; }
        # Le hub dessert N noeuds : sans table WAN on ne sait pas combien.
        ISO_WAN=1 ;;
    operator|"")
        ISO_ROLE="operator"
        if [ -n "$ISO_NODE" ]; then
            [[ "$ISO_NODE" =~ ^[1-9]$ ]] || { echo "--node : 1 a 9" >&2; exit 2; }
            if [ "$OUTPUT_SET" = "0" ]; then
                _sfx=""
                [ "$ISO_LITE" = "1" ]    && _sfx="${_sfx}-lite"
                [ "$ISO_DESKTOP" = "1" ] && _sfx="${_sfx}-desktop"
                OUTPUT="osmo-operator-${ISO_NODE}${_sfx}.iso"
            fi
            ISO_WAN_ID="${ISO_WAN_ID:-$ISO_NODE}"
            ISO_WAN=1
        elif [ "$ISO_LITE" = "1" ] || [ "$ISO_DESKTOP" = "1" ]; then
            if [ "$OUTPUT_SET" = "0" ]; then
                _sfx=""
                [ "$ISO_LITE" = "1" ]    && _sfx="${_sfx}-lite"
                [ "$ISO_DESKTOP" = "1" ] && _sfx="${_sfx}-desktop"
                OUTPUT="osmo-operator${_sfx}.iso"
            fi
        fi ;;
    *) echo "--role inconnu : $ISO_ROLE (operator|interstp)" >&2; exit 2 ;;
esac
# --arm : une image SD, pas une ISO. Le nom suit la meme regle (role, noeud,
# -lite), avec -rpi4.img a la place de .iso - sauf --output=, respecte tel quel.
if [ "${ISO_ARCH:-amd64}" = "arm64" ] && [ "$OUTPUT_SET" != "1" ]; then
    OUTPUT="${OUTPUT%.iso}-rpi4.img"
fi
case "$OUTPUT" in /*) ;; *) OUTPUT="$(pwd)/$OUTPUT" ;; esac


# Fin de module : `. fichier` rend le statut de sa DERNIERE commande, et
# build-iso.sh tourne sous set -e. Un module qui finirait par un test faux
# ("[ ... ] && { ...; }") arreterait tout, sans un mot. Toujours 0 ici.
true
```

3.193 /opt/GSM/osmo-operator/iso_modules/22-wan-table.sh

3784 octets, 73 lignes → 73 lignes

```
#!/bin/bash
# iso_modules/22-wan-table.sh - table WAN, cleanup/trap, outils requis, WORK
# Source par build-iso.sh, dans l ordre des numeros : meme shell, memes
# variables, memes fonctions. Ne s execute pas seul. `return` en tete de
# module = "rien a faire ici" (c est ainsi que --arm saute une etape).

# ── La table WAN : arretee ICI, pas au milieu de la construction ──────────
# --role=interstp implique le WAN (le hub doit savoir combien de noeuds il
# dessert). L'etape 7b la demandait alors interactivement, une heure apres le
# lancement : de quoi bloquer une construction que l'on croyait autonome, et
# faire echouer la CI, qui n'a pas de terminal pour repondre.
#
# On la fige donc maintenant, avec la table du banc pour defaut. --wan-nodes
# reste prioritaire et n'est pas touche.
# [2026-09-04] AUTOMATIQUE. La question etait posee a chaque passe fille (quatre
# fois pour --all), au milieu d une construction d une heure, pour une reponse
# qui est presque toujours le defaut. La table du banc et le hub s appliquent
# donc sans rien demander ; pour autre chose, --wan-nodes="..." et --hub-ip=...
# (ou OSMO_ISO_WAN_ASK=1 pour retrouver la question, avec un terminal).
if [ "$ISO_WAN" = "1" ] && [ -z "$ISO_WAN_NODES" ]; then
    if [ "${OSMO_ISO_WAN_ASK:-0}" = "1" ] && [ -t 0 ]; then
        echo -e "${CYAN}${BOLD}== Table WAN de l'image ==${NC}"
        echo -e "  Format : ${CYAN}<noeud>:<IP>:<indicatif>${NC}, separes par des espaces."
        echo -e "  Entree vide = le banc : ${CYAN}${ISO_WAN_NODES_DEFAULT}${NC}"
        read -rp "  Noeuds : " _wan_in || _wan_in=""
        ISO_WAN_NODES="${_wan_in:-$ISO_WAN_NODES_DEFAULT}"
        if [ "$ISO_ROLE" = "interstp" ]; then
            read -rp "  IP du hub [${ISO_HUB_IP}] : " _hub_in || _hub_in=""
            ISO_HUB_IP="${_hub_in:-$ISO_HUB_IP}"
        fi
    else
        ISO_WAN_NODES="$ISO_WAN_NODES_DEFAULT"
    fi
    echo -e "  ${GREEN}✓${NC} WAN : ${CYAN}${ISO_WAN_NODES}${NC}   hub ${CYAN}${ISO_HUB_IP}${NC}  (defaut du banc
    ↪  ; --wan-nodes= / --hub-ip= pour changer)"
fi

# Propage --no-cache aux deux builds Docker : build.sh (image osmocom-nitb) et
# build_run_image (image osmocom-run, via DOCKER_NO_CACHE).
export DOCKER_NO_CACHE="$NO_CACHE"


cleanup() { umount "$ROOTFS/var/cache/apt/archives" 2>/dev/null||true; umount "$ROOTFS"/{dev/pts,proc,sys,dev}
 ↪  2>/dev/null||true; rm -rf "$WORK"; }
trap cleanup EXIT


# Paquets hote : une fois (voir iso_host_packages, plus haut).
iso_host_packages

# Docker n'est pas auto-installe ici (paquet docker-ce hors apt standard).
_ISO_TOOLS="docker mksquashfs xorriso grub-mkrescue debootstrap git"
# [2026-09-14] qemu-aarch64-static n est exige que pour EMULER : sur un hote
# deja aarch64 (un runner ubuntu-24.04-arm, un Pi, un Ampere) le paquet n a
# aucune raison d etre la, et le job mourait sur « Manquant: qemu-aarch64-static »
# avant la premiere etape. Meme regle que 40-rootfs.sh (l.57), qui decide de
# --foreign en comparant a l architecture de l hote.
if [ "${ISO_ARCH:-amd64}" = "arm64" ]; then
    _ISO_TOOLS="docker debootstrap git mke2fs mkfs.vfat mcopy sfdisk truncate"
    [ "$ISO_ARCH" = "$(dpkg --print-architecture 2>/dev/null || echo amd64)" ] \
        || _ISO_TOOLS="$_ISO_TOOLS qemu-aarch64-static"
fi
```

```
for t in $_ISO_TOOLS; do
    command -v "$t" &>/dev/null || { echo -e "${RED}Manquant: $t${NC}"; exit 1; }
done
iso_arm_binfmt
mkdir -p "$WORK" "$ROOTFS" "$ISOROOT"

echo -e "${CYAN}${BOLD}== osmo-operator ISO builder (via build.sh + start-docker.sh) ==${NC}"


# Fin de module : `. fichier` rend le statut de sa DERNIERE commande, et
# build-iso.sh tourne sous set -e. Un module qui finirait par un test faux
# ("[ ... ] && { ...; }") arreterait tout, sans un mot. Toujours 0 ici.
true
```

3.194 /opt/GSM/osmo-operator/iso_modules/30-image-configs.sh

10433 octets, 188 lignes → 188 lignes

```
#!/bin/bash
# iso_modules/30-image-configs.sh - etape 1 (build docker) et 2 (configs via start-docker.sh)
# Source par build-iso.sh, dans l ordre des numeros : meme shell, memes
# variables, memes fonctions. Ne s execute pas seul. `return` en tete de
# module = "rien a faire ici" (c est ainsi que --arm saute une etape).

# -- Etape 1 : LE build docker (une fois ; voir iso_docker_build) ------------
iso_docker_build "$ISO_ROLE"

load_start_lib() {
    local src="$DIR/launch/start-docker.sh"
    local lib="$WORK/start.lib.sh"

    awk '
        BEGIN { skip=0 }
        /^banner[[:space:]]*$/                    { exit }
        /^\[ "\$\{1:-\}" = "stop" \]/             { exit }
        /^\[ "\$\(id -u\)" -ne 0 \]/              { exit }
        /^choose_network_mode[[:space:]]*$/       { exit }
        /^\.\//                                   { exit }
        /^case "\$NETWORK_MODE" in[[:space:]]*$/  { exit }
        { print }
    ' "$src" > "$lib"

    # La lib vit dans $WORK, pas dans le depot : sans ca, la resolution par
    # BASH_SOURCE de start-docker.sh chercherait generate_configs.sh a cote de la copie.
    export OSMO_REPO_DIR="$DIR"

    # shellcheck disable=SC1090
    source "$lib"
}

# -- Etape 2 : la bibliotheque de start-docker.sh (apply_config_templates) ---
# [2026-09-03] Plus de build_run_image ni de Dockerfile.lite ici : une seule
# image docker (etape 1), les configs viennent de ce depot, l elagage lite se
# fait sur le rootfs (etape 8c). load_start_lib reste necessaire : c est lui qui
# apporte apply_config_templates.
load_start_lib
echo -e "${GREEN}[2/9] Image source : ${CYAN}$([ "$ISO_ROLE" = "interstp" ] && [ "${OSMO_ISO_ALL_RUN:-0}" != "1" ]
 ↪  && echo osmocom-stp || echo osmocom-nitb)${NC}"

echo -e "${GREEN}[2b/9] Preparation de l'image source de l'ISO...${NC}"

ISO_N_MS=2
ISO_OP_ID=1         # operateur unique de l'ISO (PLMN 001-01)
ENCRYPTION="a5 1"   # A5/1 par defaut dans l'ISO -- la valeur suit enfin le commentaire
```

```
# L'ISO tourne en NATIF, sans bridge docker. Les 172.20.0.x existaient quand
# meme : 20-dhcp.network (plus bas) les alias sur le NIC par defaut. Mais faire
# ecouter le coeur dessus le rend tributaire de ce NIC - s'il est absent (VM
# sans carte), nomme hors de "en* eth*", ou simplement pas encore configure
# par systemd-networkd quand osmo-ggsn demarre, le bind echoue. La boucle
# locale, elle, est toujours la et prete avant tout service.
# Concerne : osmo-ggsn (gtp bind-ip), osmo-sgsn (ggsn remote-ip), osmo-upf
# (local-addr), osmo-bsc (gprs nsvc remote ip) et le log gsmtap, que l'on
# ramene ainsi sur 127.0.0.1 ou tshark capte deja.
# 127.0.0.2 et non .1 : c'est deja l'adresse que le bloc de patch plus bas
# impose aux MEMES services (gtp local-ip, gsup remote-ip, listen 23000, HLR
# remote-ip). Tant que __CONTAINER_IP__ valait 127.0.0.1, la substitution du
# gabarit et le patch qui la suit divergeaient - le GGSN pouvait annoncer une
# adresse et ecouter sur l'autre. C'est aussi ce qui remplace les 172.20.1.x
# d'avant : une adresse de boucle locale existe toujours, une adresse de NIC
# peut manquer au moment ou le service demarre.
ISO_PRIV_BASE=$(( ${ISO_NODE:-1} + 1 ))
ISO_PRIV_GW="192.168.${ISO_PRIV_BASE}.1"
ISO_PRIV_IP="192.168.${ISO_PRIV_BASE}.10"

# ── L ADRESSE DE BASE EST CELLE DU SEGMENT PRIVE, PLUS LA BOUCLE LOCALE ──────
# ip1 valait 127.0.0.2. Une boucle locale a l avantage d exister toujours, mais
# elle ne se voit que de la machine : deux noeuds ne peuvent rien se dire, et
# surtout SGSN et GGSN se retrouvaient sur LA MEME adresse. Or les deux ouvrent
# le meme socket GTP :
#     osmo-ggsn : gtp bind-ip  127.0.0.2  -> prend 2123/2152/3386 en premier
#     osmo-sgsn : gtp local-ip 127.0.0.2  -> « bind failed: Address already in
#                 use », « FATAL Cannot bind/listen on GTP socket », et systemd
#                 le relance en boucle (compteur a 58 sur le banc).
# Le packet attach restait alors en « [ .. ] » pour toujours, sans qu aucun
# journal du lanceur ne le dise - c est celui du demon qui parle.
#
# Le plan du noeud donne DEUX adresses, c est exactement ce qu il faut :
#     .10  ip1  le coeur : GGSN, NS/Gb du SGSN, UPF, nsvc du BSC
#     .1   gw prive       le point GTP du SGSN, a lui seul
# Elles sont posees en /32 par network/osmo-ip-plan.sh avant que run.sh ne
# demarre quoi que ce soit, et son repli les pose sur `lo` quand aucune carte
# ne fournit Internet : l argument « une boucle locale existe toujours » reste
# donc vrai, sans l inconvenient de l adresse unique.
HOST_IP="$ISO_PRIV_IP"     # ip1 : __CONTAINER_IP__ - ggsn/sgsn-NS/upf/bsc-nsvc
SGSN_GTP_IP="$ISO_PRIV_GW" # le point GTP du SGSN, distinct du GGSN
GATEWAY_IP="127.0.0.1"     # gw  : __GATEWAY_IP__  - log gsmtap + dns 0 du ggsn
# HLR et GSUP restent sur la boucle locale : c est un plan de controle interne
# au noeud, que personne n appelle de l exterieur, et osmo-hlr.cfg y fige son
# « bind ip 127.0.0.2 ». Les deplacer demanderait de bouger les deux ensemble
# sans rien y gagner.
HLR_IP="127.0.0.2"

# ── Le segment prive de ce noeud : 192.168.<noeud+1>.x ──────────────────────
# Meme plan que le cote docker (start-docker.sh : op_private_*), pour qu'une VM et un
# conteneur du meme rang se decrivent pareil. Le +1 laisse 192.168.1.0/24 au
# LAN du banc - un noeud qui s'y poserait entrerait en collision avec les VM et
# le hub SS7.
#
# Ces adresses REMPLACENT les 172.20.x heritees du plan docker. Elles ne
# revendiquent rien (/32) : le but n'est pas de creer un segment - une VM n'a
# pas de BTS derriere une carte - mais de donner un point d'attache stable aux
# configurations qui nomment encore une adresse privee.
# __INTER_STP_IP__ : ASP vers le STP d'un autre operateur. Inerte ici - l'ISO
# n'a qu'un operateur et passe inter_stp_shutdown=shutdown a apply_config_
# templates - mais on ne laisse pas une IP docker morte dans les configs.
INTER_STP_IP="127.0.0.1"   # ip2 : inter-operateur (ASP shutdown sur l'ISO)

echo -e "  Host IP    : ${CYAN}${HOST_IP}${NC}"
echo -e "  Gateway    : ${CYAN}${GATEWAY_IP}${NC}"
```

```
echo -e "  Inter-STP  : ${CYAN}${INTER_STP_IP}${NC}"
echo -e "  MS         : ${CYAN}${ISO_N_MS}${NC}"
echo -e "  Encryption : ${CYAN}${ENCRYPTION}${NC}"

TEMP_CONFIG="$(mktemp -d)"

# ── Point codes et rattachement SS7 ──────────────────────────────────
# Hors WAN : le plan historique, 1.<op>.<role>, et l'ASP inter-STP coupe -
# l'ISO n'a qu'un operateur, il n'a personne a qui parler en SS7.
#
# Avec --node N : le noeud entre DANS le point code, 1.<noeud><op>.<role>.
# Sans ca, trois ISO attachees au meme hub y presenteraient trois fois 1.11.2.
# Un point code est une adresse : deux equipements avec la meme, ce n'est pas
# un conflit de nom, c'est du routage faux - et silencieux.
ISO_PC_MSC="1.1.1"; ISO_PC_STP="1.1.2"; ISO_PC_BSC="1.1.3"
ISO_INTER_SHUT="shutdown"
ISO_INTER_IP="$INTER_STP_IP"
if [ -n "$ISO_NODE" ]; then
    ISO_PC_MSC="1.${ISO_NODE}${ISO_OP_ID}.1"
    ISO_PC_STP="1.${ISO_NODE}${ISO_OP_ID}.2"
    ISO_PC_BSC="1.${ISO_NODE}${ISO_OP_ID}.3"
    ISO_INTER_IP="$ISO_HUB_IP"
    ISO_INTER_SHUT="no shutdown"
    # RCTX unique lui aussi : le hub identifie chaque AS par son routing context.
    export RCTX_INTER_OVERRIDE=$(( ISO_NODE * 1000 + ISO_OP_ID * 100 + 50 ))
    # local-ip de l'ASP laissee a 0.0.0.0 : l'adresse du noeud vient du DHCP et
    # n'est pas forcement montee quand osmo-stp demarre. Se lier a une adresse
    # absente echoue au lancement, sans rapport visible avec le reseau.
    export INTER_LOCAL_IP_OVERRIDE="0.0.0.0"
    echo -e "  Noeud WAN  : ${CYAN}${ISO_NODE}${NC}  PC ${CYAN}${ISO_PC_STP}${NC}  hub ${CYAN}${ISO_HUB_IP}${NC}
     ↪  rctx ${RCTX_INTER_OVERRIDE}"
fi

apply_config_templates "$TEMP_CONFIG" \
    "$HOST_IP" "$GATEWAY_IP" \
    "1" "$ISO_PC_MSC" "$ISO_PC_STP" "$ISO_PC_BSC" \
    "001" "01" "OsmoGSM" \
    "$ISO_INTER_IP" "$ISO_INTER_SHUT" "1"

# ── Role inter-STP : la config du hub, pour N noeuds ─────────────────
if [ "$ISO_ROLE" = "interstp" ]; then
    _hub_nodes=3
    if [ -n "$ISO_WAN_NODES" ]; then
        _hub_nodes=$(printf '%s' "${ISO_WAN_NODES//,/ }" | wc -w)
    fi
    bash "$DIR/helpers/create_interop.sh" --wan "$_hub_nodes" "${ISO_WAN_OPS:-1}" \
        "$TEMP_CONFIG/osmocom/osmo-stp-interop.cfg" || exit 1
    echo -e "  ${GREEN}✓${NC} hub SS7 pour ${CYAN}${_hub_nodes}${NC} noeud(s) × ${ISO_WAN_OPS:-1} operateur(s)"
fi

# ── Les retouches NATIVES ────────────────────────────────────────────
# APRES apply_config_templates, et non avant : celui-ci ecrase systematiquement
# sms-routing.conf, osmo-sgsn.cfg et osmo-msc.cfg avec ce que disent les
# gabarits - c'est-a-dire le plan DOCKER.
#
# Cette recette vivait ICI, en deux morceaux (le sms-routing juste apres la
# substitution, les sed du SGSN et du MSC trois cents lignes plus bas), et
# NULLE PART ailleurs. Une ISO en sortait juste ; une machine qui regenerait
# ses configs ensuite - ./start-native.sh --regen - en sortait fausse, sans
# qu'un seul message ne le dise. Elle est desormais dans generate_configs.sh,
# une fois, et les deux chemins l'appellent (voir apply_native_post_patches).
# La table WAN telle que cette ISO l'embarquera : c'est elle qui donne a
# sms-routing.conf l'adresse de CHAQUE noeud. Sans elle (ISO d'un banc isole),
# le generateur n'ecrit que notre propre entree.
ISO_WAN_TMP=""
if [ -n "$ISO_WAN_NODES" ]; then
```

```
    ISO_WAN_TMP="$(mktemp -p "$WORK")"
    printf 'WAN_NODES="%s"\n' "$ISO_WAN_NODES" > "$ISO_WAN_TMP"
fi
apply_native_post_patches "$TEMP_CONFIG" "$ISO_OP_ID" "$ISO_N_MS" "$HOST_IP" \
    "${ISO_NODE:-1}" "${ISO_WAN_TMP:-/nonexistent}" "$SGSN_GTP_IP" "$HLR_IP"
echo -e "  ${GREEN}✓${NC} retouches natives : sms-routing (${CYAN}${ISO_N_MS}${NC} route(s) MS), GGSN/NS
 ↪  ${CYAN}${HOST_IP}${NC}, GTP SGSN ${CYAN}${SGSN_GTP_IP}${NC}, HLR ${CYAN}${HLR_IP}${NC}"


# Fin de module : `. fichier` rend le statut de sa DERNIERE commande, et
# build-iso.sh tourne sous set -e. Un module qui finirait par un test faux
# ("[ ... ] && { ...; }") arreterait tout, sans un mot. Toujours 0 ici.
true
```

3.195 /opt/GSM/osmo-operator/iso_modules/31-image-source.sh

5114 octets, 84 lignes → 84 lignes

```
#!/bin/bash
# iso_modules/31-image-source.sh - image source de l ISO, controle de suite, image run
# Source par build-iso.sh, dans l ordre des numeros : meme shell, memes
# variables, memes fonctions. Ne s execute pas seul. `return` en tete de
# module = "rien a faire ici" (c est ainsi que --arm saute une etape).

# ── L image source ──────────────────────────────────────────────────
# osmocom-nitb pour tout le monde (un seul build), sauf le hub demande SEUL :
# osmocom-stp, construite a l etape 1. Dans une passe --all, le hub prend aussi
# osmocom-nitb (qui porte osmo-stp) : une image osmocom-stp restee d une autre
# base sur la machine ne doit pas s inviter. OSMO_ISO_SRC_IMAGE force une image.
if [ -n "${OSMO_ISO_SRC_IMAGE:-}" ]; then
    ISO_SRC_IMAGE="$OSMO_ISO_SRC_IMAGE"
elif [ "$ISO_ROLE" = "interstp" ] && [ "${OSMO_ISO_ALL_RUN:-0}" != "1" ] \
     && docker image inspect "osmocom-stp${ISO_IMG_TAG}" >/dev/null 2>&1; then
    ISO_SRC_IMAGE="osmocom-stp${ISO_IMG_TAG}"
else
    ISO_SRC_IMAGE="${IMAGE_NITB:-osmocom-nitb}${ISO_IMG_TAG}"
fi
case "$ISO_ROLE:$ISO_LITE" in
    interstp:*) ISO_RUN_IMAGE="osmocom-stp-iso${ISO_IMG_TAG}" ;;
    *:1)        ISO_RUN_IMAGE="osmocom-run-lite-iso${ISO_IMG_TAG}" ;;
    *)          ISO_RUN_IMAGE="osmocom-run-iso-net-host${ISO_IMG_TAG}" ;;
esac
docker image inspect "$ISO_SRC_IMAGE" >/dev/null 2>&1 \
    || { echo -e "${RED}Image source ${ISO_SRC_IMAGE} introuvable${NC}" >&2; exit 1; }

# ── L image et le rootfs doivent etre de la MEME suite Ubuntu ───────
# /usr/local, /root/.env et /root/.venv-qemu sont copies tels quels : un venv
# noble (python 3.12) sur un rootfs jammy (python 3.10) n a pas d interpreteur,
# et les .so de l image cherchent une glibc que le rootfs n a pas. On lit
# l os-release de l image et on refuse le melange - --version=<suite> aligne.
_img_suite="$(docker run --rm --entrypoint sh "$ISO_SRC_IMAGE" -c '. /etc/os-release; echo "$VERSION_CODENAME"'
 ↪  2>/dev/null | tr -d '[:space:]' || true)"
if [ -n "$_img_suite" ] && [ "$_img_suite" != "$ISO_SUITE" ]; then
    if [ "${OSMO_ISO_SUITE_MISMATCH_OK:-0}" = "1" ]; then
        echo -e "  ${YELLOW}⚠ image ${ISO_SRC_IMAGE} en ${_img_suite}, rootfs en ${ISO_SUITE}
         ↪  (OSMO_ISO_SUITE_MISMATCH_OK=1 : on continue)${NC}"
    else
        echo -e "${RED}L image ${ISO_SRC_IMAGE} est construite sur ${_img_suite}, le rootfs demande est
         ↪  ${ISO_SUITE}.${NC}" >&2
        echo -e "${RED}Relancez avec --version=${_img_suite}, ou reconstruisez l image (./build.sh) sur la base
         ↪  voulue.${NC}" >&2
        exit 1
    fi
fi
```

```
echo -e "  ${GREEN}✓${NC} image source ${CYAN}${ISO_SRC_IMAGE}${NC} (${_img_suite:-suite inconnue}) -> rootfs
 ↪  ${CYAN}${ISO_SUITE}${NC}"
TMP_CID="$(docker create "$ISO_SRC_IMAGE" /bin/sh)"

# Le hub ne porte AUCUN operateur : lui pousser le jeu complet, c'est embarquer
# un osmo-stp.cfg de point-code 1.1.2 avec local-ip 172.20.0.11 a cote du
# osmo-stp-interop.cfg qui, lui, fait autorite. Deux configurations STP dans le
# meme /etc/osmocom, l'une morte mais plausible : on relance la mauvaise, le hub
# se presente au WAN avec le point-code d'un operateur, et le routage M3UA part
# sur une adresse du plan docker que la VM n'a jamais eue.
# On ne copie donc que la config du hub - et pas par lien symbolique : le pgrep
# de start-interstp.sh discrimine sur le NOM du fichier de conf passe a osmo-stp.
# On retire donc la config STP d'operateur - et elle seule. Ne pousser que
# osmo-stp-interop.cfg privait aussi le hub de run.sh, status.sh, check.sh et
# entrypoint.sh, que /etc/osmocom est le seul a fournir (l'image osmocom-stp
# part d'ubuntu:22.04 nu) : le chmod de l'etape 5 s'arretait alors sur un
# fichier absent et la construction du hub - donc ./build-iso.sh sans
# argument, qui commence par lui - echouait apres une heure de travail.
if [ "$ISO_ROLE" = "interstp" ]; then
    rm -f "$TEMP_CONFIG/osmocom/osmo-stp.cfg"
fi
docker cp "$TEMP_CONFIG/osmocom/."  "$TMP_CID:/etc/osmocom/"  2>/dev/null || true
# Le hub n'a pas d'Asterisk : lui pousser des configs SIP n'aurait pas de sens,
# et l'image n'a meme pas /etc/asterisk.
[ "$ISO_ROLE" = "interstp" ] || \
    docker cp "$TEMP_CONFIG/asterisk/." "$TMP_CID:/etc/asterisk/" 2>/dev/null || true

docker commit "$TMP_CID" "$ISO_RUN_IMAGE" >/dev/null
docker rm -f "$TMP_CID" >/dev/null 2>&1 || true
rm -rf "$TEMP_CONFIG"

echo -e "  ${GREEN}✓${NC} image ${CYAN}${ISO_RUN_IMAGE}${NC} prete"

# ── Etape 3 : (SUPPRIME) - ISO NATIF : on n'embarque PAS l'image Docker ────
# L'image osmocom-run ne sert plus que de SOURCE de build (docker cp des binaires
# et configs vers le rootfs a l'etape 6). On ne la save plus dans l'ISO : pas de
# docker au runtime, pas de tar.gz de plusieurs Go embarque.


# Fin de module : `. fichier` rend le statut de sa DERNIERE commande, et
# build-iso.sh tourne sous set -e. Un module qui finirait par un test faux
# ("[ ... ] && { ...; }") arreterait tout, sans un mot. Toujours 0 ici.
true
```

3.196 /opt/GSM/osmo-operator/iso_modules/40-rootfs.sh

4966 octets, 80 lignes → 80 lignes

```
#!/bin/bash
# iso_modules/40-rootfs.sh - etape 4 : debootstrap (ou rootfs herite)
# Source par build-iso.sh, dans l ordre des numeros : meme shell, memes
# variables, memes fonctions. Ne s execute pas seul. `return` en tete de
# module = "rien a faire ici" (c est ainsi que --arm saute une etape).

# ── Etape 4 : Bootstrap rootfs minimal ──────────────────────────────────
# ── Cache des .deb (accelere les rebuilds) ──────────────────────────────
# ISO_DEB_CACHE=<dir absolu> : cache PERSISTANT que debootstrap reutilise
# (--cache-dir) au lieu de re-telecharger la base a chaque build (les lignes
# "I: Retrieving / I: Validating"). Vide ou --no-cache -> pas de cache.
ISO_DEB_CACHE="${ISO_DEB_CACHE:-$HOME/.cache/osmo-iso-debs}"
DEBOOTSTRAP_CACHE_OPT=""
if [ -n "$ISO_DEB_CACHE" ] && [ -z "$NO_CACHE" ]; then
    # Le cache est PAR ARCHITECTURE (arm64 dans son propre dossier) : un .deb
    # amd64 et son homonyme arm64 portent le meme nom de fichier sauf l arch, et
```

```
    # debootstrap ne verifie pas l architecture de ce qu il reprend du cache.
    _DBS_CACHE="$ISO_DEB_CACHE/debootstrap"
    [ "${ISO_ARCH:-amd64}" = "amd64" ] || _DBS_CACHE="$ISO_DEB_CACHE/debootstrap-$ISO_ARCH"
    mkdir -p "$_DBS_CACHE"
    DEBOOTSTRAP_CACHE_OPT="--cache-dir=$_DBS_CACHE"
    echo -e "  ${GREEN}cache .deb debootstrap : $_DBS_CACHE${NC}"
fi
# ── Rootfs HERITE d une passe precedente (--all : lite et desktop reprennent la
# normale) ou debootstrap de zero. Un rootfs herite est complet et configure :
# les etapes 5 (injection) sont sautees, les autres rejouent - elles ecrivent
# leurs fichiers, apt ne pose que le delta.
OSMO_ISO_INHERITED=0
if [ -n "${OSMO_ISO_ROOTFS_FROM:-}" ]; then
    [ -d "$OSMO_ISO_ROOTFS_FROM" ] || { echo -e "${RED}Rootfs herite introuvable : ${OSMO_ISO_ROOTFS_FROM}${NC}"
    ↪  >&2; exit 1; }
    rmdir "$ROOTFS" 2>/dev/null || true
    case "${OSMO_ISO_ROOTFS_MODE:-move}" in
        copy)
            echo -e "${GREEN}[4/9] Rootfs repris (COPIE) de ${CYAN}${OSMO_ISO_ROOTFS_FROM}${NC}..."
            # --reflink=auto : instantane sur btrfs/xfs, copie ordinaire ailleurs.
            cp -a --reflink=auto "$OSMO_ISO_ROOTFS_FROM" "$ROOTFS" ;;
        *)
            echo -e "${GREEN}[4/9] Rootfs repris (deplace) de ${CYAN}${OSMO_ISO_ROOTFS_FROM}${NC}..."
            mv "$OSMO_ISO_ROOTFS_FROM" "$ROOTFS" ;;
    esac
    OSMO_ISO_INHERITED=1
    echo -e "  ${GREEN}✓${NC} rootfs herite $(du -sh "$ROOTFS"|cut -f1) - debootstrap et injection sautes"
else
    # Le script debootstrap de la suite : sur un hote jammy, le paquet ne connait
    # pas noble. Les scripts Ubuntu sont tous le meme (gutsy) : on lie.
    _DBS=/usr/share/debootstrap/scripts
    if [ -d "$_DBS" ] && [ ! -e "$_DBS/$ISO_SUITE" ] && [ -e "$_DBS/gutsy" ]; then
        ln -s gutsy "$_DBS/$ISO_SUITE"
        echo -e "  ${YELLOW}debootstrap ne connaissait pas ${ISO_SUITE} : script gutsy lie${NC}"
    fi
    echo -e "${GREEN}[4/9] debootstrap $ISO_SUITE ($ISO_UBUNTU, ${ISO_ARCH}, minimal)...${NC}"
    # arm64 sur hote x86 : --foreign ne fait que TELECHARGER et DEPAQUETER (rien
    # ne s execute) ; la seconde etape configure les paquets dans le chroot, ou
    # qemu-aarch64-static, copie dans le rootfs, execute les binaires aarch64
    # via binfmt. Sur amd64 le drapeau est vide et debootstrap fait tout d un coup.
    _DBS_FOREIGN=""
    if [ "${ISO_ARCH:-amd64}" != "$(dpkg --print-architecture)" ]; then _DBS_FOREIGN="--foreign"; fi
    # zstd : noble met COMPRESS=zstd dans initramfs.conf, mais minbase ne pose
    # pas le binaire. Sans lui, mkinitramfs (declenche par linux-image dans le
    # chroot) avertit "No zstd in PATH, using gzip" et sort un initrd gzip, plus
    # gros et plus lent a ouvrir. Il doit etre la AVANT le kernel, donc ici.
    debootstrap $DEBOOTSTRAP_CACHE_OPT --arch="$ISO_ARCH" $_DBS_FOREIGN --variant=minbase --include=\
systemd,systemd-sysv,dbus,kmod,zstd,\
ca-certificates,curl,gnupg,\
iproute2,iputils-ping,procps,less,nano \
        "$ISO_SUITE" "$ROOTFS" "$ISO_MIRROR"
    if [ -n "$_DBS_FOREIGN" ]; then
        install -m755 "$(command -v qemu-aarch64-static)" "$ROOTFS/usr/bin/qemu-aarch64-static"
        echo -e "  ${CYAN}debootstrap --second-stage (aarch64 sous qemu-user, plus lent)...${NC}"
        chroot "$ROOTFS" /debootstrap/debootstrap --second-stage >"$WORK/debootstrap-2nd.log" 2>&1 \
            || { echo -e "${RED}debootstrap --second-stage a echoue (voir $WORK/debootstrap-2nd.log)${NC}" >&2;
            ↪  tail -20 "$WORK/debootstrap-2nd.log" >&2; exit 1; }
    fi
    echo -e "  ${GREEN}✓${NC} rootfs base $(du -sh "$ROOTFS"|cut -f1)"
fi


# Fin de module : `. fichier` rend le statut de sa DERNIERE commande, et
# build-iso.sh tourne sous set -e. Un module qui finirait par un test faux
# ("[ ... ] && { ...; }") arreterait tout, sans un mot. Toujours 0 ici.
true
```

3.197 /opt/GSM/osmo-operator/iso_modules/50-injection-image.sh

5509 octets, 92 lignes → 92 lignes

```
#!/bin/bash
# iso_modules/50-injection-image.sh - etape 5 : .deb du build puis docker cp de l image
# Source par build-iso.sh, dans l ordre des numeros : meme shell, memes
# variables, memes fonctions. Ne s execute pas seul. `return` en tete de
# module = "rien a faire ici" (c est ainsi que --arm saute une etape).

# ── Etape 5 : LES .deb DU BUILD DOCKER D ABORD, puis le reste de l image ────
# [2026-09-03] Ce que l image a compile est sorti en paquets (packaging/
# osmo-deb.sh) : libosmocore et les osmo-*, gapk, QEMU et son arbre, osmocom-bb,
# le venv gr-gsm, le firmware. On les pose dans le rootfs par dpkg, comme sur
# n importe quelle machine, et on ne recopie de l image que ce qui n est pas
# en paquet (scripts, configs, unites, node, le depot, les ateliers restants).
# Source des paquets : le cache de l HOTE (/var/cache/osmo-debs, que build.sh
# alimente), sinon le cache embarque dans l image. Seuls ceux de la suite du
# rootfs (~noble) sont pris. Sans paquet, on retombe sur le docker cp complet.
if [ "$OSMO_ISO_INHERITED" = "1" ]; then
    echo -e "${GREEN}[5/9] Injection : rootfs herite, deja fait${NC}"
else
echo -e "${GREEN}[5/9] Injection stack Osmocom (paquets .deb du build, puis image)...${NC}"
CID=$(docker create "$ISO_RUN_IMAGE" /bin/true)
ISO_DEB_HOST_CACHE="${OSMO_DEB_CACHE:-/var/cache/osmo-debs}"
_iso_debs="$WORK/debs-build"; mkdir -p "$_iso_debs"
# Suite ET architecture : le cache de l hote porte les paquets des deux
# architectures cote a cote (..._amd64.deb, ..._arm64.deb).
cp -f "$ISO_DEB_HOST_CACHE"/osmo-build-*"~${ISO_SUITE}_${ISO_ARCH}.deb" "$_iso_debs/" 2>/dev/null || true
if ! ls "$_iso_debs"/*.deb >/dev/null 2>&1; then
    docker cp "$CID:/var/cache/osmo-debs/." "$_iso_debs/" 2>/dev/null || true
    find "$_iso_debs" -name '*.deb' ! -name "*~${ISO_SUITE}_${ISO_ARCH}.deb" -delete 2>/dev/null || true
fi
# Le hub n a besoin que du STP : libosmocore, libosmo-netif, libosmo-sigtran.
if [ "$ISO_ROLE" = "interstp" ]; then
    find "$_iso_debs" -name '*.deb' ! -name 'osmo-build-libosmocore_*' \
        ! -name 'osmo-build-libosmo-netif_*' ! -name 'osmo-build-libosmo-sigtran_*' -delete 2>/dev/null || true
fi
ISO_DEBS_USED=0
_ndebs=$(find "$_iso_debs" -maxdepth 1 -name '*.deb' | wc -l)
if [ "$_ndebs" -gt 0 ]; then
    # Poses via un repertoire DANS le rootfs (dpkg lit ses paquets sous la
    # racine), retire ensuite : les paquets ne voyagent dans l ISO que sur
    # ISO_EMBED_DEBS=1 - un arbre QEMU en zstd dans une image qui tient en RAM,
    # ca ne va pas de soi.
    install -d "$ROOTFS/var/cache/osmo-debs"
    cp -f "$_iso_debs"/*.deb "$ROOTFS/var/cache/osmo-debs/"
    if chroot "$ROOTFS" sh -c 'dpkg -i --force-overwrite /var/cache/osmo-debs/osmo-build-*.deb' \
           > "$WORK/dpkg-debs.log" 2>&1; then
        chroot "$ROOTFS" ldconfig 2>/dev/null || true
        ISO_DEBS_USED=1
        echo -e "  ${GREEN}✓${NC} ${_ndebs} paquets du build docker poses par dpkg ($(du -sh "$_iso_debs" | cut
         ↪  -f1))"
    else
        echo -e "  ${YELLOW}⚠${NC} dpkg -i des paquets du build a echoue (voir $WORK/dpkg-debs.log) - repli sur l
         ↪  image" >&2
        tail -5 "$WORK/dpkg-debs.log" | sed 's/^/     /' >&2
    fi
    [ "${ISO_EMBED_DEBS:-0}" = "1" ] || rm -rf "$ROOTFS/var/cache/osmo-debs"
else
    echo -e "  ${CYAN}aucun paquet .deb du build pour ${ISO_SUITE} (cache ${ISO_DEB_HOST_CACHE}) - tout vient de l
     ↪  image${NC}"
fi
docker cp "$CID:/usr/local/bin/." "$ROOTFS/usr/local/bin/"  2>/dev/null||true
docker cp "$CID:/usr/local/lib/." "$ROOTFS/usr/local/lib/"  2>/dev/null||true
docker cp "$CID:/usr/local/include/." "$ROOTFS/usr/local/include/" 2>/dev/null||true
```

```
docker cp "$CID:/usr/local/sbin/." "$ROOTFS/usr/local/sbin/" 2>/dev/null||true
# /opt/GSM : tout, SAUF les arbres que les paquets viennent de poser (osmocom-bb,
# qosmo, qemu-install, firmware) - et rien du tout pour le hub, qui n en
# lit pas une ligne (le depot lui-meme est clone a l etape 5a).
if [ "$ISO_ROLE" != "interstp" ]; then
    _excl=()
    if [ "$ISO_DEBS_USED" = "1" ]; then
        for _t in osmocom-bb qosmo c54x_exe grgsm_exe qemu-install firmware; do
            [ -d "$ROOTFS/opt/GSM/$_t" ] && _excl+=("--exclude=GSM/$_t" "--exclude=GSM/$_t/*")
        done
    fi
    mkdir -p "$ROOTFS/opt"
    docker cp "$CID:/opt/GSM" - 2>/dev/null | tar -x -C "$ROOTFS/opt" "${_excl[@]+"${_excl[@]}"}" 2>/dev/null ||
     ↪  true
fi
# venv python (gr-gsm + bridges) attendu par le pont (grgsm_exe, pont/*.py)
# - en paquet (grgsm-venv) quand le cache l a, depuis l image sinon.
[ -d "$ROOTFS/root/.env" ] || docker cp "$CID:/root/.env" "$ROOTFS/root/" 2>/dev/null||true
[ -d "$ROOTFS/root/.venv-qemu" ] || docker cp "$CID:/root/.venv-qemu" "$ROOTFS/root/" 2>/dev/null||true
docker cp "$CID:/opt/node"            "$ROOTFS/opt/"        2>/dev/null||true
docker cp "$CID:/etc/osmocom/."       "$ROOTFS/etc/osmocom/" 2>/dev/null||true
docker cp "$CID:/etc/asterisk/."      "$ROOTFS/etc/asterisk/" 2>/dev/null||true
for svc in osmo-bts-trx osmo-bsc osmo-msc osmo-hlr osmo-mgw osmo-stp osmo-ggsn osmo-sgsn osmo-pcu
 ↪  osmo-sip-connector; do
    docker cp "$CID:/lib/systemd/system/${svc}.service" "$ROOTFS/lib/systemd/system/" 2>/dev/null||true
done
docker rm "$CID" &>/dev/null
echo -e "  ${GREEN}✓${NC} binaires + libs + configs injectes"
fi


# Fin de module : `. fichier` rend le statut de sa DERNIERE commande, et
# build-iso.sh tourne sous set -e. Un module qui finirait par un test faux
# ("[ ... ] && { ...; }") arreterait tout, sans un mot. Toujours 0 ici.
true
```

3.198 /opt/GSM/osmo-operator/iso_modules/51-depot.sh

5824 octets, 104 lignes → 102 lignes (1 groupes compactés)

```
#!/bin/bash
# iso_modules/51-depot.sh - etape 5a : osmo-operator a jour, coeur.env
# Source par build-iso.sh, dans l ordre des numeros : meme shell, memes
# variables, memes fonctions. Ne s execute pas seul. `return` en tete de
# module = "rien a faire ici" (c est ainsi que --arm saute une etape).

# ── osmo-operator : ARBRE a jour depuis GitHub, AVEC son .git ──────────────
# La copie docker cp ci-dessus peut etre perimee ; on avance la branche main du
# depot (start-native.sh, run.sh, scripts/, configs/, build-iso.sh...) dans l'ISO.
EGPRS_BRANCH="${OSMO_EGPRS_BRANCH:-main}"
EGPRS_REPO="${OSMO_EGPRS_REPO:-https://github.com/bbaranoff/osmo-operator}"
echo -e "${GREEN}[5a/9] Clone de osmo-operator (branche ${EGPRS_BRANCH})...${NC}"
# [2026-08-31] CLONE, PAS "fetch + merge --ff-only" SUR L ARBRE DE L IMAGE.
# L ancienne version avancait en place le depot venu du docker cp. Elle avait
# deux defauts qui se voyaient au moment ou l on veut justement graver :
#   - --ff-only echoue des que l arbre de l image porte le moindre commit local
#     ou diverge, et la branche d echec se contentait d un ⚠ jaune : l ISO se
#     construisait alors avec l arbre de l IMAGE, c est-a-dire avec du code
#     vieux de la derniere reconstruction docker, sans que rien n arrete le
#     build. On croyait graver son travail, on gravait celui d avant.
#   - trois chemins (arbre avec .git / arbre sans .git / rien) pour un seul
#     besoin : avoir le depot a jour dans l ISO.
# Un clone rend le resultat previsible : ce qui est sur la branche est ce qui
# est grave, point. Le .git est conserve - c est un clone, pas un tarball - donc
```

```
# update.sh peut toujours faire son fetch au demarrage plutot que d effacer et
# recloner a chaque boot (wipe=1), et l on sait sur quel commit on tourne.
#
# On clone A COTE puis on bascule : si le reseau manque, l arbre de l image
# reste en place. Effacer d abord donnerait une ISO sans depot du tout.
EGPRS_TREE="$ROOTFS/opt/GSM/osmo-operator"
EGPRS_TMP="$WORK/osmo-operator-clone"
rm -rf "$EGPRS_TMP"
if [ "$OSMO_ISO_INHERITED" = "1" ] && [ -d "$EGPRS_TREE/.git" ]; then
    echo -e "  ${GREEN}✓${NC} osmo-operator : arbre du rootfs herite conserve"
elif GIT_TERMINAL_PROMPT=0 git clone --depth 1 -b "$EGPRS_BRANCH" "$EGPRS_REPO" "$EGPRS_TMP" >/dev/null 2>&1; then
    rm -rf "$EGPRS_TREE"
    mkdir -p "$ROOTFS/opt/GSM"
    mv "$EGPRS_TMP" "$EGPRS_TREE"
    echo -e "  ${GREEN}✓${NC} osmo-operator clone (${EGPRS_BRANCH}, .git conserve) - $(git -C "$EGPRS_TREE" log
    ↪  -1 --format='%h %s')"
else
    rm -rf "$EGPRS_TMP"
    if [ -d "$EGPRS_TREE" ]; then
        echo -e "  ${YELLOW}⚠${NC} osmo-operator : clone impossible (reseau ?) - arbre de l'image conserve" >&2
    else
        echo -e "  ${RED}✗${NC} osmo-operator : clone impossible ET absent de l'image" >&2
    fi
fi

# ── tests : /opt/GSM/tests depuis GitHub (bbaranoff), AVEC son .git ──────────
# Meme logique que osmo-operator : clone a cote puis bascule, pour qu un reseau
# absent laisse l arbre existant de l image en place (jamais une ISO sans tests
# si l image en avait deja).
TESTS_BRANCH="${OSMO_TESTS_BRANCH:-main}"
TESTS_REPO="${OSMO_TESTS_REPO:-https://github.com/bbaranoff/tests}"
TESTS_TREE="$ROOTFS/opt/GSM/tests"
TESTS_TMP="$WORK/tests-clone"
echo -e "${GREEN}[5a/9] Clone de tests (branche ${TESTS_BRANCH})...${NC}"
rm -rf "$TESTS_TMP"
if [ "$OSMO_ISO_INHERITED" = "1" ] && [ -d "$TESTS_TREE/.git" ]; then
    echo -e "  ${GREEN}✓${NC} tests : arbre du rootfs herite conserve"
elif GIT_TERMINAL_PROMPT=0 git clone --depth 1 -b "$TESTS_BRANCH" "$TESTS_REPO" "$TESTS_TMP" >/dev/null 2>&1; then
    rm -rf "$TESTS_TREE"
    mkdir -p "$ROOTFS/opt/GSM"
    mv "$TESTS_TMP" "$TESTS_TREE"
    echo -e "  ${GREEN}✓${NC} tests clone (${TESTS_BRANCH}, .git conserve) - $(git -C "$TESTS_TREE" log -1
    ↪  --format='%h %s')"
else
    rm -rf "$TESTS_TMP"
    if [ -d "$TESTS_TREE" ]; then
        echo -e "  ${YELLOW}⚠${NC} tests : clone impossible (reseau ?) - arbre de l'image conserve" >&2
    else
        echo -e "  ${YELLOW}⚠${NC} tests : clone impossible ET absent de l'image" >&2
    fi
fi

# ── Feed HLR : aligner N_MS sur le nombre de MS embarques ────────────────────
# run_modules/21-abonnes-hlr.sh retombe sur ": "${N_MS:=1}"" : sans ce fichier
# un SEUL abonne etait provisionne alors que l'ISO en declare ISO_N_MS, et les
# MS suivants se voyaient refuser le rattachement ("IMSI unknown in HLR") -
# panne lue a tort comme un defaut radio.
#
# PAS dans /opt/GSM/osmo-operator/environment : ce fichier n'est pas dans git. Il y
# a survecu au demarrage tant que personne ne mettait le depot a jour, et pas une
# minute de plus - a l'epoque osmo-update.service effacait et reclonait l'arbre
# a chaque boot (wipe=1), aujourd'hui "osmo-update" fait un git fetch, dont le
# reset --hard emporte de la meme facon ce qui n'est pas suivi. N_MS retombait
# a 1, MS#2 restait inconnu du HLR, et start-native.sh le lancait quand meme.
# /opt/GSM/qosmo/environment, lui, n'existe pas : ce depot-la nomme son
# repertoire "environnement".
```

```
# /etc/osmocom n'appartient a aucun depot : ce qui y est ecrit reste.
mkdir -p "$ROOTFS/etc/osmocom"
cat > "$ROOTFS/etc/osmocom/coeur.env" <<COEUR
# coeur.env - genere par build-iso.sh. Aligne le nombre d'abonnes provisionnes
# dans le HLR sur le nombre de MS embarques par l'ISO (ISO_N_MS).
: "\${N_MS:=$ISO_N_MS}"
: "\${OPERATOR_ID:=1}"
COEUR
echo -e "  ${GREEN}✓${NC} coeur.env : ${CYAN}N_MS=${ISO_N_MS}${NC} (/etc/osmocom)"
  ×3
# Fin de module : `. fichier` rend le statut de sa DERNIERE commande, et
# build-iso.sh tourne sous set -e. Un module qui finirait par un test faux
# ("[ ... ] && { ...; }") arreterait tout, sans un mot. Toujours 0 ici.
true
```

3.199 /opt/GSM/osmo-operator/iso_modules/51-softsim.sh

4060 octets, 81 lignes → 81 lignes

```
#!/bin/bash
# iso_modules/51-softsim.sh - etape 5a ter : softSIM (SIM via SAP, Osmocom)
# Source par build-iso.sh, dans l ordre des numeros. `return` en tete = rien a faire.
#
# softSIM (Ruby) : clone depuis gitea.osmocom.org dans /opt/GSM/softsim, avec son .git.
# Ruby 3.2+ : File.exists? et Fixnum n existent plus ; le shim tools/softsim/ruby_compat.rb
# est pose dans /usr/local/share/softsim/ (usage : ruby -r<shim> demo_server.rb ...).
# Ruby et ruby-libxml sont dans PKGS de 80-chroot.sh.
#
# Desactivable : --no-softsim (OSMO_ISO_SOFTSIM=0).
if [ "${OSMO_ISO_SOFTSIM:-1}" != "1" ]; then
    echo -e "${YELLOW}[5a/9] softSIM desactive (--no-softsim)${NC}"
    return 0
fi

# [2026-10-09] Le hub inter-STP n a pas de SIM a emuler (aucun MS) : 50-injection
# saute deja son /opt/GSM. On ne clone donc pas softsim pour lui.
if [ "$ISO_ROLE" = "interstp" ]; then
    echo -e "${CYAN}[5a/9] Role inter-STP : pas de softSIM (hub M3UA)${NC}"
    return 0
fi

SOFTSIM_REPO="${OSMO_SOFTSIM_REPO:-https://gitea.osmocom.org/sim-card/softsim.git}"
SOFTSIM_LOCAL="${OSMO_SOFTSIM_LOCAL:-/opt/GSM/softsim}"
SOFTSIM_TREE="$ROOTFS/opt/GSM/softsim"
SOFTSIM_TMP="$WORK/softsim-clone"
echo -e "${GREEN}[5a/9] softSIM (SIM via SAP)...${NC}"
rm -rf "$SOFTSIM_TMP"
if [ "$OSMO_ISO_INHERITED" = "1" ] && [ -d "$SOFTSIM_TREE/.git" ]; then
    echo -e "  ${GREEN}✓${NC} softSIM : arbre du rootfs herite conserve"
elif GIT_TERMINAL_PROMPT=0 git clone --depth 1 "$SOFTSIM_REPO" "$SOFTSIM_TMP" >/dev/null 2>&1; then
    rm -rf "$SOFTSIM_TREE"
    mkdir -p "$ROOTFS/opt/GSM"
    mv "$SOFTSIM_TMP" "$SOFTSIM_TREE"
    echo -e "  ${GREEN}✓${NC} softSIM clone ($(git -C "$SOFTSIM_TREE" log -1 --format='%h %s'))"
else
    rm -rf "$SOFTSIM_TMP"
    echo -e "  ${YELLOW}⚠${NC} softSIM : clone impossible (reseau ?)" >&2
fi
# Copie locale de reference, si l hote en a une (comme 51-shannon.sh)
if [ -d "$SOFTSIM_LOCAL" ] && [ "$SOFTSIM_LOCAL" != "$SOFTSIM_TREE" ]; then
    mkdir -p "$SOFTSIM_TREE"
    cp -a "$SOFTSIM_LOCAL/." "$SOFTSIM_TREE/" 2>/dev/null \
        && echo -e "  ${GREEN}✓${NC} softSIM : copie locale de $SOFTSIM_LOCAL posee" \
        || echo -e "  ${YELLOW}⚠${NC} softSIM : copie locale incomplete" >&2
```

```
fi

# COMP128v1 pour la SIM (Ki du HLR) : lib/comp128.rb + patch de simos_server.rb.
# Le patch n est applique qu une fois (un arbre local deja patche le garde).
if [ -d "$SOFTSIM_TREE/src" ] && [ -f "$DIR/tools/softsim/comp128.rb" ]; then
    install -Dm644 "$DIR/tools/softsim/comp128.rb" "$SOFTSIM_TREE/src/lib/comp128.rb"
    # [2026-10-14] TOUS les patches de tools/softsim (0001 COMP128v1, 0002 client
    # suivant, 0003 journal horodate, 0004 deconnexion pcsc idempotente), comme
    # tools/osmo-sim-install.sh : avant, seul le 0001 partait dans l ISO.
    for _sp in "$DIR"/tools/softsim/*.patch; do
        [ -f "$_sp" ] || continue
        if git -C "$SOFTSIM_TREE" apply --check -R "$_sp" 2>/dev/null; then
            echo -e "  ${GREEN}✓${NC} softSIM : $(basename "$_sp") deja present"
        elif git -C "$SOFTSIM_TREE" apply "$_sp" 2>/dev/null; then
            echo -e "  ${GREEN}✓${NC} softSIM : $(basename "$_sp") applique"
        else
            echo -e "  ${YELLOW}⚠${NC} softSIM : $(basename "$_sp") ne s applique pas" >&2
        fi
    done
    unset _sp
    # SIM de l abonne du HLR (IMSI 001010001000001) : le Ki vient de SOFTSIM_KI
    if python3 "$DIR/tools/softsim/make_sim_op1.py" "$SOFTSIM_TREE/src/sim.xml" \
            "$SOFTSIM_TREE/src/sim-op1.xml" --imsi "${OSMO_SIM_IMSI:-001010001000001}" >/dev/null 2>&1; then
        echo -e "  ${GREEN}✓${NC} softSIM : sim-op1.xml (IMSI ${OSMO_SIM_IMSI:-001010001000001})"
    else
        echo -e "  ${YELLOW}⚠${NC} softSIM : sim-op1.xml non genere" >&2
    fi
fi

# Shim Ruby 3.2+ : depuis le depot de construction
if [ -f "$DIR/tools/softsim/ruby_compat.rb" ]; then
    install -Dm644 "$DIR/tools/softsim/ruby_compat.rb" "$ROOTFS/usr/local/share/softsim/ruby_compat.rb"
    echo -e "  ${GREEN}✓${NC} shim Ruby : /usr/local/share/softsim/ruby_compat.rb"
fi

true
```

3.200 /opt/GSM/osmo-operator/iso_modules/52-qemu.sh

33705 octets, 564 lignes → 564 lignes

```
#!/bin/bash
# iso_modules/52-qemu.sh - etape 5b/5c/5d : QEMU, firmware, toast, lanceurs, configs
# Source par build-iso.sh, dans l ordre des numeros : meme shell, memes
# variables, memes fonctions. Ne s execute pas seul. `return` en tete de
# module = "rien a faire ici" (c est ainsi que --arm saute une etape).

# ── qosmo : arbre ELAGUE + binaire installe ──────────────────────────
# Deux choses distinctes, et l'ISO a besoin des DEUX :
#   - l'arbre qosmo (run.sh, run_modules/, environnement/) :
#     c'est LUI le mode qemu de start-native.sh. Il reste dans l'image, prive de
#     .git et de build/ (voir plus bas).
#   - le binaire qemu-system-arm, installe dans /usr/local/bin, et relie depuis
#     l'arbre sous le nom que paths.env cherche (build/qemu-system-arm).
QEMU_BUILD_LOCAL="${OSMO_QEMU_BUILD:-${OSMO_QEMU_SRC:-/opt/GSM/qosmo}/build}"
# --arm : un build QEMU de l hote est un binaire x86, il n a rien a faire dans
# un rootfs arm64. Seul le binaire venu de l image (arm64) compte.
[ "${ISO_ARCH:-amd64}" = "amd64" ] || QEMU_BUILD_LOCAL="/nonexistent/arm64-pas-de-build-hote"
echo -e "${GREEN}[5b/9] Installation QEMU (artefacts seuls, depuis ${QEMU_BUILD_LOCAL})...${NC}"
# L'elagage est HORS de la condition, et l'absence du binaire est FATALE. Avant,
# les deux etaient dans la branche "binaire present" : sur une machine ou QEMU
# n'avait pas ete recompile - le cas courant - on tombait dans le repli, qui se
# contentait d'un avertissement jaune, et l'arbre venu du docker cp
# ($CID:/opt/GSM) partait tel quel dans le squashfs, build/ compris : 1,7 Go
```

```
# dans une ISO qui tient en RAM. Le message passait inapercu au milieu d'une
# heure de construction, et la taille de l'ISO etait le seul indice.
# Echouer ici coute une relance ; ne pas echouer coute une ISO inutilisable
# (sans qemu-system-arm, MS#1 ne demarre pas) et deux fois plus lourde.
# ── qosmo : l'arbre part ENTIER, .git et build/ compris ──────────────
# [2026-08-27] L'effacement pur ("rm -rf $ROOTFS/opt/GSM/qosmo") reglait le
# poids, mais retirait de l'image le depot dont run.sh, run_modules/ et
# environnement/ SONT le mode qemu : l'ISO ne savait plus emuler le Calypso par
# elle-meme et dependait, a CHAQUE demarrage, d'un reclone GitHub par
# osmo-update.service. Pas de reseau au boot = pas de MS. Et l'arbre reclone
# arrivait sans build/, donc sans QEMU_BIN : la pile s'arretait au premier
# module alors que le binaire etait la, dans le PATH.
#
# On ne retire donc plus RIEN de cet arbre :
#   - .git (96 Mo) : c'est lui qui fait la difference entre une mise a jour
#     incrementale (git fetch) et un reclone complet. Sans .git, update.sh
#     n'avait pas le choix : il effacait et reclonait a chaque demarrage.
#   - build/ (1,5 Go d'objets) : il porte le qemu-system-arm COMPILE, celui que
#     environnement/paths.env cherche sous $QEMU_TREE/build/qemu-system-arm.
#     L'arbre embarque est donc utilisable tel quel, sans reseau et sans lien.
#
# Ce que ca coute : ~1,6 Go de plus dans le squashfs (moins une fois compresse).
# A surveiller si l'ISO doit tenir en RAM (toram).
# ── OSMO_QEMU_SRC : L'ARBRE LOCAL PREND LE PAS, QUAND ON LE DEMANDE ──────
# [2026-08-30] L'image docker clone qosmo depuis GitHub (Dockerfile:433).
# Un correctif fait ICI, dans l'arbre local, ne partait donc PAS dans l'ISO --
# il fallait le pousser sur GitHub d'abord, et rien ne le disait. C'est ainsi
# que les correctifs du shunt DSP (publication du Kc depuis le NDB) auraient pu
# etre "appliques" et absents de l'image produite.
# OSMO_QEMU_SRC=/chemin force desormais l'arbre local, en remplacant celui de
# l'image. Sans la variable, rien ne change : l'image fait foi, comme avant.
QSRC="$ROOTFS/opt/GSM/qosmo"
if [ -n "${OSMO_QEMU_SRC:-}" ] && [ -d "$OSMO_QEMU_SRC" ]; then
    rm -rf "$QSRC"
    mkdir -p "$ROOTFS/opt/GSM"
    cp -a "$OSMO_QEMU_SRC" "$QSRC"
    echo -e "  ${GREEN}✓${NC} qosmo FORCE depuis ${CYAN}${OSMO_QEMU_SRC}${NC} (OSMO_QEMU_SRC) ($(du -sh "$QSRC" |
     ↪  cut -f1))"
elif [ -d "$QSRC" ]; then
    echo -e "  ${GREEN}✓${NC} qosmo conserve ENTIER ($(du -sh "$QSRC" | cut -f1), .git + build/ compris)"
else
    # L'image ne l'avait pas : on prend l'arbre de l'hote, entier lui aussi.
    QSRC_HOST="${OSMO_QEMU_SRC:-/opt/GSM/qosmo}"
    if [ -d "$QSRC_HOST" ]; then
        mkdir -p "$ROOTFS/opt/GSM"
        cp -a "$QSRC_HOST" "$QSRC"
        echo -e "  ${GREEN}✓${NC} qosmo repris de l'hote ${CYAN}${QSRC_HOST}${NC} ($(du -sh "$QSRC" | cut -f1))"
    else
        echo -e "  ${YELLOW}!${NC} qosmo introuvable (ni image, ni hote) - l'ISO n'aura pas le mode qemu" >&2
    fi
fi

# ── Les anciens forks quittent le rootfs ──────────────────────────────
# [2026-09-25] L'image docker clone encore qosmo-grgsm et qosmo-dsp (Dockerfile),
# et 50-injection-image.sh recopie tout /opt/GSM et /usr/local/bin de l'image :
# les deux arbres (~1,6 Go avec build/), le prefixe qemu-dsp-install et leurs
# lanceurs arrivaient donc dans l'ISO sans que plus rien ne les lise. Le lien
# /opt/GSM/calypso_dsp.txt de l'image pointait dans qosmo-grgsm ; il est repose
# plus bas sur c54x_exe/rom.
for _old in qosmo-grgsm qosmo-dsp qemu-dsp-install; do
    [ -e "$ROOTFS/opt/GSM/$_old" ] || continue
    rm -rf "$ROOTFS/opt/GSM/$_old"
    echo -e "  ${GREEN}✓${NC} ancien arbre ${CYAN}/opt/GSM/$_old${NC} retire du rootfs"
done
rm -f "$ROOTFS"/usr/local/bin/qosmo-grgsm "$ROOTFS"/usr/local/bin/qosmo-dsp \
      "$ROOTFS"/usr/local/bin/qosmo-grgsm-launch "$ROOTFS"/usr/local/bin/qosmo-dsp-launch
```

```
[ -L "$ROOTFS/opt/GSM/calypso_dsp.txt" ] && [ ! -e "$ROOTFS/opt/GSM/calypso_dsp.txt" ] \
    && rm -f "$ROOTFS/opt/GSM/calypso_dsp.txt"

# ── c54x_exe et grgsm_exe : les deux couches 1 hors QEMU ──────────────────
# [2026-09-25] Les forks qosmo-dsp et qosmo-grgsm sont remplaces par UN arbre
# QEMU, /opt/GSM/qosmo (run.sh, run_modules, cfgs, environnement y ont ete
# repris), et deux executables qui compilent les sources de qosmo :
#   c54x_exe   le DSP TMS320C54x sur la mask-ROM TI (start-native.sh --dsp,
#              BANC_DSP = /opt/GSM/c54x_exe/run.sh)
#   grgsm_exe  la couche 1 gr-gsm hors QEMU (pont : /usr/local/bin/grgsm_exe)
# OSMO_C54X_SRC / OSMO_GRGSM_EXE_SRC forcent un arbre local, comme
# OSMO_QEMU_SRC pour qosmo. Non fatal : sans c54x_exe, seul --dsp manque.
# ⚠ c54x_exe se compile en -march=native : le binaire de l'hote peut ne pas
# tourner sur la machine qui boote l'ISO (make -C /opt/GSM/c54x_exe la-bas).
for _e in c54x_exe grgsm_exe; do
    case "$_e" in
        c54x_exe)  _esrc="${OSMO_C54X_SRC:-}" ;;
        grgsm_exe) _esrc="${OSMO_GRGSM_EXE_SRC:-}" ;;
    esac
    _edst="$ROOTFS/opt/GSM/$_e"
    if [ -n "$_esrc" ] && [ -d "$_esrc" ]; then
        rm -rf "$_edst"; mkdir -p "$ROOTFS/opt/GSM"
        cp -a "$_esrc" "$_edst"
        echo -e "  ${GREEN}✓${NC} $_e FORCE depuis ${CYAN}${_esrc}${NC} ($(du -sh "$_edst" | cut -f1))"
    elif [ -d "$_edst" ]; then
        echo -e "  ${GREEN}✓${NC} $_e : arbre de l image conserve ($(du -sh "$_edst" | cut -f1))"
    elif [ -d "/opt/GSM/$_e" ]; then
        mkdir -p "$ROOTFS/opt/GSM"
        cp -a "/opt/GSM/$_e" "$_edst"
        echo -e "  ${GREEN}✓${NC} $_e repris de l hote ${CYAN}/opt/GSM/$_e${NC} ($(du -sh "$_edst" | cut -f1))"
    else
        echo -e "  ${YELLOW}!${NC} $_e introuvable (ni image, ni hote)" >&2
        continue
    fi
    if [ -x "$_edst/$_e" ]; then
        echo -e "  ${GREEN}✓${NC} $_e : binaire ${CYAN}/opt/GSM/$_e/$_e${NC} present"
    else
        echo -e "  ${YELLOW}!${NC} $_e : binaire ABSENT - make -C /opt/GSM/$_e avant de graver" >&2
    fi
    # Les petits lanceurs /usr/local/bin/<exe> (bash) de l'hote.
    if [ -x "/usr/local/bin/$_e" ] && [ ! -e "$ROOTFS/usr/local/bin/$_e" ]; then
        install -Dm755 "/usr/local/bin/$_e" "$ROOTFS/usr/local/bin/$_e"
        echo -e "  ${GREEN}✓${NC} lanceur ${CYAN}/usr/local/bin/$_e${NC} (repris de l'hote)"
    fi
done

# ── ROMs du DSP TMS320C54x : sans elles, --dsp ne demarre pas ─────────────
# [2026-09-03] Sept dumps du silicium (PROM0..3, DROM, PDROM, Registers), lus
# par c54x_exe sous son --rom-dir (defaut /opt/GSM). ~330 Ko.
# [2026-09-25] Ils ne sont dans AUCUN depot : on les prend sur l'hote, dans
# /opt/GSM puis dans /opt/GSM/c54x_exe/rom (copie de secours, non suivie par
# git, avec calypso_dsp.txt). Non fatal.
_ROM_SRC="${OSMO_DSP_ROM_DIR:-/opt/GSM}"
_ROM_SECOURS=/opt/GSM/c54x_exe/rom
_rom_ok=0; _rom_miss=""
for _r in PROM0 PROM1 PROM2 PROM3 DROM PDROM Registers; do
    if [ -f "$_ROM_SRC/calypso_dsp.$_r.bin" ]; then
        install -Dm644 "$_ROM_SRC/calypso_dsp.$_r.bin" "$ROOTFS/opt/GSM/calypso_dsp.$_r.bin"
        _rom_ok=$((_rom_ok + 1))
    elif [ -f "$_ROM_SECOURS/calypso_dsp.$_r.bin" ]; then
        install -Dm644 "$_ROM_SECOURS/calypso_dsp.$_r.bin" "$ROOTFS/opt/GSM/calypso_dsp.$_r.bin"
        _rom_ok=$((_rom_ok + 1))
    elif [ -f "$ROOTFS/opt/GSM/calypso_dsp.$_r.bin" ]; then
        _rom_ok=$((_rom_ok + 1))
    else
        _rom_miss="$_rom_miss $_r"
```

```
    fi
done
# calypso_dsp.txt (listing de la ROM) : /opt/GSM/calypso_dsp.txt pointait dans
# l'ancien qosmo-grgsm ; il pointe maintenant sur la copie de c54x_exe/rom.
if [ -f "$ROOTFS/opt/GSM/c54x_exe/rom/calypso_dsp.txt" ]; then
    ln -sfn /opt/GSM/c54x_exe/rom/calypso_dsp.txt "$ROOTFS/opt/GSM/calypso_dsp.txt"
elif [ -f "$_ROM_SECOURS/calypso_dsp.txt" ]; then
    install -Dm644 "$_ROM_SECOURS/calypso_dsp.txt" "$ROOTFS/opt/GSM/calypso_dsp.txt"
fi
if [ -z "$_rom_miss" ]; then
    echo -e "  ${GREEN}✓${NC} ROMs DSP : ${CYAN}/opt/GSM/calypso_dsp.{PROM0..3,DROM,PDROM,Registers}.bin${NC}
     ↪  (7/7)"
elif [ -d "$ROOTFS/opt/GSM/c54x_exe" ]; then
    echo -e "  ${YELLOW}!${NC} ROMs DSP incompletes (${_rom_ok}/7, manquent :${_rom_miss}) -
     ↪  OSMO_DSP_ROM_DIR=/chemin ; --dsp ne demarrera pas" >&2
fi

# ── Firmware Calypso : /opt/GSM/firmware, et rien d'autre ────────────────
# [2026-08-28] Il y avait ici un bloc qui remplacait $QSRC/target/firmware par
# un lien vers /opt/GSM/firmware. Il reparait une coquille vide laissee dans
# l'arbre qosmo, sur laquelle la premiere branche de
# environnement/paths.env tombait, d'ou :
#
#   [FAIL] FIRMWARE_ELF (/opt/GSM/qosmo/target/firmware/board/compal_e88/layer1.highram.elf)
#
# La cause a ete traitee a sa source : paths.env (et local.env) du depot qemu
# ne connaissent plus qu'un seul chemin, $GSM_ROOT/firmware. Il n'y a donc plus
# de coquille a reparer, et poser le lien reintroduirait justement le deuxieme
# arbre qu'on vient de supprimer. Constate sur le banc 192.168.1.7 : ce lien
# n'existait meme pas sur l'ISO gravee, et le run chargeait deja
# /opt/GSM/firmware/board/compal_e88/layer1.highram.elf sans lui.
#
# Reste ce qui a de la valeur : verifier que le firmware EST dans le rootfs.
# Sans lui, l'ISO demarre et le MS ne part pas.
FW_ELF="board/compal_e88/layer1.highram.elf"
if [ -e "$ROOTFS/opt/GSM/firmware/$FW_ELF" ]; then
    echo -e "  ${GREEN}✓${NC} firmware : ${CYAN}/opt/GSM/firmware/${FW_ELF}${NC} (source unique ; FIRMWARE_ELF
     ↪  resolu)"
else
    echo -e "  ${YELLOW}!${NC} /opt/GSM/firmware/${FW_ELF} absent du rootfs - FIRMWARE_ELF restera non resolu" >&2
fi
# [2026-09-03] Le firmware est interchangeable (compal_e86, gta0x, ...) : les
# lanceurs lisent l1s/last_rach dans l'ELF choisi par FIRMWARE_ELF, plus besoin
# de nm ni d'adresses figees. On dit quels boards l'image embarque, pour que
# « FIRMWARE_ELF=.../board/X/layer1.highram.elf » ne soit pas une devinette.
_boards=""
for _b in "$ROOTFS"/opt/GSM/firmware/board/*/layer1.highram.elf; do
    [ -f "$_b" ] || continue
    _boards="$_boards $(basename "$(dirname "$_b")")"
done
[ -n "$_boards" ] && echo -e "  ${GREEN}✓${NC} boards layer1 embarques :${CYAN}${_boards}${NC}"

# ── Firmware audio TI TAS2781 (ampli Lenovo Legion 7) ────────────────────
# Le codec TAS2781 des Legion 7 ne sort AUCUN son tant que son firmware n est
# pas dans /lib/firmware : le pilote snd_soc_tas2781 le reclame au chargement
# et reste muet sinon. On le pose dans le rootfs (donc /lib/firmware du systeme
# installe), pas dans le /lib/firmware de l hote de build. Non fatal : une image
# sans ce binaire boote quand meme, seul l audio du Legion manque.
_TAS_URL="https://github.com/bbaranoff/sound_firmware_lenovo_legion_7/raw/refs/heads/main/TIAS2781RCA2.bin"
install -d "$ROOTFS/lib/firmware"
if [ "${ISO_ARCH:-amd64}" != "amd64" ]; then
    echo -e "  ${CYAN}·${NC} firmware TAS2781 (Legion 7, x86) : sans objet sur le Pi"
elif wget -qO "$ROOTFS/lib/firmware/TIAS2781RCA2.bin" "$_TAS_URL"; then
    echo -e "  ${GREEN}✓${NC} firmware audio : ${CYAN}/lib/firmware/TIAS2781RCA2.bin${NC} (TAS2781, Legion 7)"
else
    rm -f "$ROOTFS/lib/firmware/TIAS2781RCA2.bin"
```

```
    echo -e "  ${YELLOW}!${NC} TIAS2781RCA2.bin non telecharge (reseau ?) - audio Legion 7 muet" >&2
fi

# ── toast : codec GSM 06.10 (quut.com), absent du paquet libgsm1 ──────────────
# libgsm1 fournit la lib, pas le binaire toast/untoast/tcat. On compile les
# sources SOUS $ROOTFS/opt/GSM (donc /opt/GSM du systeme installe) et on pose le
# binaire dans $ROOTFS/usr/local/bin (/usr/local/bin du systeme). La compilation
# tourne sur l hote de build ; l ISO etant amd64 sur hote amd64, le binaire est
# bon. Non fatal : sans reseau ou sans toolchain, l ISO se construit sans toast.
_GSM_VER=gsm-1.0.24
_GSM_DIR=gsm-1.0-pl24   # le tarball se decompresse SOUS ce nom
_GSM_URL="https://www.quut.com/gsm/${_GSM_VER}.tar.gz"
if [ -x "$ROOTFS/usr/local/bin/toast" ]; then
    echo -e "  ${GREEN}✓${NC} toast deja dans le rootfs (image ?)"
elif [ "${ISO_ARCH:-amd64}" != "$(dpkg --print-architecture)" ]; then
    echo -e "  ${CYAN}·${NC} toast : compile DANS le chroot ${ISO_ARCH} plus loin (82-arm-natif), pas sur l hote"
elif ! command -v make >/dev/null 2>&1 || ! command -v cc >/dev/null 2>&1; then
    echo -e "  ${YELLOW}!${NC} toast non compile : gcc/make absents de l hote de build" >&2
else
    install -d "$ROOTFS/opt/GSM" "$ROOTFS/usr/local/bin"
    if ( cd "$ROOTFS/opt/GSM" \
         && wget -qO "${_GSM_VER}.tar.gz" "$_GSM_URL" \
         && tar xzf "${_GSM_VER}.tar.gz" \
         && cd "$_GSM_DIR" \
         && { make >/dev/null 2>&1 || true; } \
         && { [ -x bin/toast ] || make toast >/dev/null 2>&1 || true; } \
         && [ -x bin/toast ] ); then
        for _b in toast untoast tcat; do
            [ -e "$ROOTFS/opt/GSM/${_GSM_DIR}/bin/$_b" ] \
                && install -m755 "$ROOTFS/opt/GSM/${_GSM_DIR}/bin/$_b" "$ROOTFS/usr/local/bin/$_b"
        done
        echo -e "  ${GREEN}✓${NC} toast : ${CYAN}/usr/local/bin/toast${NC} (sources : /opt/GSM/${_GSM_DIR})"
    else
        echo -e "  ${YELLOW}!${NC} compilation de toast echouee (reseau ?) - ISO sans toast" >&2
    fi
fi

# ── Datadir QEMU : le lien que reclame la RELOCALISATION ──────────────────────
# [2026-08-28] Diagnostique en direct sur un banc lite (192.168.1.7), ou la
# sequence mourait sur :
#
#   [FAIL] Emulator serial PTY (QEMU (pid ...) s'est arrete avant d'allouer son PTY)
#   qemu-system-arm: could not read keymap file: 'en-us'
#
# Le bloc "Keymaps QEMU" plus bas copie bien les keymaps - dans
# /usr/local/share/qemu/keymaps. Or QEMU ne les y cherche JAMAIS, et le fichier
# etait present en trois exemplaires sur la machine pendant que QEMU jurait ne
# pas le trouver.
#
# La raison tient a get_relocated_path() : QEMU ne prend PAS son
# CONFIG_QEMU_FIRMWAREPATH tel quel. Il en deduit un chemin RELATIF a son
# bindir de compilation, puis l'applique au repertoire ou le binaire se trouve
# REELLEMENT. Ici :
#
#   compile avec   prefix=/opt/GSM/qemu-install   (donc bin/ et share/qemu/ y sont)
#   execute depuis /opt/GSM/qosmo/build/qemu-system-arm   (run.sh -> QEMU_BIN)
#   QEMU cherche   /opt/GSM/qosmo/share/qemu   <- n'existait pas
#
# Le prefix compile n'est alors plus jamais consulte. Mesure faite sur le banc,
# meme binaire, meme machine :
#
#   sans lien : "could not read keymap file: 'en-us'"   rc=1  (QEMU meurt)
#   avec lien : aucune erreur                           rc=124 (tue par timeout,
#                                                        donc il tournait)
#
# Le lien <exec_dir>/../share/qemu est le SEUL qui repare : l'autre candidat de
```

```
# QEMU, <exec_dir>/pc-bios, a ete teste sur le banc et laisse l'erreur intacte.
QINST="$ROOTFS/opt/GSM/qemu-install/share/qemu"
if [ -d "$QSRC" ] && [ -d "$QINST/keymaps" ]; then
    if [ -e "$QSRC/share/qemu/keymaps/en-us" ]; then
        echo -e "  ${GREEN}✓${NC} datadir QEMU : ${CYAN}/opt/GSM/qosmo/share/qemu${NC} deja resolu"
    else
        mkdir -p "$QSRC/share"
        rm -rf "$QSRC/share/qemu"
        ln -sfn /opt/GSM/qemu-install/share/qemu "$QSRC/share/qemu"
        echo -e "  ${GREEN}✓${NC} datadir QEMU : ${CYAN}/opt/GSM/qosmo/share/qemu${NC} ->
         ↪  /opt/GSM/qemu-install/share/qemu (keymap 'en-us' resolu)"
    fi
elif [ -d "$QSRC" ] && [ "$ISO_ROLE" != "interstp" ]; then
    echo -e "  ${YELLOW}!${NC} /opt/GSM/qemu-install/share/qemu/keymaps absent - QEMU mourra sur 'could not read
     ↪  keymap file'" >&2
fi

# Le binaire vient peut-etre DEJA de l'image : "docker cp $CID:/usr/local/bin/."
# (plus haut) copie /usr/local/bin/qemu-system-arm dans le rootfs, et c'est
# exactement celui que le conteneur utilise pour emuler le Calypso. Exiger en
# plus un build sur l'HOTE faisait echouer la construction sur une machine ou
# QEMU n'a jamais ete recompile - le cas courant - alors que l'ISO aurait ete
# parfaitement utilisable. On ne garde le caractere fatal que pour le vrai
# probleme : aucun binaire, ni sur l'hote, ni dans l'image.
#
# Le pc-bios n'est pas necessaire ici : dans l'image, ni /usr/local/share/qemu
# ni /usr/local/share/qemu-firmware n'existent, et la machine Calypso demarre
# sans fichier de firmware QEMU (elle charge sa propre ROM). Les recopier
# couterait 303 Mo dans une ISO qui tient en RAM.
ROOTFS_QEMU="$ROOTFS/usr/local/bin/qemu-system-arm"
if [ ! -x "$QEMU_BUILD_LOCAL/qemu-system-arm" ] \
   && [ ! -x "$ROOTFS_QEMU" ] && [ "$ISO_ROLE" != "interstp" ]; then
    echo -e "${RED}${BOLD}[5b/9] qemu-system-arm introuvable${NC}" >&2
    echo -e "  ${YELLOW}Ni build local : ${QEMU_BUILD_LOCAL}/qemu-system-arm${NC}" >&2
    echo -e "  ${YELLOW}Ni binaire venu de l'image : ${ROOTFS_QEMU}${NC}" >&2
    echo -e "  ${YELLOW}L'image d'operateur emule le Calypso : sans ce binaire elle n'a pas de MS.${NC}" >&2
    echo -e "  ${YELLOW}Trois issues : compiler qosmo, pointer OSMO_QEMU_BUILD sur un build${NC}" >&2
    echo -e "  ${YELLOW}existant, ou reconstruire l'image docker qui, elle, porte le binaire.${NC}" >&2
    exit 1
fi
if [ ! -x "$QEMU_BUILD_LOCAL/qemu-system-arm" ] && [ -x "$ROOTFS_QEMU" ]; then
    # strip de l hote : x86 seulement (le binutils de l hote ne lit pas l aarch64 ;
    # le binaire arm64 est garde tel quel, symboles compris).
    [ "${ISO_ARCH:-amd64}" = "$(dpkg --print-architecture)" ] && strip --strip-unneeded "$ROOTFS_QEMU" 2>/dev/null
     ↪  || true
    echo -e "  ${GREEN}✓${NC} qemu-system-arm repris de l'image ($(du -h "$ROOTFS_QEMU" | cut -f1)), pas de build
     ↪  hote necessaire"
elif [ -x "$QEMU_BUILD_LOCAL/qemu-system-arm" ]; then
    qpfx="$(sed -n 's/^prefix=//p' "$QEMU_BUILD_LOCAL/config-host.mak" 2>/dev/null)"
    qpfx="${qpfx:-/usr/local}"

    if DESTDIR="$ROOTFS" ninja -C "$QEMU_BUILD_LOCAL" install >/dev/null 2>&1; then
        echo -e "  ${GREEN}✓${NC} qemu installe dans ${ROOTFS}${qpfx} (ninja install, pas de sources)"
    else
        # repli : binaire + firmwares/keymaps strictement necessaires
        install -Dm755 "$QEMU_BUILD_LOCAL/qemu-system-arm" "$ROOTFS$qpfx/bin/qemu-system-arm"
        install -d "$ROOTFS$qpfx/share/qemu"
        cp -a "$QEMU_BUILD_LOCAL/pc-bios/." "$ROOTFS$qpfx/share/qemu/" 2>/dev/null || true
        echo -e "  ${GREEN}✓${NC} qemu-system-arm + pc-bios copies dans ${ROOTFS}${qpfx} (repli manuel)"
    fi
    strip --strip-unneeded "$ROOTFS$qpfx/bin/qemu-system-arm" 2>/dev/null || true
else
    # Seul le hub arrive ici : il ne fait que router du M3UA, il n'a pas de MS a
    # emuler. Pour l'operateur, le test ci-dessus a deja arrete la construction.
    echo -e "  ${CYAN}Role inter-STP : pas de QEMU (aucun MS a emuler)${NC}"
fi
```

```
# ── Lanceur C qosmo : ce que 40-qemu.sh appelle ──────────
# [2026-09-03] Chaque fork porte tools/qosmo-launch/qosmo-launch.c, compile dans
# SON dossier (make) et installe dans /usr/local/bin sous le nom du fork. Sans
# lui, run.sh retombe sur qemu-system-arm direct : l'ISO marche, mais sans les
# liens pty stables ni les options reseau/sockets. On construit sur l'hote (C
# pur, libc seule, aucun warning sous -Wall -Wextra) et on copie ; a defaut de
# source, on reprend le binaire deja installe sur l'hote. Jamais fatal.
if [ "$ISO_ROLE" != "interstp" ]; then
    for fork in qosmo; do
        lsrc=""
        for c in "/opt/GSM/$fork/tools/qosmo-launch" "$ROOTFS/opt/GSM/$fork/tools/qosmo-launch"; do
            [ -f "$c/qosmo-launch.c" ] && { lsrc="$c"; break; }
        done
        if [ "${ISO_ARCH:-amd64}" != "$(dpkg --print-architecture)" ]; then
            # Compilation croisee : ni gcc de l hote ni binaire de l hote, ce
            # serait du x86. Le lanceur est compile dans le chroot (82-arm-natif)
            # s il n est pas deja venu de l image.
            if [ -x "$ROOTFS/usr/local/bin/$fork" ]; then
               echo -e "  ${GREEN}✓${NC} lanceur ${CYAN}/usr/local/bin/$fork${NC} (deja dans l'image ${ISO_ARCH})"
            else
                echo -e "  ${CYAN}·${NC} lanceur $fork : compile dans le chroot ${ISO_ARCH} plus loin"
            fi
        elif [ -n "$lsrc" ] && command -v gcc >/dev/null 2>&1 \
           && make -s -C "$lsrc" ALIAS="$fork" TREE="/opt/GSM/$fork" "$fork" >/dev/null 2>&1 && [ -x "$lsrc/$fork"
            ↪  ]; then
            install -Dm755 "$lsrc/$fork" "$ROOTFS/usr/local/bin/$fork"
            echo -e "  ${GREEN}✓${NC} lanceur ${CYAN}/usr/local/bin/$fork${NC} (compile depuis $lsrc)"
        elif [ -x "/usr/local/bin/$fork" ]; then
            install -Dm755 "/usr/local/bin/$fork" "$ROOTFS/usr/local/bin/$fork"
            echo -e "  ${GREEN}✓${NC} lanceur ${CYAN}/usr/local/bin/$fork${NC} (repris de l'hote)"
        elif [ -x "$ROOTFS/usr/local/bin/$fork" ]; then
            echo -e "  ${GREEN}✓${NC} lanceur ${CYAN}/usr/local/bin/$fork${NC} (deja dans l'image)"
        else
            echo -e "  ${YELLOW}!${NC} lanceur $fork absent (ni source, ni binaire) : run.sh appellera
             ↪  qemu-system-arm directement" >&2
        fi
        # ── Console gdb en telnet (44-gdb-telnet.sh) : le panneau cmd.gdb ─────
        # tools/gdb-telnet.py sert `telnet localhost 44444` -> gdb-multiarch sur le
        # gdbstub ARM, cible en marche ; il source tools/cmd.gdb, GENERE par
        # tools/gdb_cmd.sh (68 commandes osmocom : dsp, sb, tasks, fake_sb,
        # trace_frames, help_osmo...). On le regenere dans l'arbre embarque pour
        # qu'il corresponde au gdb_cmd.sh qui part. gdb-multiarch et telnet sont
        # dans PKGS (role operateur), il n'y a rien d'autre a installer.
        if [ -f "$ROOTFS/opt/GSM/$fork/tools/gdb_cmd.sh" ]; then
            # GDB_CMD_HERE : le chemin que cmd.gdb aura DANS L'IMAGE, pas celui du
            # rootfs de build (sinon `source /var/tmp/iso-build-NNN/rootfs/...`).
            if (cd "$ROOTFS/opt/GSM/$fork/tools" && GDB_CMD_HERE="/opt/GSM/$fork/tools" bash ./gdb_cmd.sh
             ↪  >/dev/null 2>&1) \
               && [ -s "$ROOTFS/opt/GSM/$fork/tools/cmd.gdb" ]; then
                echo -e "  ${GREEN}✓${NC} gdb telnet ${CYAN}$fork${NC} : tools/cmd.gdb genere ($(grep -c '^define
                 ↪  ' "$ROOTFS/opt/GSM/$fork/tools/cmd.gdb") commandes), serveur 44-gdb-telnet.sh"
            else
                echo -e "  ${YELLOW}!${NC} $fork : tools/gdb_cmd.sh n'a pas produit cmd.gdb - la console telnet
                 ↪  marchera sans le panneau" >&2
            fi
        fi
    done
fi
# ── QEMU_BIN dans l'arbre : le lien, SEULEMENT si le binaire n'y est pas ──
# environnement/paths.env du depot qemu resout QEMU_BIN a
# $QEMU_TREE/build/qemu-system-arm. L'arbre embarque le porte deja (build/ part
# entier) : dans ce cas on ne touche a RIEN - un "ln -sf" par-dessus remplacerait
# le binaire compile par un lien, c'est-a-dire l'effacerait.
# Le lien ne sert qu'au cas contraire (arbre venu d'ailleurs, build/ absent) :
# sans lui, run.sh s'arrete des le premier module -
#     [FAIL] Prerequisite checks (dépendances introuvables : QEMU_BIN)
```

```
# - alors que le binaire est la, dans /usr/local/bin.
# osmo-qemu-link.service (etape [6/9]) applique la meme regle a chaque demarrage.
if [ -d "$QSRC" ] && [ ! -e "$QSRC/build/qemu-system-arm" ]; then
    qbin=""
    for c in "$ROOTFS/usr/local/bin/qemu-system-arm" \
             "$ROOTFS${qpfx:-/usr/local}/bin/qemu-system-arm"; do
        [ -x "$c" ] && { qbin="${c#$ROOTFS}"; break; }
    done
    if [ -n "$qbin" ]; then
        mkdir -p "$QSRC/build"
        ln -sfn "$qbin" "$QSRC/build/qemu-system-arm"
        echo -e "  ${GREEN}✓${NC} QEMU_BIN : ${CYAN}/opt/GSM/qosmo/build/qemu-system-arm${NC} ->
         ↪  ${CYAN}${qbin}${NC}"
    elif [ "$ISO_ROLE" != "interstp" ]; then
        echo -e "  ${YELLOW}!${NC} binaire QEMU introuvable dans le rootfs - QEMU_BIN restera non resolu" >&2
    fi
elif [ -e "$QSRC/build/qemu-system-arm" ]; then
    echo -e "  ${GREEN}✓${NC} QEMU_BIN : ${CYAN}/opt/GSM/qosmo/build/qemu-system-arm${NC} (binaire compile de
     ↪  l'arbre)"
fi

# ── Keymaps QEMU : 917 ko qui decident si la machine demarre ──────────────
# [2026-08-27] Le commentaire du bloc d'installation ci-dessus dit vrai pour le
# pc-bios - 303 Mo
# de ROMs (bios.bin, edk2, efi-*.rom) que la machine Calypso n'ouvre jamais,
# elle charge la sienne. Il est FAUX pour les keymaps, qui ne sont pas du
# firmware : l'interface graphique les lit a l'initialisation, machine Calypso
# comprise. Sans $prefix/share/qemu/keymaps, qemu-system-arm ecrit
#     qemu-system-arm: could not read keymap file: 'en-us'
# et s'arrete AVANT le premier cycle. Vu de start-native.sh, ca donne
#     [FAIL] Calypso emulator (QEMU) (started but never ready)
# c'est-a-dire une ISO sans MS - la panne exacte que le bloc precedent veut
# eviter. On copie donc les keymaps SEULES : 917 ko, pas 303 Mo.
if [ "$ISO_ROLE" != "interstp" ]; then
    if [ -d "$ROOTFS/usr/local/share/qemu/keymaps" ]; then
        echo -e "  ${GREEN}✓${NC} keymaps QEMU deja en place (${CYAN}/usr/local/share/qemu/keymaps${NC})"
    else
        qkm=""
        for c in "$QEMU_BUILD_LOCAL/pc-bios/keymaps" \
                 "$QSRC/pc-bios/keymaps" \
                 "$ROOTFS/opt/GSM/qemu-install/share/qemu/keymaps" \
                 "$ROOTFS/usr/share/qemu/keymaps" \
                 /usr/local/share/qemu/keymaps \
                 /usr/share/qemu/keymaps; do
            [ -d "$c" ] && { qkm="$c"; break; }
        done
        if [ -n "$qkm" ]; then
            mkdir -p "$ROOTFS/usr/local/share/qemu"
            cp -a "$qkm" "$ROOTFS/usr/local/share/qemu/"
            echo -e "  ${GREEN}✓${NC} keymaps QEMU ($(du -sh "$ROOTFS/usr/local/share/qemu/keymaps" | cut -f1))
             ↪  copiees depuis ${CYAN}${qkm}${NC}"
        else
            # Pas fatal : le binaire peut avoir ete configure avec un autre
            # prefixe, ou une version future ne plus les lire. Mais c'est la
            # premiere chose a regarder si QEMU "demarre puis s'arrete".
            echo -e "  ${YELLOW}!${NC} keymaps QEMU introuvables - si QEMU s'arrete au demarrage," >&2
            echo -e "  ${YELLOW}  cherchez \"could not read keymap file\" dans logs/qemu.log${NC}" >&2
        fi
    fi
fi

echo -e "${GREEN}[5c/9] Ajustements osmocom dans le rootfs...${NC}"
echo -e "${GREEN}[5d/9] Patch configs ISO...${NC}"

# LA MEME recette que sur $TEMP_CONFIG, rejouee sur le rootfs. Elle est
# idempotente : ce qui est deja juste ne bouge pas. On la rejoue quand meme
```

```
# parce que /etc/osmocom du rootfs vient de l'IMAGE docker (docker cp a
# l'etape 5), pas de $TEMP_CONFIG - l'image peut porter des fichiers que la
# substitution n'a pas traverses.
apply_native_post_patches "$ROOTFS/etc" "$ISO_OP_ID" "$ISO_N_MS" "$HOST_IP" \
    "${ISO_NODE:-1}" "${ISO_WAN_TMP:-/nonexistent}" "$SGSN_GTP_IP" "$HLR_IP"

if [ -f "$ROOTFS/etc/osmocom/run.sh" ]; then
    chmod +x "$ROOTFS/etc/osmocom/run.sh"
else
    # Garde-fou : sous set -e, un run.sh absent tuait la construction tout au
    # bout de l'etape 5. Le fichier vient de l'image, pas du depot : s'il
    # manque, c'est l'image qu'il faut regarder, pas une heure de build qu'il
    # faut perdre.
    echo -e "  ${YELLOW}!${NC} /etc/osmocom/run.sh absent de l'image ${CYAN}${ISO_SRC_IMAGE}${NC}"
fi

echo -e "  ${GREEN}✓${NC} retouches natives rejouees sur le rootfs (SGSN, MSC, sms-routing, run.sh)"
mkdir -p "$ROOTFS/usr/bin"
# ── /usr/local/bin AUSSI DANS /usr/bin, MAIS JAMAIS SUR UN BINAIRE DE DPKG ──
# [2026-09-04] Ce `cp -a` recopiait TOUT /usr/local/bin dans /usr/bin. Or
# /usr/local/bin contient le wrapper tcpdump (l'image l'apporte en 50-injection,
# 83-espace-ram.sh le repose) - et le wrapper, lui, appelle /usr/bin/tcpdump.
# Recopie par-dessus le vrai binaire, il s'appelait donc LUI-MEME, indefiniment.
# L'apt de l'etape 8 ne rattrapait rien : le paquet etant deja installe dans
# l'image, dpkg n'avait aucune raison de reecrire le fichier.
#
# Ce que ca donnait sur la machine demarree, et pourquoi ca ne ressemblait pas a
# un probleme de tcpdump : run_modules/75-gsmtap.sh attend son pcap, ne le voit
# jamais venir, et rend "[FAIL] GSMTAP capture (pcap)". run.sh sort alors en 1
# (nb_fail > 0) ; osmo-banc.service est un oneshot, donc systemd conclut a un
# service en echec et SIGKILL tout le cgroup - la pile entiere, montee et
# fonctionnelle, tuee par une capture reseau.
#
# On ne copie donc que ce qui n'appartient a AUCUN paquet. La liste des fichiers
# de dpkg se lit directement dans le rootfs : pas de chroot (qui ne marcherait
# pas en construction croisee arm64), pas de dpkg-query.
_ubin_dpkg="$WORK/usr-bin-dpkg.txt"
cat "$ROOTFS"/var/lib/dpkg/info/*.list 2>/dev/null \
    | grep '^/usr/bin/' | sort -u > "$_ubin_dpkg" || true
for _f in "$ROOTFS/usr/local/bin/"*; do
    [ -e "$_f" ] || continue
    _b="${_f##*/}"
    if grep -qxF "/usr/bin/$_b" "$_ubin_dpkg"; then
        echo -e "  ${YELLOW}!${NC} /usr/bin/$_b appartient a un paquet - laisse intact
        ↪  (${CYAN}/usr/local/bin/$_b${NC} le precede dans le PATH)"
        continue
    fi
    cp -a "$_f" "$ROOTFS/usr/bin/" 2>/dev/null || true
done

mkdir -p "$ROOTFS/root/.osmocom/bb"
if [ -f "$ROOTFS/opt/GSM/osmo-operator/mobile.cfg" ]; then
    cp "$ROOTFS/opt/GSM/osmo-operator/mobile.cfg" "$ROOTFS/root/.osmocom/bb/mobile.cfg"
elif [ -f "$ROOTFS/opt/GSM/osmo-operator/configs/mobile.cfg" ]; then
    cp "$ROOTFS/opt/GSM/osmo-operator/configs/mobile.cfg" "$ROOTFS/root/.osmocom/bb/mobile.cfg"
elif [ -f "$ROOTFS/etc/osmocom/mobile.cfg" ]; then
    cp "$ROOTFS/etc/osmocom/mobile.cfg" "$ROOTFS/root/.osmocom/bb/mobile.cfg"
fi

# ── PAS D'UTILISATEUR osmocom : ROOT, ET RIEN D'AUTRE ──────────────────────
# L'image portait un compte "osmocom" force a l'UID 0 (usermod -o -u 0). Un
# compte qui EST root sans le dire coute plus qu'il ne rapporte :
#   - GDM refuse toute session pour l'uid 0, il fallait donc neutraliser la
#     regle PAM "user != root" pour qu'un autologin sur ce compte aboutisse ;
#   - deux noms pour le meme uid donnent deux HOME (/home/osmocom et /root) et
#     donc deux .osmocom/bb : les mobiles ecrits dans l'un, lus dans l'autre ;
```

```
#   - "ls -l" affiche tantot root tantot osmocom pour un meme proprietaire,
#     selon l'ordre de /etc/passwd - de quoi chercher longtemps un probleme de
#     droits qui n'existe pas.
# Cette image tourne en root, assume : on SUPPRIME le compte et on rend les
# unites systemd a root. Les .service viennent des paquets Osmocom amont, qui
# posent "User=osmocom / Group=osmocom" ; sans compte, ils echouent au
# demarrage sur "Failed to determine user credentials" - un demon qui ne part
# pas, et rien dans son propre journal pour le dire.
sed -i -e 's/^User=osmocom$/User=root/' -e 's/^Group=osmocom$/Group=root/' \
       "$ROOTFS/lib/systemd/system"/osmo-*.service \
       "$ROOTFS/etc/systemd/system"/osmo-*.service 2>/dev/null || true

chroot "$ROOTFS" userdel -r osmocom 2>/dev/null || true
chroot "$ROOTFS" groupdel osmocom  2>/dev/null || true
rm -rf "$ROOTFS/home/osmocom"
# /var/lib/osmocom (bases HLR, etats GTP) et /var/log/osmocom appartenaient au
# compte supprime : sans ce chown ils gardent un UID orphelin, et l'ecriture
# echoue des le premier demarrage ("Unable to create file").
chown -R 0:0 "$ROOTFS/root/.osmocom" "$ROOTFS/var/lib/osmocom" \
             "$ROOTFS/var/log/osmocom" 2>/dev/null || true

# Le compte "osmocom" de l image Docker etait un alias d UID 0 : on le retire
# ici ; le VRAI compte osmocom, non privilegie, est recree en 8d.
echo -e "  ${GREEN}✓${NC} alias uid-0 osmocom retire (unites osmo-* rendues a root) + /usr/bin + mobile.cfg prets"


# Fin de module : `. fichier` rend le statut de sa DERNIERE commande, et
# build-iso.sh tourne sous set -e. Un module qui finirait par un test faux
# ("[ ... ] && { ...; }") arreterait tout, sans un mot. Toujours 0 ici.
true
```

3.201 /opt/GSM/osmo-operator/iso_modules/60-dashboard.sh

11983 octets, 189 lignes → 189 lignes

```
#!/bin/bash
# iso_modules/60-dashboard.sh - etape 6 : dashboard web
# Source par build-iso.sh, dans l ordre des numeros : meme shell, memes
# variables, memes fonctions. Ne s execute pas seul. `return` en tete de
# module = "rien a faire ici" (c est ainsi que --arm saute une etape).

# ── Etape 6 : Injection du dashboard web ──────────────────────────────
echo -e "${GREEN}[6/9] Dashboard web (git clone)...${NC}"
WEB="$ROOTFS/opt/GSM/osmo-egprs-web"
WEB_REPO="${OSMO_WEB_REPO:-https://github.com/bbaranoff/osmo-egprs-web.git}"
# main, PAS "test". La branche de travail du depot web partait dans toutes les
# ISO : une image gravee recevait ce qui n'etait pas encore relu, et deux ISO
# construites a deux semaines d'ecart n'embarquaient pas le meme dashboard sans
# qu'aucune option ne l'ait demande. OSMO_WEB_BRANCH=test reste possible, mais
# il faut le vouloir.
WEB_BRANCH="${OSMO_WEB_BRANCH:-main}"

mkdir -p "$WEB/web"
# Source AUTORITAIRE = le VRAI git bbaranoff/osmo-egprs-web (clone ci-dessous).
# La copie locale /opt/GSM/osmo-egprs-web n'est plus utilisee que comme override
# EXPLICITE : OSMO_WEB_LOCAL=/chemin ./build-iso.sh. Sinon -> git.
# Le patch natif plus bas est idempotent (skip si server.js est deja en mode natif).
LOCAL_WEB="${OSMO_WEB_LOCAL:-}"
if [ -n "$LOCAL_WEB" ] && [ -f "$LOCAL_WEB/server.js" ]; then
    cp "$LOCAL_WEB/server.js" "$WEB/server.js"
    [ -f "$LOCAL_WEB/package.json" ] && cp "$LOCAL_WEB/package.json" "$WEB/package.json"
    [ -d "$LOCAL_WEB/web" ]          && cp -r "$LOCAL_WEB/web/."     "$WEB/web/"
    [ -f "$LOCAL_WEB/start-web.sh" ] && cp "$LOCAL_WEB/start-web.sh" "$WEB/" && chmod +x "$WEB/start-web.sh"
    [ -f "$LOCAL_WEB/Dockerfile" ]   && cp "$LOCAL_WEB/Dockerfile"   "$WEB/Dockerfile"
```

```
    # Le depot suit les fichiers : c'est lui qui evite le reclone au demarrage.
    [ -d "$LOCAL_WEB/.git" ]         && cp -a "$LOCAL_WEB/.git"     "$WEB/"
    echo -e "  ${GREEN}✓${NC} osmo-egprs-web depuis source LOCALE ($LOCAL_WEB)"
else
    WEB_TMP="$WORK/osmo-egprs-web"
    GIT_TERMINAL_PROMPT=0 git clone --depth 1 -b "$WEB_BRANCH" "$WEB_REPO" "$WEB_TMP" 2>&1 | tail -2 || true
    # [2026-08-27] Le clone entier part dans l'image, .git COMPRIS. Avant, on ne
    # prelevait que quelques fichiers : l'ISO recevait un dossier sans depot, et
    # update.sh, faute de .git, ne pouvait qu'EFFACER et RECLONER a chaque
    # demarrage (wipe=1) - sans reseau, plus de dashboard du tout.
    # cp -a : les fichiers deja poses par un override local ne sont pas effaces,
    # ils sont recouverts par ceux du depot.
    [ -d "$WEB_TMP/.git" ] && cp -a "$WEB_TMP/." "$WEB/"
    # Layout REEL du repo : server.js / package.json / web/ / start-web.sh a la
    # RACINE (fallback sous server/ pour un ancien layout).
    if   [ -f "$WEB_TMP/server.js" ];        then cp "$WEB_TMP/server.js"        "$WEB/server.js"
    elif [ -f "$WEB_TMP/server/server.js" ]; then cp "$WEB_TMP/server/server.js" "$WEB/server.js"; fi
    if   [ -f "$WEB_TMP/package.json" ];        then cp "$WEB_TMP/package.json"        "$WEB/package.json"
    elif [ -f "$WEB_TMP/server/package.json" ]; then cp "$WEB_TMP/server/package.json" "$WEB/package.json"; fi
    [ -d "$WEB_TMP/web" ]          && cp -r "$WEB_TMP/web/."     "$WEB/web/"
    [ -f "$WEB_TMP/start-web.sh" ] && cp "$WEB_TMP/start-web.sh" "$WEB/" && chmod +x "$WEB/start-web.sh"
    [ -f "$WEB_TMP/Dockerfile" ]   && cp "$WEB_TMP/Dockerfile"   "$WEB/Dockerfile"
    if [ -f "$WEB/server.js" ]; then
        echo -e "  ${GREEN}✓${NC} osmo-egprs-web depuis le git ${CYAN}$WEB_REPO${NC} ($WEB_BRANCH)"
    else
        echo -e "  ${RED}✗ clone osmo-egprs-web sans server.js - dashboard incomplet${NC}"
    fi
fi

# ── LE TUTORIEL, SERVI PAR LE DASHBOARD ──────────────────────────────
# Ici et pas ailleurs : $WEB vient d etre peuple (clone ou source locale), et
# c est la seule racine statique que le dashboard expose. /usr/share garde un
# exemplaire pour le repli hors ligne, mais c est CELUI-CI que l icone ouvre -
# voir /usr/local/bin/osmo-tutorial.
if [ -f "$DIR/data/tutorial.html" ]; then
    mkdir -p "$WEB/web"
    cp -f "$DIR/data/tutorial.html" "$WEB/web/tutorial.html"
    echo -e "  ${GREEN}✓${NC} tutoriel servi par le dashboard : ${CYAN}/tutorial.html${NC}"
fi
# [2026-10-23] la page d aide (le « ? » du dock) aussi : Firefox est un snap et ne lit pas
# file:///usr/share, l icone ouvrait une page vide. Voir /usr/local/bin/osmo-info.
if [ -f "$DIR/configs/info.html" ]; then
    mkdir -p "$WEB/web"
    cp -f "$DIR/configs/info.html" "$WEB/web/info.html"
    echo -e "  ${GREEN}✓${NC} aide servie par le dashboard : ${CYAN}/info.html${NC}"
fi

# Patch server.js : mode natif (no-docker). VTY en telnet direct sur 127.0.0.1
# (ou ip netns exec) au lieu de docker exec. Idempotent ; n'echoue pas le build.
if [ -f "$WEB/server.js" ] && command -v python3 >/dev/null 2>&1; then
python3 - "$WEB/server.js" <<'PYEOF' || echo -e "  ${YELLOW}[web] patch natif non applique (server.js amont a
 ↪  change ?)${NC}"
import sys, re
p = sys.argv[1]
s = open(p, encoding='utf-8').read()
if 'const NATIVE' in s:
    print('  [web] server.js deja en mode natif - skip'); sys.exit(0)
HELPERS = """
// ── Native (no-docker) mode ──────────────────────────────
const NATIVE        = (process.env.OSMO_NATIVE !== '0');
const OP_IDS        = (process.env.OSMO_OP_IDS || '1').split(',')
                        .map(function(s){ return parseInt(s, 10); })
                        .filter(function(n){ return !isNaN(n); });
const NETNS_PREFIX  = process.env.OSMO_NETNS_PREFIX || '';
function vtyProc(container, port, ip, id) {
  if (NATIVE) {
```

```
    if (NETNS_PREFIX) return { bin: 'ip', args: ['netns','exec', NETNS_PREFIX + id, 'telnet', ip, String(port)] };
    return { bin: 'telnet', args: [ip, String(port)] };
  }
  return { bin: 'docker', args: ['exec','-i', container, 'telnet', ip, String(port)] };
}
function shCmd(container, id, inner) {
  if (NATIVE) {
    if (NETNS_PREFIX) return 'ip netns exec ' + NETNS_PREFIX + id + ' bash -c "' + inner + '"';
    return 'bash -c "' + inner + '"';
  }
  return 'docker exec ' + container + ' bash -c "' + inner + '"';
}
"""
n = [0]
def sub(pat, rep, flags=0):
    global s
    s, c = re.subn(pat, rep, s, flags=flags); n[0]+=c; return c
sub(r"(const VTY_RETRY_DELAY = 2000;)", r"\1\n" + HELPERS.replace('\\','\\\\'))
sub(r"(function discoverOperators\(\) \{)", r"\1\n  if (NATIVE) return Promise.resolve(OP_IDS.slice());")
sub(r"var proc = spawn\('docker', \[\s*'exec', '-i', container, 'telnet', targetIp, String\(port\)\s*\], \{
 ↪  stdio: \['pipe','pipe','pipe'\] \}\);",
    "var vc = vtyProc(container, port, targetIp, String(container).replace(PREFIX, ''));\n    var proc =
 ↪  spawn(vc.bin, vc.args, { stdio: ['pipe','pipe','pipe'] });", re.DOTALL)
sub(r"return execAsync\(\s*'docker inspect.*?\)\.then\(function\(running\) \{",
    "var runningProbe = NATIVE\n    ? execAsync(shCmd(container, id, 'ss -tln 2>/dev/null | grep -q :' +
 ↪  VTY_PORTS.bsc + ' && echo true || echo false'), 3000)\n    : execAsync('docker inspect -f
 ↪  \\'{{.State.Running}}\\' ' + container + ' 2>/dev/null', 3000);\n  return runningProbe.then(function(running)
 ↪  {", re.DOTALL)
sub(r"execAsync\(\s*'docker exec ' \+ container \+ ' bash -c \"ss -tlnp.*?', 3000\s*\)",
    "execAsync(\n        shCmd(container, id, 'ss -tlnp 2>/dev/null | grep :7890 | wc -l'), 3000\n      )",
 ↪  re.DOTALL)
sub(r"log\('VTY open: docker exec.*?\], \{ stdio: \['pipe','pipe','pipe'\] \}\);",
    "var vc = vtyProc(this.container, this.port, this.ip, this.opId);\n  log('VTY open: ' + vc.bin + ' ' +
 ↪  vc.args.join(' ') + ' (attempt ' + (this.retries + 1) + ')');\n\n  this.proc = spawn(vc.bin, vc.args, { stdio:
 ↪  ['pipe','pipe','pipe'] });", re.DOTALL)
open(p,'w',encoding='utf-8').write(s)
print('  [web] server.js patche mode natif (%d remplacements)' % n[0])
sys.exit(0 if n[0] >= 6 else 2)
PYEOF
fi

# ── LE DASHBOARD DOIT SE LEVER SEUL, ICI COMME SUR LE DISQUE ──────────────
# Tout etait deja dans l image - server.js, node, node_modules - SAUF les deux
# unites systemd. services/ n etait copie nulle part par ce script : l ISO
# arrivait donc avec un dashboard complet et rien pour le demarrer, et il
# fallait lancer install-web-service.sh a la main a chaque demarrage. C est
# exactement ce qu on a constate sur le banc.
#
# DEUX UNITES, ET PAS UNE :
#   osmo-egprs-web.service          le dashboard lui-meme.
#   osmo-egprs-web-install.service  un oneshot qui rejoue install-web-service.sh
#                                   AU BOOT. Il porte le certificat TLS, et
#                                   c est la seule place correcte pour lui : une
#                                   cle posee ici, au build, serait la MEME dans
#                                   toutes les ISO tirees de cette image -
#                                   n importe qui pourrait se faire passer pour
#                                   la console. Genere au boot, il porte le nom
#                                   et les adresses REELS de la machine, ce qui
#                                   vaut aussi pour le systeme installe : le
#                                   disque recoit les unites avec le squashfs et
#                                   fabrique SON propre certificat au premier
#                                   demarrage, different de celui de la cle.
#
# On active l ONESHOT, pas le service : le script se termine par un
# `systemctl restart osmo-egprs-web` et l unite le dit - l ordonner avant le
# service creerait un cycle. Le dashboard est tire par lui.
```

```
_SVC_SRC="$DIR/services"
# [2026-08-31] SEUL LE SERVICE DU DASHBOARD EST POSE. Le oneshot d installation
# (osmo-egprs-web-install.service) n est plus deploye : il rejouait au premier
# demarrage un script qui TELECHARGE node s il manque, donc un boot qui exige
# Internet. L installation est faite au BUILD (Dockerfile, RUN bash
# install-web-service.sh) : l image arrive complete, le boot ne fait que lancer.
if [ -f "$_SVC_SRC/osmo-egprs-web.service" ]; then
    install -d "$ROOTFS/etc/systemd/system/multi-user.target.wants"
    cp -f "$_SVC_SRC/osmo-egprs-web.service"         "$ROOTFS/etc/systemd/system/"
    # Le depot web embarque sa propre copie du unit, qui a deja DIVERGE de
    # celle-ci (cf. Dockerfile) : on impose celle du depot operateur, une seule
    # verite.
    [ -d "$ROOTFS/opt/GSM/osmo-egprs-web" ] && \
        cp -f "$_SVC_SRC/osmo-egprs-web.service" \
              "$ROOTFS/opt/GSM/osmo-egprs-web/osmo-egprs-web.service"
    ln -sf /etc/systemd/system/osmo-egprs-web.service \
           "$ROOTFS/etc/systemd/system/multi-user.target.wants/osmo-egprs-web.service"
    echo -e "  ${GREEN}✓${NC} dashboard : unite ${CYAN}osmo-egprs-web${NC} posee et activee au boot"
else
    echo -e "  ${RED}✗ services/osmo-egprs-web*.service introuvables - le dashboard ne demarrerait pas seul${NC}"
     ↪  >&2
    exit 1
fi

# /usr/local/bin/node pointait sur /opt/node/bin/node, absent de l image : un
# lien mort AVANT /usr/bin dans le PATH. `node` marche par chance, parce que le
# shell continue son parcours ; un script qui teste `-x /usr/local/bin/node`,
# lui, se trompe. On ne garde le lien que s il mene quelque part.
if [ -L "$ROOTFS/usr/local/bin/node" ] && [ ! -e "$ROOTFS/usr/local/bin/node" ]; then
    rm -f "$ROOTFS/usr/local/bin/node" "$ROOTFS/usr/local/bin/npm" "$ROOTFS/usr/local/bin/npx"
    echo -e "  ${GREEN}✓${NC} liens morts /usr/local/bin/node,npm,npx retires (node reste en /usr/bin)"
fi


# Fin de module : `. fichier` rend le statut de sa DERNIERE commande, et
# build-iso.sh tourne sous set -e. Un module qui finirait par un test faux
# ("[ ... ] && { ...; }") arreterait tout, sans un mot. Toujours 0 ici.
true
```

3.202 /opt/GSM/osmo-operator/iso_modules/70-scripts.sh

13653 octets, 216 lignes → 216 lignes

```
#!/bin/bash
# iso_modules/70-scripts.sh - etape 7/7b : scripts projet, WAN, NetworkManager
# Source par build-iso.sh, dans l ordre des numeros : meme shell, memes
# variables, memes fonctions. Ne s execute pas seul. `return` en tete de
# module = "rien a faire ici" (c est ainsi que --arm saute une etape).

# ── Etape 7 : Injection des scripts projet et installation du lanceur start-native.sh ──
echo -e "${GREEN}[7/9] Scripts projet et adaptation ISO...${NC}"
# ── UN SEUL ARBRE DU DEPOT : /opt/GSM/osmo-operator ──────────────────────────
# Ici vivait la fabrication d'un SECOND arbre, /opt/GSM/osmo-operator : une copie
# PARTIELLE du depot (une liste de fichiers nommes un a un, plus sept
# repertoires), sans .git, figee a la construction. L'ISO partait donc avec
# deux osmo-operator :
#
#   /opt/GSM/osmo-operator   l'arbre COMPLET, avec son .git, mis a jour a
#                         l'etape [5a/9] et par "osmo-update" ensuite ;
#   /opt/GSM/osmo-operator       une copie partielle que plus rien ne mettait a jour.
#
# Et c'est le second que visaient les liens osmo-start-direct / osmo-start-lab,
# le message de login, l'alias osmo-lab et l'unite du hub SS7. Autrement dit :
# on mettait a jour un arbre, on en executait un autre. Tout ce qui a ete ajoute
```

```
# au depot depuis la derniere construction - un module, un script network/, une
# option - existait sur la machine et restait sans effet, parce que le lanceur
# lance n'etait pas celui qu'on venait de corriger.
#
# Le filet que la copie apportait - "un lanceur present meme sans reseau" - est
# conserve, mais AU MEME ENDROIT : si l'arbre complet n'a pas pu etre recupere,
# on le remplit depuis le depot de construction, et il n'y a toujours qu'un
# seul chemin.
P="$ROOTFS/opt/GSM/osmo-operator"
if [ ! -x "$P/start-native.sh" ]; then
    echo -e "  ${YELLOW}!${NC} /opt/GSM/osmo-operator sans lanceur (image perimee, clone impossible)"
    echo -e "    -> remplissage depuis le depot de construction ${CYAN}${DIR}${NC}"
    mkdir -p "$P"
    # --exclude .git : on ne fabrique pas un faux depot. S'il en manquait un,
    # c'est que le reseau a manque ; osmo-update le reconstituera.
    tar -C "$DIR" --exclude=.git --exclude='*.iso' -cf - . | tar -C "$P" -xf -
    find "$P" -name "*.sh" -exec chmod +x {} \;
fi
if [ ! -x "$P/start-native.sh" ]; then
    echo -e "  ${RED}✗${NC} start-native.sh introuvable - l'ISO n'aura pas de lanceur" >&2
    exit 1
fi
ln -sf /opt/GSM/osmo-operator/start-native.sh "$ROOTFS/usr/local/bin/osmo-start-native" 2>/dev/null || true
ln -sf /opt/GSM/osmo-operator/start-native.sh "$ROOTFS/usr/local/bin/osmo-start-direct" 2>/dev/null || true   #
 ↪  ancien nom
# [2026-09-25] Le fond par defaut (80-chroot.sh le lit dans CET arbre, qui vient
# du clone GitHub) : pose depuis le depot de construction, pour qu une image non
# encore poussee parte quand meme avec son fond.
[ -f "$DIR/configs/gsm-lab-wallpaper.jpg" ] \
    && install -Dm644 "$DIR/configs/gsm-lab-wallpaper.jpg" "$P/configs/gsm-lab-wallpaper.jpg"
# [2026-10-11] Le fond nu va avec (meme scene que l encart, cf. tools/paysage-render.py).
[ -f "$DIR/configs/gsm-lab-fond.jpg" ] \
    && install -Dm644 "$DIR/configs/gsm-lab-fond.jpg" "$P/configs/gsm-lab-fond.jpg"
# [2026-10-12] Le fond en DEUX ETATS (tools/osmo-fond.py) : la ville seule au
# repos (gsm-lab-ville.jpg, le defaut de schema), la ville avec l antenne quand
# un banc tourne (gsm-lab-antenne.jpg) - fondu de l un a l autre.
for _bg in gsm-lab-ville.jpg gsm-lab-antenne.jpg; do
    [ -f "$DIR/configs/$_bg" ] && install -Dm644 "$DIR/configs/$_bg" "$P/configs/$_bg"
done
unset _bg
ln -sf /opt/GSM/osmo-operator/tools/osmo-fond.py "$ROOTFS/usr/local/bin/osmo-fond" 2>/dev/null || true
# osmo-op : quel operateur l encart et le Conky regardent (les fleches, en
# ligne de commande). C est aussi ce qu appelle le raccourci clavier pose par
# iso_modules/80-chroot.sh - Ctrl+AltGr+Droite / Ctrl+AltGr+Gauche.
ln -sf /opt/GSM/osmo-operator/tools/osmo-op.sh "$ROOTFS/usr/local/bin/osmo-op" 2>/dev/null || true
ln -sf /opt/GSM/osmo-operator/tools/osmo-drivers.sh "$ROOTFS/usr/local/bin/osmo-drivers" 2>/dev/null || true
ln -sf /opt/GSM/osmo-operator/tools/overlay-draw.py "$ROOTFS/usr/local/bin/overlay-draw" 2>/dev/null || true
# [2026-10-24] La fenetre du C123 emule (ecran + clavier de qosmo) dans le PATH :
# l alias c123 de /etc/profile.d/98-osmo-banc-aliases.sh ne vit que dans un shell
# de connexion ; le lien sert partout ailleurs (Alt+F2, sudo, scripts).
ln -sf /opt/GSM/osmo-operator/tools/osmo-c123-gtk.py "$ROOTFS/usr/local/bin/c123" 2>/dev/null || true
ln -sf /opt/GSM/osmo-operator/tools/osmo-c123-gtk.py "$ROOTFS/usr/local/bin/osmo-c123" 2>/dev/null || true
# [2026-10-24] Les alias du banc (banc-rssi, banc-trx, banc-mobile, c123-1...) : le fichier dans
# profile.d pour les shells de connexion, et les memes noms en vraies commandes (tools/osmo-banc-cmd.sh).
install -Dm644 "$P/scripts/osmo-banc-aliases.sh" "$ROOTFS/etc/profile.d/98-osmo-banc-aliases.sh"
bash "$P/tools/osmo-banc-cmd.sh" --install "$ROOTFS/usr/local/bin" >/dev/null
# [2026-10-24] /root/gnuarm/install/bin dans le PATH des qu il existe (supplement --gnuarm) :
# le fichier est pose ici, il ne fait rien tant que la chaine est absente.
install -Dm644 "$P/scripts/osmo-gnuarm-profile.sh" "$ROOTFS/etc/profile.d/97-osmo-gnuarm.sh"
# [2026-10-24] WireGuard du WAN (wiki/WAN.md) : « wg 1 » (hub 10.99.0.1), « wg N ... » (10.99.0.N),
# fonction de profile.d qui laisse passer le vrai wg ; osmo-wg pour sudo et les scripts. Les cles
# sont generees au premier « wg N » sur la machine, JAMAIS dans l image (elles seraient partagees).
install -Dm644 "$P/scripts/osmo-wg-profile.sh" "$ROOTFS/etc/profile.d/97-osmo-wg.sh"
ln -sf /opt/GSM/osmo-operator/tools/osmo-wg.sh "$ROOTFS/usr/local/bin/osmo-wg" 2>/dev/null || true
# (osmo-start-lab -> start-docker.sh retire le 2026-09-02 : start-docker.sh est le lanceur
#  Docker, et cette image n a pas Docker.)
```

```
if [ -f "$DIR/launch/osmo-launch.sh" ]; then
    cp "$DIR/launch/osmo-launch.sh" "$ROOTFS/opt/osmo-launch.sh" && chmod +x "$ROOTFS/opt/osmo-launch.sh"
    ln -sf /opt/osmo-launch.sh "$ROOTFS/usr/local/bin/osmo-launch"
fi
echo -e "  ${GREEN}✓${NC} lanceurs -> ${CYAN}/opt/GSM/osmo-operator${NC} (arbre unique, avec .git)"

# [2026-09-09] /etc/osmocom/gapk-start.sh vient de l image Docker (COPY scripts/.)
# et lib/audio.sh le PREFERE a scripts/gapk-start.sh. Une image en cache, batie
# avant le correctif du superviseur (emetteur RTP en doublon vers le BTS : le
# « 600 qu on entend mal »), embarquerait l ancien. On aligne la copie du rootfs
# sur le depot, toujours - c est idempotent et c est le depot qui fait foi.
if [ -f "$P/scripts/gapk-start.sh" ]; then
    install -d "$ROOTFS/etc/osmocom"
    install -m 755 "$P/scripts/gapk-start.sh" "$ROOTFS/etc/osmocom/gapk-start.sh"
    echo -e "  ${GREEN}✓${NC} /etc/osmocom/gapk-start.sh aligne sur scripts/ (superviseur gapk sans emetteur en
     ↪  doublon)"
fi

# ── LA VOIX DANS L IMAGE : io-tch-format ET LE TAMPON DU MOBILE ──────────────
# [2026-09-09] MEME CORRECTIF QUE update.sh, applique ici au ROOTFS - et par
# le MEME script, scripts/voix-forks.sh, qui porte le pourquoi en entier. Ces deux reglages ne vivent pas dans
 ↪  osmo-operator mais dans
# qosmo (run_modules, cfgs), que 50-injection-image.sh recopie tels quels
# depuis l image de reference : une image batie sur une machine dont les forks
# n ont pas le correctif l embarquerait sans, et l ISO sortirait avec la voix
# robotisee sans qu aucun compteur ne bronche.
#
#   io-tch-format rtp : sans lui, le mobile lit et ecrit ses trames TCH dans
#   la disposition Texas Instruments des vrais Calypso, alors que pont.py les
#   code avec libosmocoding, en disposition RTP/RFC3551. C est le gabarit
#   cfgs/mobile_group1.cfg qui compte : run_modules/20-mobile-cfg.sh le
#   RECOPIE par-dessus ~/.osmocom/bb/ a chaque demarrage.
#
#   CALYPSO_PULSE_LATENCY_MSEC : le mobile ecrit une trame GSM (20 ms) par
#   snd_pcm_writei ; un tampon trop court fait rendre -EPIPE a chaque
#   ecriture, et pq_alsa.c y repond par un snd_pcm_prepare() muet qui demonte
#   et remonte le flux PulseAudio - 50 fois par seconde pendant un appel.
#   Mesure du 09/09 : 80 ms -> 16 remontages en 10 s, 200 -> 0. On pose 320.
if [ -x "$P/scripts/voix-forks.sh" ]; then
    # [2026-09-14] PAS DE COULEUR DANS UN REMPLACEMENT sed. Cette ligne etait
    #     ... | sed "s|^  \[voix\]|  ${GREEN}\xe2\x9c\x93${NC}|"
    # or GREEN vaut '\033[0;32m' - une chaine faite pour `echo -e`, pas pour
    # sed : dans le remplacement, GNU sed lit \0 comme « toute la chaine
    # trouvee » (synonyme de &). Chaque ligne ressortait donc
    #     [voix]33[0;32m✓  [voix]33[0m /opt/GSM/... : io-tch-format rtp repose
    # - le [voix] reinjecte deux fois et l echappement en clair. On relit les
    # lignes et on les reecrit avec echo -e, comme partout ailleurs ici.
    "$P/scripts/voix-forks.sh" "$ROOTFS" | while IFS= read -r _l; do
        case "$_l" in
            "  [voix]"*) echo -e "  ${GREEN}✓${NC}${_l#"  [voix]"}" ;;
            *)           echo "$_l" ;;
        esac
    done
fi

# ── WAN : table des noeuds figee dans l'image ────────────────────────────────
if [ "$ISO_WAN" = "1" ]; then
    echo -e "${GREEN}[7b/9] WAN - table des noeuds embarquee...${NC}"
    # shellcheck source=network/wan-nodes.sh
    . "$DIR/network/wan-nodes.sh"
    WAN_OPS="$ISO_WAN_OPS"
    if [ -n "$ISO_WAN_NODES" ]; then
        wan_nodes_parse "$ISO_WAN_NODES" || exit 1
        WAN_NODE_ID="${ISO_WAN_ID:-0}"
    else
        # Construction interactive : memes questions que ./launch/start-docker.sh --wan.
```

```
        # Le numero du noeud demande ici n'est qu'un defaut : chaque machine
        # qui demarre l'ISO se re-reconnait a son IP.
        WAN_NODE_ID="${ISO_WAN_ID:-0}"
        wan_nodes_prompt || exit 1
    fi
    wan_nodes_validate || exit 1
    WAN_AUTO=1 WAN_CONF_FILE="$ROOTFS/etc/osmo-wan.conf" wan_nodes_save
    wan_nodes_summary
    echo -e "  ${GREEN}✓${NC} /etc/osmo-wan.conf fige dans l'ISO (WAN_AUTO=1)"
    echo -e "  ${CYAN}Au boot :${NC} start-native.sh applique le WAN tout seul ;"
    echo -e "  ${CYAN}sans --wan a la construction, l'ISO n'a AUCUN WAN.${NC}"
else
    echo -e "  ${CYAN}[7b/9] WAN non embarque (--wan absent) - ISO autonome${NC}"
fi

# ── Etape 8 : (SUPPRIME) - ISO NATIF, plus de Docker au runtime ──────────
# L'ancien load-osmocom-image.service chargeait osmocom-run.tar.gz via 'docker
# load' au boot ; son ExecStartPre 'while ! docker info' bloquait indefiniment la
# file systemd en natif (docker jamais up) → boot fige. Le lab tourne desormais
# en natif (start-native.sh) : pas d'image Docker a charger, pas de ce service.

# ── Etape 9 : Configuration chroot (paquets) ─────────────────────────────
# ── Ce que le chroot ne peut pas ecrire lui-meme ─────────────────────────
# Le script du chroot est passe a "bash -c" en QUOTES SIMPLES : rien n'y est
# substitue a l'ecriture, ce qui est voulu, mais une seule apostrophe dans le
# corps referme la chaine et tout ce qui suit change de sens. Les fichiers qui
# en contiennent - un heredoc quote, une commande shell imbriquee - s'ecrivent
# donc ICI, dans le rootfs, ou le quoting est normal. Le chroot ne fait plus que
# les activer.

# NetworkManager pilote le bureau ; ce qui appartient au coeur paquet ne lui
# appartient pas. Sans cette regle, NM reprend apn0 ou un tun du GGSN et coupe
# la session de donnees d'un abonne parce qu'il l'a jugee "non configuree".
mkdir -p "$ROOTFS/etc/NetworkManager/conf.d"
cat > "$ROOTFS/etc/NetworkManager/conf.d/10-osmo-networkd.conf" <<'NMCONF'
# Ecrit par build-iso.sh. NetworkManager gere les cartes physiques et le
# bureau ; les interfaces du coeur paquet restent a systemd-networkd et aux
# scripts du banc.
[main]
plugins=keyfile

[keyfile]
unmanaged-devices=interface-name:apn*;interface-name:tun*;interface-name:veth*;interface-name:docker*;interface-
 ↪  name:br-*;interface-name:osmo*
NMCONF

# ── FIREFOX : PLUS RIEN A FAIRE ICI ──────────────────────────────────────
# [2026-09-04] Ce module posait /usr/local/sbin/osmo-firefox-snap et son unite
# de premier demarrage : une centaine de lignes pour installer Firefox par snap,
# hors ligne depuis /var/lib/osmo-snaps puis depuis le magasin. Le navigateur
# vient desormais du .deb de Mozilla, installe dans le chroot par
# 80-chroot.sh - donc PRESENT DANS L IMAGE, et tenu a jour par apt sur le
# systeme installe.
#
# Ce que le snap coutait, et pourquoi on ne le regrette pas : l unite restait en
# "activating" des minutes au premier boot (1,5 Go a deballer, snapd qui refuse
# tant qu un changement est en cours), et quand elle echouait - medium lent, pas
# de reseau - le banc demarrait SANS navigateur, donc sans tableau de bord. Le
# bac a sable, lui, refusait le socket PulseAudio et les file:// hors /home, ce
# qui a coute deux enquetes entieres (micro muet, tutoriel "introuvable").
# Le .deb n a ni bac a sable, ni service d installation, ni delai au boot.


# Fin de module : `. fichier` rend le statut de sa DERNIERE commande, et
# build-iso.sh tourne sous set -e. Un module qui finirait par un test faux
# ("[ ... ] && { ...; }") arreterait tout, sans un mot. Toujours 0 ici.
```

```
true
```

3.203 /opt/GSM/osmo-operator/iso_modules/71-debs-banc.sh

2508 octets, 42 lignes → 42 lignes

```
#!/bin/bash
# iso_modules/71-debs-banc.sh - etape 7c : .deb du banc embarques
# Source par build-iso.sh, dans l ordre des numeros : meme shell, memes
# variables, memes fonctions. Ne s execute pas seul. `return` en tete de
# module = "rien a faire ici" (c est ainsi que --arm saute une etape).

# ── Etape 7c : les paquets .deb du banc, EMBARQUES dans l image ──────────
# packaging/build-debs.sh fabrique un .deb par composant (osmo-operator, pont,
# qemu-calypso, calypso-firmware) depuis les arbres que cette ISO embarque de
# toute facon. On les range dans /var/cache/osmo-debs : une machine installee
# depuis l ISO, ou n importe quelle autre, peut alors faire
#     dpkg -i /var/cache/osmo-debs/*.deb
# et obtenir le banc sans Docker ni git. L ISO elle-meme continue de tourner
# sur les arbres complets (le depot et qosmo avec leur .git : c est un
# atelier) - les paquets ne les remplacent pas, ils voyagent avec.
# Non fatal : sans dpkg-deb ou sans qemu construit, l image sort sans eux.
# --arm : build-debs.sh compile le lanceur C et lit les binaires par ldd - sur
# l hote ce serait du x86 etiquete arm64. Il tourne DANS le chroot, apres l apt
# de l etape 8 (82-arm-natif).
if [ "${ISO_WITH_DEBS:-1}" = "0" ]; then
    echo -e "  ${CYAN}·${NC} --without-debs : aucun paquet .deb embarque dans l'image"
elif [ "${ISO_ARCH:-amd64}" != "$(dpkg --print-architecture)" ]; then
    echo -e "  ${CYAN}·${NC} paquets .deb du banc : fabriques dans le chroot ${ISO_ARCH} plus loin"
elif [ -x "$DIR/packaging/build-debs.sh" ] && command -v dpkg-deb >/dev/null 2>&1; then
    echo -e "${GREEN}[7c/9] Paquets .deb du banc...${NC}"
    _DEBS="$WORK/debs"
    if OSMO_OPERATOR_SRC="$DIR" QOSMO_SRC="$ROOTFS/opt/GSM/qosmo" \
       FIRMWARE_SRC="$ROOTFS/opt/GSM/firmware" \
       "$DIR/packaging/build-debs.sh" --out "$_DEBS" >"$WORK/build-debs.log" 2>&1; then
        install -d "$ROOTFS/var/cache/osmo-debs"
        cp -f "$_DEBS"/*.deb "$ROOTFS/var/cache/osmo-debs/"
        echo -e "  ${GREEN}✓${NC} $(ls "$_DEBS"/*.deb | wc -l) paquets dans ${CYAN}/var/cache/osmo-debs${NC} ($(du
         ↪  -sh "$_DEBS" | cut -f1))"
    else
        echo -e "  ${YELLOW}!${NC} paquets .deb non construits (voir $WORK/build-debs.log) - l image sort sans eux"
    fi
fi


# Fin de module : `. fichier` rend le statut de sa DERNIERE commande, et
# build-iso.sh tourne sous set -e. Un module qui finirait par un test faux
# ("[ ... ] && { ...; }") arreterait tout, sans un mot. Toujours 0 ici.
true
```

3.204 /opt/GSM/osmo-operator/iso_modules/80-chroot.sh

72688 octets, 1131 lignes → 1131 lignes

```
#!/bin/bash
# iso_modules/80-chroot.sh - etape 8 : configuration du rootfs dans le chroot (apt, noyau, bureau)
# Source par build-iso.sh, dans l ordre des numeros : meme shell, memes
# variables, memes fonctions. Ne s execute pas seul. `return` en tete de
# module = "rien a faire ici" (c est ainsi que --arm saute une etape).

echo -e "${GREEN}[8/9] Configuration chroot...${NC}"
mount --bind /proc "$ROOTFS/proc"; mount --bind /sys "$ROOTFS/sys"
```

```
# ── LE CACHE APT DU CHROOT EST PERSISTANT ──────────────────────────────────
# [2026-09-04] Chaque passe retelechargeait ~700 paquets (le hub, la normale,
# et toute reconstruction). Le repertoire ou apt (et apt-fast, DLDIR) range ses
# .deb est un bind du cache de l hote, ISO_DEB_CACHE/apt-archives : telecharge
# une fois, reutilise par toutes les passes et les builds suivants. Il est
# demonte AVANT le squashfs (les .deb ne partent pas dans l ISO), et le
# chroot ne fait plus "apt-get clean" quand il est monte - ce serait vider le
# cache de l hote. --no-cache : pas de bind, comportement d avant.
ISO_APT_CACHE_BOUND=0
if [ -n "$ISO_DEB_CACHE" ] && [ -z "$NO_CACHE" ]; then
    mkdir -p "$ISO_DEB_CACHE/apt-archives/partial" "$ROOTFS/var/cache/apt/archives"
    if mount --bind "$ISO_DEB_CACHE/apt-archives" "$ROOTFS/var/cache/apt/archives"; then
        ISO_APT_CACHE_BOUND=1
        echo -e "  ${GREEN}cache apt du chroot : $ISO_DEB_CACHE/apt-archives ($(du -sh
        ↪  "$ISO_DEB_CACHE/apt-archives" | cut -f1), $(find "$ISO_DEB_CACHE/apt-archives" -maxdepth 1 -name
        ↪  '*.deb' | wc -l) paquets)${NC}"
    fi
fi
mount --bind /dev "$ROOTFS/dev";   mount --bind /dev/pts "$ROOTFS/dev/pts" 2>/dev/null||true
# Le resolv.conf de l HOTE, et seulement pour la duree du chroot : c est lui qui
# donne un DNS a apt pendant la construction. Il est REMPLACE a la fin
# (81-cloture-systeme.sh) - voir le bloc DNS la-bas pour ce qu il cassait quand
# il restait.
cp /etc/resolv.conf "$ROOTFS/etc/resolv.conf" 2>/dev/null||true

# L installeur apt-fast, le meme que celui du Dockerfile, dans le rootfs.
install -m755 "$DIR/packaging/apt-fast-install.sh" "$ROOTFS/usr/local/sbin/apt-fast-install"

# ISO_ROLE passe par l environnement : le script est en quotes simples, rien n y
# est substitue a l ecriture - c est voulu (aucune surprise d expansion), donc la
# seule facon de lui dire quelle image on construit est de le lui passer.
# LC_ALL=C.UTF-8 : la locale de l hote (fr_FR.UTF-8) fuyait dans le chroot, ou
# elle n est pas encore generee ; perl (dpkg-preconfigure, debconf) le disait a
# chaque paquet - "Setting locale failed" - et retombait sur C. On le fait
# d emblee, sans bruit. C.UTF-8 existe toujours dans la libc, sans locale-gen.
chroot "$ROOTFS" env PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin \
                  LC_ALL=C.UTF-8 LANG=C.UTF-8 LANGUAGE= \
                  ISO_ROLE="$ISO_ROLE" ISO_LITE="$ISO_LITE" ISO_APT_CACHE_BOUND="$ISO_APT_CACHE_BOUND" \
                  ISO_ARCH="$ISO_ARCH" ISO_MIRROR="$ISO_MIRROR" \
                  ISO_DESKTOP="$ISO_DESKTOP" OSMO_ISO_KB="$OSMO_ISO_KB" \
                  OSMO_ISO_SOFTSIM="${OSMO_ISO_SOFTSIM:-1}" bash -c '
set -e; export DEBIAN_FRONTEND=noninteractive
export DPKG_OPTIONS="--force-confold --force-confdef"

# ── apt/dpkg rapides ──────────────────────────────────────────────────────
# Ce rootfs est jetable : il est fabrique, empaquete en squashfs, puis efface.
# Les garanties de durabilite que dpkg paie a chaque fichier - un fsync par
# fichier deballe - n ont donc aucune valeur ici, et elles dominent le temps de
# construction. force-unsafe-io les coupe : c est le reglage qu utilisent les
# images Docker officielles, pour la meme raison.
#
# Le reste ne joue pas sur la durabilite mais sur ce qui est TELECHARGE :
#   Languages=none    supprime les traductions de descriptions (inutiles ici)
#   Pipeline-Depth    plusieurs requetes en vol au lieu d une a la fois
#   Retries           un miroir qui bronche ne fait plus echouer la construction
#                     entiere - ce chroot tourne sous set -e
mkdir -p /etc/dpkg/dpkg.cfg.d /etc/apt/apt.conf.d
echo "force-unsafe-io" > /etc/dpkg/dpkg.cfg.d/02-unsafe-io
# Les reglages de telechargement (Languages, Retries, Pipeline, Use-Pty) sont
# poses par apt-fast-install, plus bas, dans /etc/apt/apt.conf.d/90osmo-operator
# - les memes que dans l image docker et sur l hote.

APT_OPTS="-o Dpkg::Options::=--force-confold -o Dpkg::Options::=--force-confdef"

# Preseed debconf
echo "wireshark-common wireshark-common/install-setuid boolean true" | debconf-set-selections
```

```
echo "keyboard-configuration keyboard-configuration/layoutcode string us" | debconf-set-selections
echo "keyboard-configuration keyboard-configuration/modelcode string pc105" | debconf-set-selections
echo "console-setup console-setup/charmap47 select UTF-8" | debconf-set-selections

# ${ISO_SUITE} vient de l environnement du chroot (export plus haut) ; le
# heredoc est NON quote pour que la substitution ait lieu ici.
_S="${ISO_SUITE:-noble}"
# Le miroir suit l architecture (arm64 : ports.ubuntu.com). restricted en plus
# sur arm64 : linux-firmware-raspi (start4.elf, le firmware de demarrage du Pi)
# y est range.
_M="${ISO_MIRROR:-http://archive.ubuntu.com/ubuntu}"
_C="main universe multiverse"
[ "${ISO_ARCH:-amd64}" = "arm64" ] && _C="$_C restricted"
# ── GIT : plus de forcage HTTP/1.1 ──────────────────────────────────────
# [2026-09-03] RETIRE, ici comme dans le Dockerfile et update.sh. Le reglage
# http.version=HTTP/1.1 datait d un incident de reseau ("expected flush after
# ref listing") ; il ralentissait tous les clones de l image. Si une machine
# retombe dessus, c est un reglage LOCAL a poser sur elle :
#     git config --global http.version HTTP/1.1
git config --system --unset http.version 2>/dev/null || true

cat > /etc/apt/sources.list <<SOURCES
deb $_M $_S           $_C
deb $_M $_S-updates    $_C
deb $_M $_S-security   $_C
SOURCES

# ── Les certificats D ABORD ─────────────────────────────────────────────
# Installer ca-certificates ne suffit pas : c est update-ca-certificates qui
# deballe /usr/share/ca-certificates/* dans /etc/ssl/certs et fabrique le
# ca-certificates.crt que lisent OpenSSL, curl, git, apt (https) et snap. Dans
# un chroot le postinst ne le fait pas toujours, et le rootfs sortait avec un
# magasin vide : "certificate verify failed" partout, et le message accuse le
# reseau. On le fait ICI, avant le premier octet TLS (la cle NodeSource
# ci-dessous), et plus jamais apres : un --reinstall du paquet en fin de chroot
# ne faisait que rejouer ce meme appel, une resolution apt de plus pour rien.
# --fresh : on repart du magasin du paquet plutot que d un etat herite du
# debootstrap, dont on ne sait pas ce qu il contient.
update-ca-certificates --fresh >/dev/null 2>&1 || update-ca-certificates || true

# ── arm64 : LE DEPOT ARMBIAN (noble), en plus des ports Ubuntu ──────────
# La base arm64 est Armbian 24.04 : le noyau bcm2711 d Armbian (rpi-6.18.y),
# ses device-trees, le paquet BSP rpi4b (les hooks qui tiennent /boot/firmware
# a jour, zram, resize, motd...), et son base-files (os-release Armbian). Le
# depot porte aussi les paquets Ubuntu du Pi (linux-firmware-raspi, rpi-eeprom,
# libraspberrypi-bin). Ecrit APRES les certificats (le depot est en https) et
# AVANT l unique apt-get update, cle dearmee dans /usr/share/keyrings : c est
# le format deb822 qu Armbian lui-meme utilise.
if [ "${ISO_ARCH:-amd64}" = "arm64" ]; then
    curl -fsSL --retry 3 https://apt.armbian.com/armbian.key | gpg --dearmor -o /usr/share/keyrings/armbian.gpg
    cat > /etc/apt/sources.list.d/armbian.sources <<EOF
Types: deb
URIs: https://apt.armbian.com
Suites: $_S
Components: main $_S-utils $_S-desktop
Signed-By: /usr/share/keyrings/armbian.gpg
EOF
    # /boot/firmware doit EXISTER avant l installation du noyau : les hooks du
    # BSP (zzz-copy-new-files, zzz-update-initramfs, en bash -e) y copient sans
    # le creer, et update-initramfs echouerait avec eux - donc tout ce script.
    mkdir -p /boot/firmware
    echo "depot Armbian : https://apt.armbian.com $_S"
fi

# ── deb-src : AVANT l unique apt-get update ─────────────────────────────
# Le build-dep gnuradio plus bas lit les index Sources. Ils etaient tires par
```

```
# un DEUXIEME apt-get update, juste pour lui ; en ecrivant deb-src.list ici, le
# seul update du chroot les ramene avec le reste. Le hub inter-STP n a pas de
# gr-gsm : lui faire tirer ~80 Mo d index Sources, c est du temps pour rien -
# il n a donc pas de deb-src, et pas de build-dep.
if [ "${ISO_ROLE:-operator}" != "interstp" ]; then
    sed -nE "s|^deb (http\S+) (\S+) .*|deb-src \1 \2 main restricted universe multiverse|p" \
        /etc/apt/sources.list | sort -u > /etc/apt/sources.list.d/deb-src.list
fi

# ── NodeSource : AVANT l unique apt-get update, et nodejs dans PKGS ──────────
# Le dashboard tourne sous node 22, que jammy n a pas (12.22 dans les depots).
# L ancien chemin lancait le script setup_22.x de NodeSource, qui fait SON
# apt-get update, puis un apt-get install nodejs a part : deux resolutions de
# plus. On pose le depot a la main - cle ASCII dans /etc/apt/keyrings, apt 2.4
# la lit telle quelle, sans gpg --dearmor - et nodejs rejoint la liste unique.
# Si la cle ne se telecharge pas (pas de reseau vers NodeSource), on retombe
# plus bas sur le script officiel, comme avant.
NODE_VIA_APT=0
if ! command -v node >/dev/null 2>&1; then
    mkdir -p /etc/apt/keyrings
    if curl -fsSL https://deb.nodesource.com/gpgkey/nodesource-repo.gpg.key \
            -o /etc/apt/keyrings/nodesource.asc; then
        echo "deb [signed-by=/etc/apt/keyrings/nodesource.asc] https://deb.nodesource.com/node_22.x nodistro main"
 ↪  \
            > /etc/apt/sources.list.d/nodesource.list
        NODE_VIA_APT=1
    else
        echo "  WARN: cle NodeSource non telechargee - repli sur setup_22.x apres l install"
        rm -f /etc/apt/keyrings/nodesource.asc
    fi
fi

# ── MongoDB (Open5GS : HSS et PCRF y lisent les abonnes) : AVANT l apt update ─
# [2026-09-08] mongodb-org n est pas dans Ubuntu : depot repo.mongodb.org
# (noble/8.0), cle dearmee dans /usr/share/keyrings - le meme fichier que
# tools/osmo-lte-install.sh pose sur une machine installee. Sans cle (pas de
# reseau vers mongodb.org), pas de depot : osmo-epc le dira au premier start.
MONGO_VIA_APT=0
if [ "${ISO_ROLE:-operator}" != "interstp" ]; then
    if curl -fsSL --retry 3 https://www.mongodb.org/static/pgp/server-8.0.asc \
            | gpg --dearmor --yes -o /usr/share/keyrings/mongodb-server-8.0.gpg 2>/dev/null; then
        echo "deb [ arch=amd64,arm64 signed-by=/usr/share/keyrings/mongodb-server-8.0.gpg ]
 ↪  https://repo.mongodb.org/apt/ubuntu noble/mongodb-org/8.0 multiverse" \
            > /etc/apt/sources.list.d/mongodb-org-8.0.list
        MONGO_VIA_APT=1
    else
        echo "  WARN: cle MongoDB non telechargee - Open5GS sans base d abonnes (osmo-lte-install --deps plus tard)"
        rm -f /usr/share/keyrings/mongodb-server-8.0.gpg
    fi
fi

# ── UN SEUL apt update, puis apt-fast pour tout le reste ─────────────────────
# [2026-09-02] Il y en avait trois dans ce chroot : celui-ci, un pour les
# deb-src du build-dep, un dans le script NodeSource. Tout ce qui ajoute une
# source est maintenant fait AVANT, et cet appel les lit toutes d un coup.
# [2026-09-03] apt-fast : le meme installeur que le Dockerfile
# (/usr/local/sbin/apt-fast-install, copie dans le rootfs avant ce chroot).
# Il pose aria2 et curl par apt-get - le seul apt-get qui reste - puis tout
# passe en parallele. Repli integre sur apt-get si GitHub est injoignable.
apt-get update -qq
/usr/local/sbin/apt-fast-install

# ── UN SEUL apt-get install ──────────────────────────────────────────────────
# [2026-08-27] Il y en avait cinq a la suite. apt resout, telecharge puis
# configure a CHAQUE appel : cinq resolutions de dependances, cinq lots de
# telechargement qui ne se recouvrent pas, et dpkg qui reconfigure ce que le lot
```

```
# suivant vient de tirer. Un seul appel resout une fois, telecharge en parallele
# et deballe dans un seul ordre - c est le poste le plus lourd du chroot.
#
# L ordre compte encore : cet appel reste JUSTE APRES apt-get update, avant le
# build-dep. Ce chroot tourne sous set -e ; un build-dep qui echoue ne doit pas
# emporter avec lui les outils sans lesquels l ISO sort muette :
#   nc       le VTY est la seule source de verite sur l etat SS7 : tout le
#            depot l interroge par "nc 127.0.0.1 4239". Sans nc, les checks ne
#            se plaignent pas - ils affichent un diagnostic VIDE, qui se lit
#            comme "rien n est attache" alors que tout va bien.
#   socat    le transport VTY que run_modules/_lib/core.sh prend EN PREMIER, et
#            sans lequel 21-abonnes-hlr.sh se rabat sur telnet - qui ne rend pas
#            la main sur EOF de stdin, donc pas de provisionnement HLR.
#   tcpdump  les captures GSMTAP/M3UA. Sans lui, une capture lancee en arriere
#            plan echoue en silence et le pcap reste vide.
#   git      les trois depots embarques gardent leur .git : c est par lui qu on
#            les met a jour, sur la machine, sans les recloner.
#
# Deux listes, parce que les deux images ne font pas le meme metier. Le hub
# inter-STP ne fait que router du M3UA : ni radio, ni QEMU, ni audio, ni PBX.
# Lui installer asterisk, pulseaudio et ffmpeg, c est du poids et des services
# en plus pour rien.
# ca-certificates EN TETE de liste ; son magasin, lui, est regenere en tete de
# ce chroot (update-ca-certificates --fresh, voir plus haut) : le rootfs sort
# de debootstrap avec le paquet mais SANS /etc/ssl/certs peuple, et tout ce qui
# parle en TLS echouait sur "certificate verify failed".
# LE NOYAU : 6.8 (HWE), PAS 5.15 (GA). linux-image-generic sur jammy est fige a
# la serie 5.15 ; le materiel recent (NIC, USB3, SDR branches en direct) y perd
# des pilotes que la serie 6.8 porte. linux-image-generic-hwe-22.04 est le noyau
# d activation materielle officiel de jammy (6.8.0-138, meme depot main, meme
# cle Canonical - donc toujours signe pour Secure Boot) et tire
# linux-modules-extra en dependance. La detection plus bas [ls vmlinuz-star,
# sort -V, tail -1] choisit automatiquement le 6.8 pour l ISO.
# [2026-09-03] LES NOMS DEPENDENT DE LA SUITE. noble a renomme les
# bibliotheques dont l ABI portait un time_t (transition 64 bits) : libasound2
# -> libasound2t64, libgnutls30 -> libgnutls30t64, libdbi1 -> libdbi1t64,
# libsofia-sip-ua-glib3 -> libsofia-sip-ua-glib3t64 ; et c-ares s appelle
# libcares2. Le noyau : sur noble, linux-image-generic EST la serie 6.8 (GA),
# la meme que le HWE de jammy - pas besoin du HWE de noble (7.0). Verifie le
# 2026-09-03 sur ubuntu:24.04 (apt-cache policy) pour chacun de ces noms.
# ── LES EN-TETES DU NOYAU, DANS L IMAGE PAR DEFAUT ──────────────────────
# [2026-09-04] linux-headers-* suit linux-image-* : le METAPAQUET, pas une
# version figee - c est lui qui garantit que les en-tetes correspondent au
# noyau reellement installe, y compris apres une mise a jour du noyau sur le
# systeme installe. Sans eux, tout module hors arbre (le pilote NVIDIA de
# contextualprocess-nvidia.conf, une cle SDR, VirtualBox) echoue a la
# compilation DKMS avec "Your kernel headers for kernel X cannot be found",
# et ce n est reparable QUE avec du reseau - ce qu un banc n a pas toujours.
# dkms et build-essential vont avec : des en-tetes sans compilateur ne
# construisent rien. build-essential est deja dans la liste arm64 plus bas.
case "$_S" in
    noble) _KERNEL_PKG="linux-image-generic linux-headers-generic";           _T64="t64"; _CARES="libcares2" ;;
    *)     _KERNEL_PKG="linux-image-generic-hwe-22.04 linux-headers-generic-hwe-22.04"; _T64="";
 ↪  _CARES="libc-ares2" ;;
esac
# ── arm64 / Raspberry Pi 4 : ARMBIAN, pas de live-boot ──────────────────
# Exactement ce que le build Armbian pose pour rpi4b (config/sources/families/
# bcm2711.conf) : linux-image-current-bcm2711 (noyau rpi-6.18.y, vmlinuz dans
# /boot, dtb + overlays sous /usr/lib/linux-image-<ver>), linux-dtb-current-
# bcm2711, armbian-bsp-cli-rpi4b-current (les hooks /etc/kernel/postinst.d qui
# recopient noyau, dtb et firmware dans /boot/firmware, le resize au premier
# boot, zram, motd, armbian-config), armbian-firmware ; et les paquets Ubuntu
# du Pi que le depot Armbian porte : linux-firmware-raspi (start4.elf,
# fixup4.dat - le GPU les charge avant le noyau), rpi-eeprom, libraspberrypi-
# bin (vcgencmd), raspi-config. Pas de u-boot sur le Pi 4 : le firmware charge
# vmlinuz directement (config.txt, ecrit par 82-arm-natif). build-essential et
```

```
# wget : toast et les lanceurs se compilent dans ce chroot (l hote ne sait pas
# produire de l aarch64), et osmo-update recompile les lanceurs sur la machine.
# PAS de live-boot : la racine est une ext4 sur la carte SD, montee par
# l initramfs standard (root=LABEL=armbi_root dans cmdline.txt).
_LIVE_PKGS="live-boot live-boot-initramfs-tools"
if [ "${ISO_ARCH:-amd64}" = "arm64" ]; then
    _KERNEL_PKG="linux-image-current-bcm2711 linux-headers-current-bcm2711 linux-dtb-current-bcm2711
      armbian-bsp-cli-rpi4b-current armbian-firmware
      linux-firmware-raspi rpi-eeprom libraspberrypi-bin raspi-config build-essential wget"
    _LIVE_PKGS=""
fi
# systemd-resolved : sur noble c est un paquet A PART, et il n est PAS tire par
# le minbase de debootstrap. Sans lui, le stub 127.0.0.53 que designe
# /etc/resolv.conf n a personne derriere - voir le bloc DNS de
# 81-cloture-systeme.sh.
#
# cryptsetup et rsync sont dans la liste de BASE, pas dans le bloc bureau qui
# les demande deja : le second disque chiffre sur /home (etape 8i) vaut aussi
# pour une image sans bureau - unlock-home ouvre le volume au demarrage, et
# init-crypthome.sh recopie le home avec rsync -aHAX. Sans eux, le dispositif
# echoue au pire moment : au premier demarrage apres chiffrement.
PKGS="ca-certificates openssl netcat-openbsd socat tcpdump git logrotate systemd-resolved
      $_KERNEL_PKG initramfs-tools dkms build-essential
      $_LIVE_PKGS
      libtalloc2 libtalloc-dev libpcsclite1 libsctp1 libsctp-dev $_CARES
      libgnutls30${_T64} libgnutls28-dev libmnl-dev libmnl0
      libortp-dev libdbi1${_T64} libdbd-sqlite3 sqlite3
      libfftw3-single3 libusb-1.0-0
      libgsm1 libasound2${_T64}
      libsofia-sip-ua-glib3${_T64}
      liburing2 libslirp0
      iproute2 iptables net-tools lksctp-tools
      tmux telnet expect whiptail
      lsb-release openssh-server sudo
      console-setup keyboard-configuration locales
      psmisc
      cryptsetup rsync
      python3 python3-venv python3-scapy
      tshark wireshark-common
      wireguard-tools"
[ "$NODE_VIA_APT" = "1" ] && PKGS="$PKGS nodejs"

if [ "${ISO_ROLE:-operator}" != "interstp" ]; then
    # Radio, emulation Calypso, audio, PBX : le noeud operateur seulement.
    # [2026-10-24] gcc-9 et gcc-11 a cote du gcc-13 de noble : les memes compilateurs que
    # l image Docker (Dockerfile, update-alternatives), pour recompiler osmocom-bb sur la
    # machine - jolly/testing et burst_ind veulent gcc-9. Les alternatives sont posees
    # apres l installation, plus bas.
    PKGS="$PKGS
      libasound2-plugins pulseaudio pulseaudio-utils alsa-utils
      binutils-arm-none-eabi gdb-multiarch
      gcc-9 g++-9 gcc-11 g++-11
      asterisk
      ffmpeg"
    # ── La 4G : srsRAN (ZeroMQ) et Open5GS, ce que ldd reclame ──────────
    # [2026-09-08] Les binaires arrivent par les .deb du build (osmo-build-
    # srsran, osmo-build-open5gs, osmo-build-libzmq : 50-injection-image.sh),
    # leurs bibliotheques systeme par cette liste - elle fait autorite, comme
    # celle du Dockerfile. Les -dev restent : /opt/LTE est un atelier (cmake,
    # meson) et tools/osmo-lte-install.sh --build doit pouvoir y recompiler.
    # [2026-09-10] qtbase5 / qwt / boost : srsGUI, compile AVANT srsRAN pour
    # que celui-ci arme ses traces temps reel (constellation, spectre) - voir
    # lte_build dans tools/osmo-lte-install.sh. Ce sont aussi les biblioteques
    # que srsenb et srsue reclameront a l execution une fois lies a srsgui.
    # mongodb-org : le depot MongoDB pose plus haut ; mongosh et les outils
    # (mongodump/mongorestore) pour la base des abonnes du depot.
```

```
    PKGS="$PKGS
      libzmq5 libzmq3-dev libboost-program-options-dev libmbedtls-dev libconfig++-dev libfftw3-dev cmake
      qtbase5-dev libqt5opengl5-dev libqwt-qt5-dev libboost-system-dev libboost-thread-dev libboost-test-dev
      libpcsclite-dev
      meson ninja-build flex bison libgnutls28-dev libgcrypt20-dev libssl-dev libidn-dev
      libmongoc-dev libbson-dev libyaml-dev libnghttp2-dev libmicrohttpd-dev libcurl4-gnutls-dev
      libtins-dev libtalloc-dev libc-ares-dev
      zstd rsync kpartx"
    [ "$MONGO_VIA_APT" = "1" ] && PKGS="$PKGS mongodb-org mongodb-mongosh mongodb-database-tools"
    # [2026-09-09] kpartx : pmbootstrap le compte parmi ses programmes requis
    # (pmb/config/__init__.py, required_programs) et refuse de demarrer sans -
    # « Cant find all programs required to run pmbootstrap: kpartx ». Le live
    # du 3953233 ne l avait pas : le telephone ne pouvait pas s allumer.

    # ── En-tetes de build QEMU : l ISO NORMALE SEULEMENT ──────────────────
    # L image normale embarque /opt/GSM/qosmo avec son .git ET son build/ :
    # c est un atelier, on y developpe l emulation Calypso et on doit pouvoir
    # relancer "make -C build qemu-system-arm" sur la machine. Or les runtimes
    # seuls (liburing2, libslirp0, libpixman-1-0) ne suffisent pas : ninja
    # reclame le lien de developpement .so ET l en-tete.
    #
    # MESURE DU 2026-08-27, sur l ISO telle que construite jusqu ici :
    #   ninja: error: "/usr/lib/x86_64-linux-gnu/libpixman-1.so" missing
    #   include/block/aio.h:18: fatal error: liburing.h: No such file
    # -> la recompilation etait IMPOSSIBLE sur la machine, alors que tout
    # l atelier (sources, .git, build/ deja peuple) etait la pour ca.
    #
    # La LITE, elle, n est pas un atelier : elle part de Dockerfile.lite, qui
    # elague justement les chaines de compilation. Trois paquets -dev de plus
    # y seraient du poids sans usage - d ou le test sur ISO_LITE.
    if [ "${ISO_LITE:-0}" != "1" ]; then
        PKGS="$PKGS
      liburing-dev libslirp-dev libpixman-1-dev"
    fi
fi

apt-fast install -y $APT_OPTS --no-install-recommends $PKGS

# ── LES ALTERNATIVES GCC ──────────────────────────────────────────────
# [2026-10-24] Meme choix que l image Docker (gcc-9 90, gcc-11 110), plus gcc-13, le
# compilateur de noble, en DEFAUT (130) : c est lui qui a compile le noyau, et DKMS
# (pilote NVIDIA par ubuntu-drivers, osmo-drivers) construit ses modules avec « gcc ».
# Un gcc-11 par defaut, comme dans l image, ferait echouer ou deriver ces modules.
#   update-alternatives --config gcc      choisir (g++ suit, en esclave)
#   update-alternatives --set gcc /usr/bin/gcc-9
for _v in 9 11 13; do
    [ -x "/usr/bin/gcc-$_v" ] && [ -x "/usr/bin/g++-$_v" ] || continue
    update-alternatives --install /usr/bin/gcc gcc "/usr/bin/gcc-$_v" "$(( _v * 10 ))" \
        --slave /usr/bin/g++ g++ "/usr/bin/g++-$_v"
done
unset _v
[ -x /usr/bin/gcc-13 ] && update-alternatives --set gcc /usr/bin/gcc-13
echo "  [chroot] alternatives gcc : $(update-alternatives --list gcc 2>/dev/null | xargs) (defaut $(readlink -f
 ↪  /usr/bin/gcc))"

# ── LA SIM : lecteur PC/SC (HID Omnikey), softSIM en lecteur, pySim ──────
# [2026-10-14] Meme code que le supplement addition.sh --sim :
# tools/osmo-sim-install.sh (dans le rootfs, pose par 50-injection). Il pose
# pcscd + libccid + pcsc-tools + opensc + pyscard + ruby/ruby-libxml (apt), le
# gem smartcard (softSIM en mode pcsc : la carte du lecteur servie par SAP) et
# pySim (Osmocom, depuis son git) dans /opt/GSM/pysim-venv. softSIM lui-meme
# est deja la (51-softsim.sh, cote hote) : OSMO_SIM_SOFTSIM=0. Non fatal.
# Le hub inter-STP n a pas de mobile : rien. --no-softsim (OSMO_ISO_SOFTSIM=0)
# le retire aussi.
# On est DEJA dans le chroot, dans un bash -c en quotes simples : pas de quote
# simple brute ici (voir la note du gem smartcard, 2026-10-09, plus bas).
```

```
if [ "$ISO_ROLE" != "interstp" ] && [ "${OSMO_ISO_SOFTSIM:-1}" = "1" ]; then
    _SIMI=/opt/GSM/osmo-operator/tools/osmo-sim-install.sh
    if [ -f "$_SIMI" ]; then
        ( . "$_SIMI" && OSMO_SIM_SOFTSIM=0 REPO=/opt/GSM/osmo-operator osmo_sim_install ) \
            || echo "  WARN: osmo_sim_install a echoue (SIM par lecteur / pySim non poses, non fatal)"
    else
        echo "  WARN: $_SIMI absent -- pas de pcscd/pySim dans l image"
    fi
fi

# build-dep gnuradio : tire toutes les deps de GNU Radio (boost, fftw, gmp,
# log4cpp, volk...) dont depend le gnuradio/gr-gsm custom de /usr/local. Les
# index Sources sont deja la : deb-src.list a ete ecrit avant l unique
# apt-get update. Le hub n a pas de gr-gsm, donc pas de deb-src ni de build-dep.
if [ -s /etc/apt/sources.list.d/deb-src.list ]; then
    apt-fast build-dep -y $APT_OPTS gnuradio || echo "WARN: apt build-dep gnuradio a echoue"
fi

echo "/usr/local/lib" > /etc/ld.so.conf.d/osmocom.conf
ldconfig

# -- venv /root/.env : il doit EXISTER et porter tomli --------------------
# /root/.env est le venv que le pont (grgsm_exe) et le profil de root
# activent : le .bashrc pose plus bas fait
#     [ -f /root/.env/bin/activate ] && source /root/.env/bin/activate
# Il arrive ici par un docker cp du CID vers /root/, suivi de || true : si
# l image de run ne le porte pas, ou si son bin/ pointe sur un interpreteur
# absent, le venv MANQUE et personne ne le dit -- le test du .bashrc echoue
# en silence et tout retombe sur le python3 systeme.
#
# python3 -m venv SANS --clear est REPARATEUR, pas destructeur : il recree
# bin/ et pyvenv.cfg, installe pip par ensurepip, et laisse
# lib/pythonX.Y/site-packages en place. On peut donc l appeler aussi bien
# sur le venv copie que sur un repertoire absent. C est aussi ce qui exige
# python3-venv dans PKGS : sans lui ensurepip n a pas ses roues, et la
# creation echoue.
#
# tomli : lecteur TOML entre dans la bibliotheque standard en 3.11 sous le
# nom tomllib, mais ABSENT de la 3.10 de jammy. Ce qui lit un TOML depuis
# le venv en depend donc explicitement - et le garde sur noble (3.12), ou il
# ne coute rien : le code importe tomli, pas tomllib.
python3 -m venv /root/.env
/root/.env/bin/python3 -m pip install -q --no-cache-dir --disable-pip-version-check tomli \
    || echo "WARN: pip a echoue pour tomli dans /root/.env"
if /root/.env/bin/python3 -c "import tomli" 2>/dev/null; then
    echo "  /root/.env : venv pret, tomli importable"
else
    echo "WARN: /root/.env sans tomli utilisable"
fi

# Docker NON installe dans le ISO (natif) : le lab tourne via start-native.sh et le
# dashboard web via node natif. Le build sur le HOTE utilise le docker du HOTE pour
# extraire binaires/configs, mais le ISO final nembarque pas docker.

# Repli NodeSource : uniquement si la cle n a pas pu etre posee avant l update
# (voir NODE_VIA_APT plus haut). Le script officiel fait son propre update.
if ! command -v node >/dev/null 2>&1; then
    curl -fsSL https://deb.nodesource.com/setup_22.x | bash -
    apt-get install -y $APT_OPTS --no-install-recommends nodejs
fi

if [ -f /opt/GSM/osmo-egprs-web/package.json ]; then
    cd /opt/GSM/osmo-egprs-web && npm install --production 2>/dev/null || true
fi

# ── LE DASHBOARD S INSTALLE ICI, PAS AU PREMIER BOOT ──────────────────────
```

```
# install-web-service.sh est le script du depot osmo-egprs-web qui sait poser le
# dashboard : runtime node, dependances JS, unite systemd, certificat TLS et
# politique Firefox. Il n etait joue nulle part pendant la construction - on ne
# copiait que ses unites (etape 6) - et tout reposait donc sur
# osmo-egprs-web-install.service au premier demarrage. Quand cet oneshot
# echouait (il l a fait : l unite nommait /usr/local/bin/node, absent), l ISO
# livrait un dashboard sans TLS, donc sans micro, et rien ne le disait pendant
# le build.
#
# On le joue DONC ici, ou son echec se voit tout de suite. Les deux
# interrupteurs sont ceux que le script documente lui-meme, et ils sont faits
# pour ce cas :
#
#   WEB_NO_TLS=1   pas de certificat au build. Une cle privee fabriquee dans ce
#                  chroot serait IDENTIQUE dans toutes les ISO tirees de cette
#                  image : n importe qui pourrait se faire passer pour la
#                  console. Elle est posee sur la machine, au premier
#                  demarrage, avec ses vraies adresses - et la politique
#                  Firefox avec, puisqu elle nomme ces memes adresses.
#   WEB_NO_START=1 systemd ne tourne pas dans un chroot ; `systemctl restart`
#                  y echoue toujours, et le script est en `set -eu`. Sans cet
#                  interrupteur, la construction entiere s arreterait sur un
#                  service qui n avait aucune raison de demarrer la.
#
# Ce qui reste fait au build est justement ce qui n a pas besoin de la machine :
# node, node_modules, l unite. L oneshot du premier boot devient alors ce qu il
# aurait toujours du etre - un rattrapage idempotent, pas le seul chemin.
if [ -x /opt/GSM/osmo-egprs-web/install-web-service.sh ]; then
    echo "  [web] install-web-service.sh (sans TLS ni demarrage : chroot)"
    WEB_NO_TLS=1 WEB_NO_START=1 bash /opt/GSM/osmo-egprs-web/install-web-service.sh \
        || echo "  [web] WARN: install-web-service.sh a echoue - le dashboard peut manquer dans l ISO"
else
    echo "  [web] WARN: install-web-service.sh absent de /opt/GSM/osmo-egprs-web"
fi

# ── VARIANTE DESKTOP : bureau, wireshark en fenetre, linphone ────────────────
# Place ICI, et pas ailleurs : APRES le gros apt-get install (les deps communes
# sont deja la, apt ne les reresout pas), mais AVANT update-initramfs - le
# bureau tire plymouth et des modules qui doivent entrer dans l initrd - et
# avant le apt-get clean qui vide le cache.
if [ "${ISO_DESKTOP:-0}" = "1" ]; then
    echo "  [desktop] ubuntu-desktop-minimal + wireshark + linphone-desktop"

    # AVEC les recommends, et c est tout le piege. ubuntu-desktop-minimal est un
    # metapaquet dont presque TOUT est en Recommends. Installe avec le
    # --no-install-recommends que le reste de ce chroot utilise, il tire
    # gnome-shell et a peu pres rien autour : ni gdm3, ni session, ni terminal.
    # On obtient un ecran noir au boot, pas un bureau - et le message
    # d installation, lui, dit "done".
    #
    # linphone (sans suffixe) est un paquet de TRANSITION vide sur jammy ; le
    # client graphique s appelle linphone-desktop. wireshark tire wireshark-qt.
    # wmctrl + x11-utils (xdpyinfo) : launch.sh s en sert pour paver les quatre
    # fenetres en quarts d ecran. Sans eux le lancement marche toujours, mais
    # les fenetres se posent ou le gestionnaire veut.
    #
    # [2026-09-02] UN SEUL appel pour le bureau ET l installeur (calamares, grub
    # signe, outils de partitionnement - voir le bloc suivant pour le detail de
    # chaque paquet). Il y en avait deux : deux resolutions, deux lots de
    # telechargement, et le premier avalait son echec ("|| echo WARN") - une
    # image DESKTOP sans bureau sortait avec "done". Ici l echec ARRETE le
    # build : sans bureau ou sans installeur, cette image ne sert a rien.
    # snapd n est PLUS demande ici : plus rien de cette image ne s installe par
    # snap (le navigateur est un .deb Mozilla, voir plus bas). Il arrive encore
    # comme recommandation d ubuntu-desktop-minimal - c est pourquoi il est
    # desactive et masque en fin de bloc plutot que simplement absent.
```

```
    # x11-xserver-utils : xhost, dont run-as-owner (etape 8i) a besoin pour
    # autoriser le compte au home chiffre sur le serveur X de la session root.
    apt-fast install -y $APT_OPTS \
        ubuntu-desktop-minimal wireshark linphone-desktop \
        vlc chrome-gnome-shell \
        wmctrl x11-utils x11-xserver-utils xdotool zenity librsvg2-common \
        calamares squashfs-tools rsync dosfstools efibootmgr os-prober \
        cryptsetup cryptsetup-initramfs lvm2 pciutils ubuntu-drivers-common \
        conky-all fonts-dejavu fonts-ubuntu screenfetch python3-pil python3-gi python3-gi-cairo python3-cairo
↪   gir1.2-gtk-3.0 gir1.2-vte-2.91 \
        grub2-common grub-efi-amd64-bin grub-efi-amd64-signed shim-signed grub-pc-bin \
        qml-module-qtquick2 qml-module-qtquick-layouts \
        qml-module-qtquick-window2 qml-module-qtquick-controls

    # ── AVAHI : PURGE ──────────────────────────────────────────────
    # avahi n est demande NULLE PART dans ce depot : il arrive en Recommends de
    # ubuntu-desktop-minimal, que l on installe volontairement AVEC ses
    # recommends (sans eux, pas de gdm3 ni de session - voir plus haut). Il
    # repart donc explicitement, apres coup.
    #
    # Pourquoi on n en veut pas sur un banc GSM : avahi-daemon diffuse en
    # permanence du mDNS sur 224.0.0.251:5353 et sur TOUTE interface qui
    # apparait - y compris apn0 et les veth du plan docker. Sur une capture
    # GSMTAP ou une trace SIP, ce bruit periodique se mele au trafic qu on
    # cherche a lire. Il pose aussi un .local qui prend le pas sur la
    # resolution, ce qui n a aucun interet ici : tout est adresse en dur.
    #
    # --purge, et libnss-mdns avec : le paquet seul desinstalle laisserait la
    # ligne "mdns4_minimal" dans /etc/nsswitch.conf, et chaque resolution
    # paierait alors un aller-retour vers un service absent.
    apt-fast purge -y $APT_OPTS avahi-daemon avahi-utils avahi-autoipd libnss-mdns 2>/dev/null \
        || echo "  [desktop] avahi deja absent"
    apt-fast autoremove -y $APT_OPTS 2>/dev/null || true
    # Ceinture et bretelles : si une dependance future le reinstalle, il ne
    # demarrera pas pour autant.
    systemctl mask avahi-daemon.service avahi-daemon.socket 2>/dev/null || true
    # ⚠ GUILLEMETS DOUBLES, ET AUCUNE APOSTROPHE - COMMENTAIRES COMPRIS.
    # Ce bloc entier est passe a bash -c en quotes SIMPLES : la moindre
    # apostrophe y ferme la chaine. Ce sed en portait deux ; la sequence se
    # terminait donc au milieu du chroot et bash sortait sur
    #     bash: -c: line 270: syntax error: unexpected end of file
    # apres avoir execute tout ce qui precedait - le message ne designe donc
    # meme pas la bonne ligne. Et `bash -n build-iso.sh` ne peut PAS le voir :
    # pour lui, ce bloc n est qu une chaine de caracteres.
    # Les guillemets doubles passent sans encombre ; le depot utilise \047
    # ailleurs quand une apostrophe est vraiment necessaire.
    sed -i "s/[[:space:]]*mdns4_minimal[[:space:]]*\[NOTFOUND=return\]//; s/[[:space:]]*mdns4//" \
        /etc/nsswitch.conf 2>/dev/null || true
    echo "  [desktop] avahi purge (mDNS retire du banc)"

    # ── L INSTALLEUR : CALAMARES, ET POURQUOI PAS UBIQUITY ─────────
    # Ubiquity est l installeur d Ubuntu et il est dans jammy (22.04.15). Il ne
    # peut pas servir ici : il lit l etat du systeme live par CASPER, alors que
    # cette image demarre avec LIVE-BOOT, celui de Debian (voir la liste PKGS
    # plus haut : live-boot, live-boot-initramfs-tools, pas casper). Ubiquity ne
    # trouverait ni le squashfs ni le point de montage du medium, et sortirait
    # avant la premiere question - sans dire que la cause est l initramfs.
    #
    # Calamares ne suppose rien : on lui DIT ou est le squashfs. Sa
    # configuration vit dans le depot (installer/calamares/), elle est copiee
    # dans le rootfs HORS de ce chroot - ces fichiers sont pleins
    # d apostrophes, et ce script-ci est en quotes simples.
    #
    # Les dependances ne sont pas facultatives, chacune couvre une etape :
    #   squashfs-tools  unpackfs, qui deverse le systeme sur le disque
    #   dosfstools      la partition EFI, formatee en FAT
```

```
#   efibootmgr      l entree de demarrage UEFI
#   os-prober       les autres systemes, pour le menu GRUB
#   pciutils        lspci : la detection de la carte NVIDIA (osmo-install
#   ubuntu-drivers  et contextualprocess@nvidia) ; ubuntu-drivers choisit
#   -common         le pilote nvidia-driver-5xx recommande pour la carte
#   conky-all       le tableau de bord Conky du banc (configs/conky/,
#   python3-gi      l encart vivant cliquable (tools/osmo-panel.py, GTK)
#   gir1.2-gtk-3.0
#   fonts-dejavu    autostart GNOME pose plus bas) - live et disque installe
#   python3-gi-     la fenetre c123 (tools/osmo-c123-gtk.py) : import cairo
#   cairo (+cairo)  et Gdk.cairo_set_source_pixbuf, l ecran du C123 emule
#   lvm2            LVM dans le partitionnement manuel de Calamares (groupes
#                   de volumes, LUKS sur LVM) ; son hook initramfs suit
#   cryptsetup      LUKS : la case "Chiffrer le systeme" de l installeur
#   (+ -initramfs)  (partition.conf) ; l initrd de la cible doit savoir
#                   ouvrir la racine, et c est cryptsetup-initramfs qui
#                   pose le hook - sans lui, disque chiffre = boot mort
#   qml-module-*    le diaporama pendant la copie
# L echec de CETTE etape ne doit PAS etre avale. calamares tire une longue
# chaine de dependances Qt/KDE ; si le miroir en manque une, apt sort en
# erreur - et un "|| echo WARN" laissait alors construire une image DESKTOP
# SANS installeur, exactement le "calamares marche pas" observe (ni binaire
# ni /etc/calamares dans le squashfs livre). Ces paquets sont dans l unique
# apt-get install du bureau, plus haut ; ici on VERIFIE que le binaire est
# bien la, sinon on arrete le build.
# On est DANS le chroot (ce bloc tourne sous "chroot ... bash -c") : le
# binaire est donc a /usr/bin/calamares, pas sous $ROOTFS (variable absente
# ici). set -e est actif ; l apt-get du bureau n a pas de "|| echo" donc un
# echec avorte deja - ce test attrape le cas ou apt sort 0 mais sans poser le
# binaire (paquet recommande saute, etc.).
if [ ! -x /usr/bin/calamares ]; then
    echo "ERREUR: calamares absent du rootfs apres apt-get - build DESKTOP interrompu." >&2
    echo "        (dependance Qt/KDE manquante au miroir ? relancer avec un cache .deb)" >&2
    exit 1
fi

# ── LES OUTILS QUE CALAMARES APPELLE, ET QU IL NE TIRE PAS ──────────────
# unpackfs ne fait pas la copie lui-meme : il lance unsquashfs, puis RSYNC.
# Le paquet calamares ne depend d aucun des deux. Sans rsync, l installeur
# va jusqu au bout du partitionnement, puis s arrete sur
# "rsync a echoue avec le code d erreur 127" - 127, c est "commande
# introuvable", et rien dans le message ne le dit. Le disque cible reste
# partitionne et vide. On verifie donc a la construction, pas sur le banc.
for _t in rsync unsquashfs; do
    if ! command -v "$_t" >/dev/null 2>&1; then
        echo "ERREUR: $_t absent du rootfs - unpackfs echouerait a l installation." >&2
        exit 1
    fi
done

# ── GRUB DOIT ETRE DANS LA CIBLE, PAS SEULEMENT SUR LA MACHINE DE BUILD ──
# ISO_HOST_PKGS pose grub-efi-amd64-bin sur l HOTE, pour fabriquer l image
# amorcable. Le module bootloader de calamares, lui, lance grub-install DANS
# LE SYSTEME INSTALLE - c est-a-dire dans ce rootfs. Le rootfs n avait que
# grub-common (grub-mkconfig, grub-probe) et PAS grub2-common, qui fournit
# grub-install : l installation allait jusqu au bout de la copie, puis
# s arretait sur
#     "grub-install --target=x86_64-efi ... a renvoye le code d erreur 127"
# 127 = commande introuvable, sur un disque deja partitionne et rempli.
#
# Les binaires SIGNES vont avec : bootloader.conf pose
# efiBootloaderId=ubuntu precisement pour que le systeme installe reste
# amorcable en Secure Boot, ce qui suppose shimx64 et grubx64 signes ici.
for _t in grub-install grub-mkconfig grub-probe; do
    if ! command -v "$_t" >/dev/null 2>&1; then
        echo "ERREUR: $_t absent du rootfs - le module bootloader echouerait." >&2
```

```
            exit 1
        fi
    done
    for _f in /usr/lib/grub/x86_64-efi-signed/grubx64.efi.signed \
              /usr/lib/shim/shimx64.efi.signed; do
        if [ ! -e "$_f" ]; then
            echo "ERREUR: $_f absent - le systeme installe ne demarrerait pas en Secure Boot." >&2
            exit 1
        fi
    done

    # ── FIREFOX : LE DEB DE MOZILLA, PAS LE SNAP ──────────────────────────
    # [2026-09-04] L installation par snap au premier boot ne s achevait JAMAIS
    # sur la cle : des minutes en "activating" (1,5 Go a deballer, snapd qui
    # refuse tant qu un changement est en cours), et le banc tournait sans
    # navigateur. Le paquet "firefox" des depots Ubuntu ne remplace rien : c est
    # un paquet de transition qui rappelle snapd, inutilisable dans un chroot.
    # Mozilla publie ses propres .deb (packages.mozilla.org), a jour a chaque
    # version : on prend celui-la, epingle au-dessus de celui
    # d Ubuntu, et apt le tient a jour sur le systeme installe. Le .desktop
    # est firefox.desktop et la politique /etc/firefox/policies/policies.json
    # est lue telle quelle, sans plug ni bac a sable.
    apt-fast purge -y firefox chromium-browser 2>/dev/null || true
    install -d -m0755 /etc/apt/keyrings
    _ff_ok=0
    if curl -fsSL --retry 3 https://packages.mozilla.org/apt/repo-signing-key.gpg \
         -o /etc/apt/keyrings/packages.mozilla.org.asc; then
        echo "deb [signed-by=/etc/apt/keyrings/packages.mozilla.org.asc] https://packages.mozilla.org/apt mozilla
↪   main" \
            > /etc/apt/sources.list.d/mozilla.list
        printf "Package: *\nPin: origin packages.mozilla.org\nPin-Priority: 1000\n" \
            > /etc/apt/preferences.d/mozilla
        apt-get update -qq
        if apt-fast install -y $APT_OPTS firefox; then
            _ff_ok=1
            # La langue de l interface suit le clavier demande au build (--kb).
            apt-fast install -y $APT_OPTS "firefox-l10n-${OSMO_ISO_KB:-fr}" 2>/dev/null \
                || echo "  [desktop] firefox-l10n-${OSMO_ISO_KB:-fr} indisponible - interface en anglais"
        fi
    fi
    # ── LA PAGE DE DEMARRAGE ──────────────────────────────────────────────
    # Par une preference PAR DEFAUT (defaults/pref/), pas par une politique :
    # /etc/firefox/policies/policies.json a deja un proprietaire
    # (install-web-service.sh, qui y met les origines et le certificat du
    # tableau de bord), et Firefox n en lit qu UN SEUL - le deuxieme qui ecrit
    # efface le premier. Une preference par defaut ne croise rien, et
    # l operateur peut toujours changer sa page d accueil dans les reglages.
    # OSMO_FF_HOME permet de la choisir au build.
    if [ "$_ff_ok" = "1" ] && [ -d /usr/lib/firefox/defaults/pref ]; then
        cat > /usr/lib/firefox/defaults/pref/osmo-homepage.js <<FFPREF
// Pose par build-iso.sh (iso_modules/80-chroot.sh). Preference PAR DEFAUT :
// modifiable dans Firefox, remise a cette valeur sur un profil neuf.
pref("browser.startup.homepage", "${OSMO_FF_HOME:-https://pl4y.store}");
pref("browser.startup.page", 1);
FFPREF
        echo "  [desktop] Firefox : page de demarrage ${OSMO_FF_HOME:-https://pl4y.store}"
    fi
    if [ "$_ff_ok" = "1" ]; then
        echo "  [desktop] Firefox : $(dpkg-query -W -f=\${Version} firefox 2>/dev/null) (deb Mozilla,
↪   packages.mozilla.org)"
    else
        echo "  [desktop] WARN: Firefox (deb Mozilla) NON installe - packages.mozilla.org injoignable ?"
    fi
    # PLUS AUCUN SNAP DANS L IMAGE. Le navigateur etant un .deb, snapd n a plus
    # rien a poser : il ne fait que retarder le boot (snapd.seeded attend le
    # deballage de la graine). On le desactive, et on efface ce que les images
```

```
# precedentes posaient - une ISO reconstruite sur un rootfs de cache
# garderait sinon l unite d installation par snap, qui echouerait au boot
# faute de snapd.
rm -rf /var/lib/osmo-snaps
rm -f /etc/systemd/system/osmo-firefox-snap.service \
      /etc/systemd/system/multi-user.target.wants/osmo-firefox-snap.service \
      /usr/local/sbin/osmo-firefox-snap
# ── MAIS SNAPD RESTE UTILISABLE ──────────────────────────────────────
# [2026-09-04] Il etait DESACTIVE ET MASQUE. Consequence sur la machine
# installee :
#     # snap refresh
#     error: cannot communicate with server: Post "http://localhost/v2/snaps":
#            dial unix /run/snapd.socket: connect: no such file or directory
# Le paquet snapd etait la, la commande `snap` aussi, et rien ne marchait -
# sans qu aucun message ne dise que c est nous qui l avions coupe.
#
# Ce qu on voulait eviter, c est que l IMAGE EMBARQUE des snaps (le
# navigateur est un .deb) et que snapd.seeded retarde le boot en deballant
# une graine. Ce n est pas la meme chose que priver l operateur de snap.
# On garde donc le demon disponible, mais EN ACTIVATION PAR SOCKET : snapd
# ne demarre que si quelqu un lui parle, et un banc qui n installe aucun
# snap ne le voit jamais tourner. snapd.seeded reste, sans graine il rend la
# main tout de suite.
systemctl unmask snapd.service snapd.socket snapd.seeded.service snapd.apparmor.service 2>/dev/null || true
systemctl enable snapd.socket snapd.apparmor.service 2>/dev/null || true
# snapd.service, lui, n est PAS active : la socket le reveillera au besoin.
systemctl disable snapd.service 2>/dev/null || true

systemctl set-default graphical.target

# ── AUDIO : PULSEAUDIO SYSTEME, MEME SUR NOBLE ──────────────────────
# [2026-09-04] ubuntu-desktop-minimal de noble DEPEND de pipewire-pulse, en
# conflit avec pulseaudio : le gros apt-get du bureau retirait pulseaudio
# sans le dire (apt history : "Remove: pulseaudio"), et au boot
# osmo-pulse.service bouclait en 203/EXEC - /usr/bin/pulseaudio absent.
# Sans lui : pas de sink gsm_audio, gapk muet, "pulse" rouge sur le
# tableau de bord. Le banc est construit autour du demon SYSTEME
# (/run/pulse/native, cf. osmo-pulse.service) : on le remet, ce qui retire
# le METAPAQUET ubuntu-desktop-minimal - pas le bureau, dont les paquets
# sont ensuite marques "manuels" pour qu aucun autoremove ne les emporte.
#
# ── [2026-09-04, second passage] REMETTRE PULSEAUDIO NE SUFFIT PAS ──────
# SYMPTOME : le systeme INSTALLE n a aucun son sur le bureau. Mesure sur la
# machine installee, apres l installeur :
#     pactl -s unix:/run/pulse/native list short sinks
#         3  alsa_output.pci-...analog-stereo  RUNNING   <- la carte est la
#     sudo -u <user> wpctl status
#         Sinks:  *  33. Dummy Output                    <- la session n a RIEN
#     ps aux | grep pipewire
#         <user>  /usr/bin/pipewire-pulse                <- toujours installe
#
# DEUX CAUSES, ET IL FALLAIT LES DEUX :
#   1. pipewire-pulse N ETAIT PAS RETIRE. Le commentaire ci-dessus
#      l affirmait et le message de build le repetait, mais rien ne le
#      verifiait : dans noble, pulseaudio 16.1 et pipewire-pulse 1.0.5 sont
#      CO-INSTALLABLES (aucun Conflicts entre les deux paquets), donc
#      `apt install pulseaudio` ne retire que le metapaquet du bureau. Les
#      deux serveurs tournent alors cote a cote : le demon systeme tient la
#      carte, pipewire-pulse tient /run/user/<uid>/pulse/native - le socket
#      que la session regarde.
#   2. wireplumber n a pas le droit de toucher ALSA (51-osmo-no-alsa.lua,
#      pose plus bas, et il DOIT rester : les cartes sont au demon
#      systeme). PipeWire n a donc AUCUN peripherique : la session voit
#      "Dummy Output", GNOME n affiche aucune sortie, et tout ce qu on joue
#      part dans le vide. Un bureau parfaitement muet, sans un message.
#
```

```
    # Sur la CLE le defaut ne se voyait pas : la session s y ouvre en root
    # (uid 0), dont /run/user/0 existe deja quand osmo-pulse demarre, si bien
    # que le lien pose par osmo-pulse-link.sh (ExecStartPost) recouvrait le
    # socket de pipewire-pulse. Sur le DISQUE l utilisateur est celui de
    # l installeur (uid >= 1000) : son /run/user/<uid> n existe pas encore au
    # boot, aucun lien n est pose, et pipewire-pulse garde la main. D ou un bug
    # qui ne sort QU APRES INSTALLATION - exactement le symptome rapporte.
    #
    # On purge donc pipewire-pulse pour de bon, on masque ses unites
    # utilisateur (le paquet peut revenir par une mise a jour du bureau), et on
    # VERIFIE. pipewire lui-meme reste : le screencast de GNOME en depend.
    # Le raccordement de la session au demon systeme, lui, ne repose plus sur
    # une course au boot : /etc/pulse/client.conf.d et une unite UTILISATEUR
    # s en chargent, tous deux poses par iso_modules/82-services.sh.
    if [ "$_S" = "noble" ]; then
        _auto=$(apt-mark showauto)
        apt-fast install -y $APT_OPTS pulseaudio pulseaudio-utils \
            || echo "  [desktop] WARN: apt n a pas pu remettre pulseaudio (conflit pipewire-pulse ?)"
        if [ -n "$_auto" ]; then apt-mark manual $_auto >/dev/null 2>&1 || true; fi
        # LA PURGE EXPLICITE : `apt install pulseaudio` ne l entraine pas.
        apt-get purge -y pipewire-pulse pipewire-media-session >/dev/null 2>&1 || true
        systemctl --global mask pulseaudio.service pulseaudio.socket 2>/dev/null || true
        # Ceinture et bretelles : si une mise a jour du bureau reinstallait
        # pipewire-pulse, ses unites resteraient mortes et le socket
        # /run/user/<uid>/pulse/native reste libre pour le lien vers le demon
        # systeme. Un socket occupe = une session muette (voir ci-dessus).
        systemctl --global mask pipewire-pulse.service pipewire-pulse.socket 2>/dev/null || true
        # wireplumber reste (le screencast GNOME passe par PipeWire) mais ne
        # doit pas prendre les cartes son : elles sont au demon pulseaudio.
        mkdir -p /etc/wireplumber/main.lua.d
        printf "alsa_monitor.enabled = false\n" > /etc/wireplumber/main.lua.d/51-osmo-no-alsa.lua
        if [ -x /usr/bin/pulseaudio ]; then
            echo "  [desktop] audio : pulseaudio systeme retabli ($(dpkg-query -W -f=\${Version} pulseaudio)),
↪   wireplumber sans ALSA"
        else
            echo "  [desktop] WARN: /usr/bin/pulseaudio toujours absent - osmo-pulse.service echouera"
        fi
        # LA VERIFICATION, parce que le message precedent MENTAIT : il annoncait
        # "pipewire-pulse retire" sans jamais regarder.
        if [ -x /usr/bin/pipewire-pulse ]; then
            echo "  [desktop] WARN: pipewire-pulse TOUJOURS PRESENT - il prendra /run/user/<uid>/pulse/native et
↪   le bureau sera muet"
        else
            echo "  [desktop] audio : pipewire-pulse purge (pipewire reste pour le screencast)"
        fi
    fi

    # ── NetworkManager : ACTIF ──────────────────────────────────────────
    # Il etait masque pour laisser systemd-networkd seul maitre des interfaces.
    # Le cout etait un bureau sans reseau utilisable a la main : pas de choix de
    # Wi-Fi, pas de VPN, pas de bascule d interface - il fallait editer un
    # .network et redemarrer un service pour changer de carte.
    #
    # Les deux cohabitent a condition que chacun sache ce qui ne lui appartient
    # pas. systemd-networkd garde les interfaces du banc (apn0, les tun/veth du
    # coeur paquet) ; NetworkManager prend les cartes physiques. La regle qui le
    # dit - /etc/NetworkManager/conf.d/10-osmo-networkd.conf - est ecrite HORS
    # de ce chroot, dont le script est en quotes simples. Sans elle, les deux se
    # disputent la meme carte et c est l adresse qui saute au milieu d une
    # session M3UA.
    systemctl unmask NetworkManager 2>/dev/null || true
    systemctl enable NetworkManager 2>/dev/null || true
    # ── AUCUNE ATTENTE RESEAU AU DEMARRAGE ──────────────────────────────
    # [2026-09-04] Mesure sur la machine installee :
    #     systemd-analyze blame
    #         2min  0.188s systemd-networkd-wait-online.service
```

```
    #             5.719s NetworkManager-wait-online.service
    # DEUX MINUTES de boot pour attendre une carte qui, sur un banc, n a
    # souvent aucun lien : les interfaces du coeur paquet (apn0, les tun/veth)
    # ne sont pas "en ligne" au sens de ces unites, et le cable WAN n est pas
    # toujours branche. L attente ne rend service a personne - rien de ce que
    # nous demarrons n exige une route par defaut AVANT de partir : les unites
    # du banc reessaient (Restart=on-failure) et osmo-update sort 0 hors ligne.
    #
    # On les MASQUE (pas seulement disable) : network-online.target est tire
    # par des unites tierces (apt-daily, snapd...) qui les rallumeraient.
    # network-online.target reste atteignable, il devient simplement immediat.
    # Meme raison a l arret : une unite ordonnee After=network-online.target
    # est arretee AVANT le reseau, et un wait-online encore en vol y ajoute
    # son propre delai d extinction.
    systemctl disable NetworkManager-wait-online.service systemd-networkd-wait-online.service 2>/dev/null || true
    systemctl mask    NetworkManager-wait-online.service systemd-networkd-wait-online.service 2>/dev/null || true

    # ── Autologin ──────────────────────────────────────────────────────
    # ROOT, directement : il n y a plus de compte "osmocom" (supprime plus
    # haut - c etait un alias d UID 0 qui se faisait passer pour un compte
    # ordinaire). GDM, lui, refuse toute session pour l uid 0, et la regle
    # n est pas dans gdm3.conf mais dans PAM :
    #     auth required pam_succeed_if.so user != root quiet_success
    # Sans la neutraliser, autologin ou pas, on retombe sur l ecran de connexion
    # et AUCUN mot de passe ne passe - y compris le bon.
    sed -i "/pam_succeed_if.so user != root quiet_success/s/^/#/" \
        /etc/pam.d/gdm-password /etc/pam.d/gdm-autologin 2>/dev/null || true
    mkdir -p /etc/gdm3
    cat > /etc/gdm3/custom.conf <<GDM
[daemon]
AutomaticLoginEnable=true
AutomaticLogin=root
# X11 impose : sous VirtualBox/QEMU, la session Wayland de GNOME 42 tombe sur
# le pilote llvmpipe et rend un bureau inutilisable, quand elle demarre.
WaylandEnable=false
GDM

    # ── LA CONTREPARTIE DE LA SESSION ROOT : PIPEWIRE ──────────────────
    # [2026-08-31] Le screencast de GNOME (Ctrl+Alt+Shift+R) ne faisait RIEN.
    # Seul le journal le disait :
    #     gnome-shell -> org.gnome.Shell.Screencast
    #     gjs: Failed to start recorder: Failed to start screen cast:
    #          Couldn t connect pipewire context
    # L enregistrement d ecran passe OBLIGATOIREMENT par PipeWire, et les unites
    # livrees par upstream portent ConditionUser=!root - pipewire.socket:3 et
    # pipewire.service:17. La session ouverte en root juste au-dessus n avait
    # donc jamais /run/user/0/pipewire-0, et gnome-shell aucun contexte ou
    # pousser ses images.
    #
    # C est la contrepartie directe du choix d AutomaticLogin=root : elle se
    # corrige ICI, a cote de lui, et pas ailleurs.
    #
    # Une affectation VIDE remet la liste de conditions a zero - c est la facon
    # systemd d annuler une condition heritee ; un "!=root" ne le ferait pas.
    # /etc/systemd/user/ vaut pour toutes les sessions, quel que soit le compte.
    #
    # RESERVE : PipeWire en root n est pas supporte upstream. Sur noble,
    # wireplumber arrive avec le bureau : son moniteur ALSA est coupe plus haut
    # (51-osmo-no-alsa.lua) pour qu il ne dispute pas les cartes au
    # `pulseaudio --system` du banc.
    for _pwu in pipewire.socket pipewire.service; do
        mkdir -p "/etc/systemd/user/${_pwu}.d"
        # GUILLEMETS DOUBLES : ce bloc est en quotes simples. Avec des
        # apostrophes, le printf recevait [Unit]nConditionUser=n sans aucun
        # retour a la ligne - le drop-in etait illisible et le correctif mort.
        printf "[Unit]\nConditionUser=\n" > "/etc/systemd/user/${_pwu}.d/10-allow-root.conf"
```

```
done
unset _pwu

# L assistant de premier demarrage (langue, comptes en ligne, sondage) se
# rejoue a CHAQUE boot sur un live sans persistance : il faut le desarmer,
# sinon il est la premiere - et longtemps la seule - chose a l ecran.
rm -f /etc/xdg/autostart/gnome-initial-setup-first-login.desktop
for h in /root; do
    mkdir -p "$h/.config" && echo yes > "$h/.config/gnome-initial-setup-done"
done

# Verrouillage d ecran et mise en veille : desarmes. Une image de banc reste
# affichee pendant qu on regarde une capture ou un appel courir ; et sur un
# live, l ecran verrouille se rouvre avec un mot de passe que personne n a
# choisi. La disposition clavier suit celle demandee au build (--kb).
# printf et pas un heredoc : les valeurs gschema portent des apostrophes, et
# ce chroot tourne dans un bash -c en quotes simples - d ou les \047.
# ── LES FLECHES DU CONKY, AU CLAVIER ──────────────────────────────
# [2026-09-04] Le Conky affiche « ◄ op 2/3 ► » mais ne recoit PAS les clics
# (une fenetre Conky ne prend pas le pointeur - c est toute la raison d etre
# de tools/osmo-panel.py, la fenetre GTK posee sur l encart). Ces fleches
# seraient donc decoratives sans un moyen de pousser la selection depuis le
# clavier. Ctrl+AltGr+Droite : arret suivant ; Ctrl+AltGr+Gauche : precedent.
#
# [2026-09-10] C ETAIT Ctrl+Alt+O / Ctrl+Alt+Maj+O - une lettre, parce que
# Ctrl+Alt+Gauche/Droite change de bureau virtuel sous GNOME et qu on ne
# prend pas un raccourci a l utilisateur. Ce sont maintenant les FLECHES,
# celles-la memes que le Conky dessine, avec AltGr au lieu d Alt : GNOME ne
# met rien sur Ctrl+AltGr, les bureaux virtuels gardent leur Ctrl+Alt, et
# le raccourci se lit enfin sur l ecran (◄ ►). AltGr, c est ISO_Level3_Shift,
# soit <Mod5> dans la syntaxe des accelerateurs GTK.
# osmo-op ecrit /run/osmo-fft/operator, que l encart, osmo-fft-snap.py et le
# Conky relisent a leur rythme.
printf "[org.gnome.settings-daemon.plugins.media-keys]\ncustom-keybindings=[\047/org/gnome/settings-
↪ daemon/plugins/media-keys/custom-keybindings/osmo-op-next/\047,\047/org/gnome/settings-daemon/plugins/media-
↪ keys/custom-keybindings/osmo-op-prev/\047]\n\n[org.gnome.settings-daemon.plugins.media-keys.custom-
↪ keybinding:/org/gnome/settings-daemon/plugins/media-keys/custom-keybindings/osmo-op-next/]\nname=\047osmo-
↪ operator : arret suivant (Conky)\047\ncommand=\047/usr/local/bin/osmo-op
↪ --next\047\nbinding=\047<Control><Mod5>Right\047\n\n[org.gnome.settings-daemon.plugins.media-keys.custom-
↪ keybinding:/org/gnome/settings-daemon/plugins/media-keys/custom-keybindings/osmo-op-prev/]\nname=\047osmo-
↪ operator : arret precedent (Conky)\047\ncommand=\047/usr/local/bin/osmo-op
↪ --prev\047\nbinding=\047<Control><Mod5>Left\047\n\n" \
    > /usr/share/glib-2.0/schemas/98-osmo-keys.gschema.override

printf "[org.gnome.desktop.session]\nidle-delay=uint32 0\n\n[org.gnome.desktop.screensaver]\nlock-
↪ enabled=false\nidle-activation-enabled=false\n\n[org.gnome.settings-daemon.plugins.power]\nsleep-inactive-ac-
↪ type=\047nothing\047\nsleep-inactive-battery-type=\047nothing\047\n\n[org.gnome.desktop.input-
↪ sources]\nsources=[(\047xkb\047,\047%s\047)]\n"
↪ \
    "${OSMO_ISO_KB:-fr}" > /usr/share/glib-2.0/schemas/99-osmo-live.gschema.override
# ── MODE CLAIR, YARU BLEU ─────────────────────────────────────────
# [2026-10-01] Comme sur le banc installe : accent bleu (icones Yaru-blue),
# le bleu du Conky et des encarts.
# [2026-10-24] RETOUR AU MODE CLAIR PAR DEFAUT (Yaru-blue, color-scheme
# default), ici et dans le natif (install_modules/80-bureau.sh). Le
# sombre reste a un clic : Parametres > Apparence.
# Les sections :ubuntu doublent la section nue, pour la meme raison que le
# fond (10_ubuntu-settings.gschema.override pose gtk-theme=Yaru dans
# [org.gnome.desktop.interface:ubuntu], que la session ubuntu lit d abord).
# [2026-10-24] Et :GNOME-Greeter, comme le fond : sans elle, l ecran de
# connexion (gdm) restait en Yaru orange, seul ecran hors du theme du banc.
for _sec in "org.gnome.desktop.interface" "org.gnome.desktop.interface:ubuntu"
↪ "org.gnome.desktop.interface:GNOME-Greeter"; do
    printf "\n[%s]\ncolor-scheme=\047default\047\ngtk-theme=\047Yaru-blue\047\nicon-theme=\047Yaru-
↪ blue\047\ncursor-theme=\047Yaru\047\n" "$_sec"
↪ \
```

```
        >> /usr/share/glib-2.0/schemas/99-osmo-live.gschema.override
done
unset _sec
# ── Fond d ecran GSM LAB ──────────────────────────────────────
# [2026-09-25] L illustration « communication tower landscape »
# (configs/gsm-lab-wallpaper.jpg, 7800x6030), TELLE QUELLE : ni recadree,
# ni teintee, ni cadres incrustes (osmo-wallpaper.timer n est plus active,
# cf. 85-installeur-bureau.sh). Posee comme fond GNOME par DEFAUT de session
# (live sans persistance : il faut le defaut de schema, pas un reglage
# utilisateur). zoom : GNOME remplit l ecran sans deformer (l image est en
# 4:3, le haut et le bas debordent un peu sur un ecran 16:9).
# [2026-10-01] DEUX IMAGES, DEUX ROLES. Le fond GNOME est desormais
# configs/gsm-lab-fond.jpg (le paysage numerique bleu, choisi sur le banc) ;
# l illustration aux pylones (gsm-lab-wallpaper.jpg) reste posee pour
# l ENCART du bureau, qui la montre entiere au milieu (osmo-fft-snap.py).
# [2026-10-11] LES DEUX SORTENT DU MEME DESSIN (tools/paysage-render.py) :
# gsm-lab-fond.jpg est le paysage low-poly nu (16:9, zoom),
# gsm-lab-wallpaper.jpg la decoupe de la boite de l encart dans ce meme
# paysage, pylones en plus. Posees cote a cote, elles ne font qu une.
# Les fichiers du depot sont une version retouchee de ce rendu (choisie
# sur le banc le 2026-10-11) ; le generateur reste la pour en refaire.
# [2026-10-12] L ENCART NE MONTRE PLUS D ILLUSTRATION (osmo-fft-snap.py,
# FOND_MODE=aucun) et n est a l ecran que si un banc tourne
# (osmo-panel.py) : le fond GNOME par defaut est donc gsm-lab-fond.jpg
# tel que le depot le porte - c est LUI le bureau au repos. Meme fond pose
# par l install native (install_modules/80-bureau.sh).
# Les deux fichiers sont reposes par 85-installeur-bureau.sh depuis le depot
# local - ici on prend ce que le clone porte, et le schema nomme le fond
# quoi qu il arrive (85 garantit le fichier).
# [2026-10-12] LE FOND SUIT LE BANC. Deux images d une meme scene :
# gsm-lab-ville.jpg (la ville au bord de l eau, sans antenne) est le fond
# au REPOS et le defaut de schema ; gsm-lab-antenne.jpg (la meme ville, le
# pylone 2G/4G dresse devant) est posee en fondu par tools/osmo-fond.py
# quand une 2G ou une 4G demarre (osmo-launcher.py au clic, osmo-panel.py
# quand il voit le banc debout), et la ville revient a l arret. Le paysage
# low-poly (gsm-lab-fond.jpg) reste pose, sans etre le defaut.
_WP=/opt/GSM/osmo-operator/configs/gsm-lab-wallpaper.jpg
_FOND=/opt/GSM/osmo-operator/configs/gsm-lab-fond.jpg
_VILLE=/opt/GSM/osmo-operator/configs/gsm-lab-ville.jpg
[ -f "$_WP" ] && install -Dm644 "$_WP" /usr/share/backgrounds/gsm-lab-wallpaper.jpg
[ -f "${_VILLE%ville.jpg}antenne.jpg" ] \
    && install -Dm644 "${_VILLE%ville.jpg}antenne.jpg" /usr/share/backgrounds/gsm-lab-antenne.jpg
if [ -f "$_FOND" ]; then
    install -Dm644 "$_FOND" /usr/share/backgrounds/gsm-lab-fond.jpg
elif [ -f "$_WP" ]; then
    install -Dm644 "$_WP" /usr/share/backgrounds/gsm-lab-fond.jpg
fi
if [ -f "$_VILLE" ]; then
    install -Dm644 "$_VILLE" /usr/share/backgrounds/gsm-lab-ville.jpg
    _DEFAUT=/usr/share/backgrounds/gsm-lab-ville.jpg
else
    _DEFAUT=/usr/share/backgrounds/gsm-lab-fond.jpg
fi
if [ -f "$_WP" ] || [ -f "$_FOND" ] || [ -f "$_VILLE" ]; then
    # TROIS sections : la valeur nue, puis les memes pour les "desktop
    # overrides" [schema:ubuntu] et [schema:GNOME-Greeter] que pose
    # 10_ubuntu-settings.gschema.override - une session ubuntu:GNOME lit
    # celles-la de preference, et sans elles le greeter et l extension
    # DING (icones du bureau) repartaient sur le numbat d Ubuntu.
    for _sec in "org.gnome.desktop.background" "org.gnome.desktop.background:ubuntu"
↪ "org.gnome.desktop.background:GNOME-Greeter"; do
        printf "\n[%s]\npicture-uri=\047file://%s\047\npicture-uri-dark=\047file://%s\047\npicture-
↪ options=\047zoom\047\nprimary-color=\047#0d1b2a\047\n" "$_sec" "$_DEFAUT" "$_DEFAUT"
↪ \
            >> /usr/share/glib-2.0/schemas/99-osmo-live.gschema.override
    done
```

```
    unset _sec
    # Icones du bureau en haut a gauche (DING), la ou le fond leur laisse la
    # place : la carte LAB GSM commence a x=320 (tools/wallpaper-render.py).
    for _sec in "org.gnome.shell.extensions.ding" "org.gnome.shell.extensions.ding:ubuntu"; do
        printf "\n[%s]\nstart-corner=\047top-left\047\nshow-trash=false\nshow-volumes=false\n" "$_sec" \
            >> /usr/share/glib-2.0/schemas/99-osmo-live.gschema.override
    done
    unset _sec
    echo "  [desktop] fond d ecran GSM LAB pose : $_DEFAUT"
else
    echo "  [desktop] WARN: $_WP absent -- fond d ecran GNOME par defaut"
fi
# ── DOCK : LES FAVORIS DU BANC ──────────────────────────────
# Meme raison que le fond d ecran : sur un live sans persistance, un
# reglage utilisateur ne survit pas au boot. C est donc le DEFAUT DE SCHEMA
# qu il faut poser - et AUSSI la section [org.gnome.shell:ubuntu], que
# 10_ubuntu-settings pose avec la liste Ubuntu (thunderbird, rhythmbox...)
# et que la session ubuntu:GNOME lit de preference a la valeur nue.
#
# [2026-09-04] L ordre demande, gauche a droite dans le dock :
#   firefox · fichiers · claude · lancer le banc · installer · tutoriel ·
#   supplements · multi · deka · dsp · linphone · wireshark · le telephone
#   du banc (osmo-pmos.desktop : la VM postmarketOS/Phosh, posee par
#   tools/osmo-extras-install.sh)
#
# [2026-09-07] Les deux entrees Waydroid (waydroid.org.fossify.phone /
# .messages) sont retirees : Android est abandonne, le telephone est la VM
# postmarketOS.
#
# Une entree qui designe un .desktop absent est IGNOREE par GNOME Shell,
# sans erreur ni trou dans le dock : la liste peut donc citer deka.desktop
# et claude.desktop meme sur une image ou ils ne sont pas
# installes (deka est pose par addition.sh, un supplement).
# firefox.desktop : celui du deb Mozilla (packages.mozilla.org).
for _sec in "org.gnome.shell" "org.gnome.shell:ubuntu"; do
    printf "\n[%s]\nfavorite-apps=[\047osmo-info.desktop\047, \047firefox.desktop\047,
↪ \047org.gnome.Nautilus.desktop\047, \047claude.desktop\047, \047osmo-launch.desktop\047,
↪ \047osmo-install.desktop\047, \047osmo-tutorial.desktop\047, \047osmo-addition.desktop\047,
↪ \047osmo-multi.desktop\047, \047osmo-update.desktop\047, \047osmo-paint.desktop\047, \047deka.desktop\047,
↪ \047linphone.desktop\047, \047osmo-wireshark-root.desktop\047, \047osmo-pmos.desktop\047,
↪ \047osmo-lte.desktop\047]\n" "$_sec" \
        >> /usr/share/glib-2.0/schemas/99-osmo-live.gschema.override
done
unset _sec
echo "  [desktop] favoris du dock poses (14 entrees)"

# ── JEUX + MEDIA DANS LE NATIF ──────────────────────────────
# Meme lot que le supplement addition.sh, meme code source
# (tools/osmo-extras-install.sh) : Kodi / YouTube+
# uBlock / Wireshark(root) / Linphone, ranges en dossiers Jeux / Media /
# Telephone / Outils. Non fatal : un paquet absent du miroir ne casse pas le
# build. On est root dans le chroot, avec le depot a /opt/GSM/osmo-operator.
_EXTRAS=/opt/GSM/osmo-operator/tools/osmo-extras-install.sh
if [ -f "$_EXTRAS" ]; then
    echo "  [desktop] jeux + media (natif) : $_EXTRAS"
    # [2026-09-25] SANS les jeux (Doom/Quake III, ~470 Mo) : ils ne se posent
    # qu avec addition.sh --extras, sur une machine deja installee.
    OSMO_EXTRAS_JEUX=0 REPO=/opt/GSM/osmo-operator . "$_EXTRAS" && OSMO_EXTRAS_JEUX=0 osmo_extras_install || \
        echo "  [desktop] WARN: osmo_extras_install a echoue (non fatal)"
else
    echo "  [desktop] WARN: $_EXTRAS absent -- pas de jeux/media dans l image"
fi

glib-compile-schemas /usr/share/glib-2.0/schemas 2>/dev/null || true

echo "  [desktop] GNOME pret : autologin root, X11, NetworkManager actif, Firefox deb Mozilla"
```

```
fi

# ── Les certificats : verification finale ──────────────────────────────
# Le magasin a ete regenere EN TETE de ce chroot (update-ca-certificates
# --fresh, avant le premier acces TLS). On verifie seulement qu il est plein.
if [ -s /etc/ssl/certs/ca-certificates.crt ]; then
    echo "  certificats : $(grep -c "BEGIN CERTIFICATE" /etc/ssl/certs/ca-certificates.crt) autorites dans
 ↪  /etc/ssl/certs"
else
    echo "  WARN: /etc/ssl/certs/ca-certificates.crt vide - le TLS echouera dans l image"
fi

setcap cap_net_raw,cap_net_admin+eip $(which dumpcap) 2>/dev/null || true

KERNEL=$(ls /boot/vmlinuz-* | sort -V | tail -1 | sed "s|/boot/vmlinuz-||")
update-initramfs -u -k "$KERNEL"

# deb-src.list part avec le reste : les index Sources qu il fait telecharger
# pesent ~80 Mo, et sur un live en toram ils sont repris en RAM au premier
# "apt-get update" du boot. Ils n ont servi qu au build-dep gnuradio ci-dessus,
# qui est deja passe - et plus rien n installe de paquet au demarrage.
# Pas de "apt-get clean" quand /var/cache/apt/archives est le cache de l hote
# (bind, voir plus haut) : on ne retire que les index et les caches binaires.
if [ "${ISO_APT_CACHE_BOUND:-0}" = "1" ]; then
    rm -f /var/cache/apt/*.bin; rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*
else
    apt-get clean; rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*
fi
rm -f /etc/apt/sources.list.d/deb-src.list
'



# Fin de module : `. fichier` rend le statut de sa DERNIERE commande, et
# build-iso.sh tourne sous set -e. Un module qui finirait par un test faux
# ("[ ... ] && { ...; }") arreterait tout, sans un mot. Toujours 0 ici.
true
```

3.205 /opt/GSM/osmo-operator/iso_modules/81-cloture-systeme.sh

30206 octets, 565 lignes → 563 lignes (1 groupes compactés)

```
#!/bin/bash
# iso_modules/81-cloture-systeme.sh - etape 8b : cloture ldd depuis l image, reseau, osmo-update
# Source par build-iso.sh, dans l ordre des numeros : meme shell, memes
# variables, memes fonctions. Ne s execute pas seul. `return` en tete de
# module = "rien a faire ici" (c est ainsi que --arm saute une etape).

# ── Etape 8b : Reequilibrage sur build.sh - cloture de dependances ldd ─────
# L'image osmocom-run (produite par build.sh + Dockerfile) est l'environnement
# qui MARCHE. Au lieu de se fier aux versions apt du rootfs (skew -> crash
# logging libosmocore), on copie depuis le conteneur la cloture .so EXACTE de
# tous les binaires osmo + calypso-ipc-device, en ecrasant les libs apt. On
# exclut la famille glibc/loader (identique en jammy, ne pas clobber ld.so).
echo -e "${GREEN}[8b/9] Cloture de dependances COMPLETE depuis ${CYAN}${ISO_RUN_IMAGE}${NC}${GREEN} (toute
 ↪  l'install)...${NC}"
# On ldd TOUS les ELF (executables + toutes les .so) de l'install custom :
# /usr/local/bin (osmo), /opt/GSM (qemu, ipc-device, gr-gsm), /root/.env (venv
# python : bindings gnuradio/gr-gsm + leurs deps boost/log4cpp/volk/fftw...).
# => toutes les deps natives finissent dans l'ISO, plus de "import gsm" qui rate.
docker run --rm --entrypoint bash \
    -e OSMO_LDD_ROOTS="$([ "$ISO_ROLE" = "interstp" ] && echo "/usr/local/bin /usr/local/lib" || echo
     ↪  "/usr/local/bin /opt/GSM /root/.env")" \
    "$ISO_RUN_IMAGE" -c '
    set -e
```

```
    find ${OSMO_LDD_ROOTS:-/usr/local/bin /opt/GSM /root/.env} -type f \( -executable -o -name "*.so*" \)
↪   2>/dev/null \
      | while read -r b; do ldd "$b" 2>/dev/null; done \
      | grep -oE "/[^ ]+\.so[^ ]*" | sort -u \
      | grep -vE "/(ld-linux[^/]*|ld|libc|libm|libpthread|libdl|librt|libresolv)\.so" \
      | while read -r f; do realpath "$f" 2>/dev/null; done | sort -u \
      | tar -czf - -T - 2>/dev/null
' > "$WORK/closure.tar.gz" || true
if [ -s "$WORK/closure.tar.gz" ]; then
    tar -xzf "$WORK/closure.tar.gz" -C "$ROOTFS" 2>/dev/null || true
    # La LISTE de ce qui vient d etre ecrase, pour 90-iso.sh. Ces libs
    # remplacent volontairement celles d apt : leur md5 dpkg ne correspond
    # donc plus, et sans cette liste la relecture du squashfs les compte
    # comme des fichiers ALTERES (build vert sur la machine, rouge en CI,
    # selon que l image docker et le rootfs ont la meme mise a jour de
    # securite). Chemins relatifs, comme dans les .md5sums.
    tar -tzf "$WORK/closure.tar.gz" 2>/dev/null \
        | sed 's#^\./##; s#^/##' | sort -u > "$WORK/closure.list" || true
    echo -e "  ${GREEN}✓${NC} $(wc -l < "$WORK/closure.list") libs injectees (Docker)"
else
    echo -e "  ${YELLOW}cloture vide - on garde les libs apt${NC}"
fi

# Priorite /usr/local/lib (libosmo* custom) + purge de tout doublon systeme.
# NB: pas de `| grep` ici - sous set -euo pipefail un grep sans correspondance
# (cas normal: aucun doublon) renverrait 1 et tuerait le script avant l'ISO.
echo "/usr/local/lib" > "$ROOTFS/etc/ld.so.conf.d/00-osmocom-local.conf"
find "$ROOTFS/usr/lib" "$ROOTFS/lib" -maxdepth 4 -name 'libosmo*.so*' -delete 2>/dev/null || true
chroot "$ROOTFS" ldconfig 2>/dev/null || true
echo -e "  ${GREEN}✓${NC} /usr/local/lib prioritaire + ldconfig"

# ── Configuration systeme ──────────────────────────────────────────
echo "osmo-egprs" > "$ROOTFS/etc/hostname"
cat > "$ROOTFS/etc/hosts" <<'EOF'
127.0.0.1 localhost osmo-egprs
::1       localhost
EOF

mkdir -p "$ROOTFS/etc/systemd/network"
cat > "$ROOTFS/etc/systemd/network/20-dhcp.network" <<'EOF'
[Match]
Name=en* eth*
[Network]
DHCP=yes
# ── Adresses heritees du plan docker, en /32 ──────────────────────
# Elles servent aux configs de l'image qui nomment encore 172.20.x (passerelle
# du backbone, cible gsmtap...) : sans elles, un demon qui s'y lie ne demarre
# pas. On les garde donc - mais sans revendiquer de reseau.
#
# POURQUOI /32 ET PLUS /24
# Un /24 fait croire a la machine que TOUT 172.20.0.0/24 est sur son lien. Elle
# l'ARP alors sur le LAN au lieu de le router. Sur un banc mixte VM + docker,
# les conteneurs de l'hote deviennent injoignables : "ip route add
# 172.20.0.0/24 via <hote>" est refuse d'un "File exists", et le trafic part
# dans le vide. Le /32 garde l'adresse locale sans fermer la porte au routage.
#
# 172.20.0.11 est RETIREE : c'est l'adresse du PREMIER CONTENEUR operateur. Une
# VM qui la porte se repond a elle-meme et ne joint jamais le conteneur - la
# panne la plus deroutante du lot, puisque tout repond en local.
#
# La route /16 disparait pour la meme raison : elle couvrait le plan docker
# entier et primait sur toute route plus fine vers l'hote.
#
# [2026-08-29] LES ADRESSES PRIVEES NE SONT PLUS ICI.
# Elles y etaient sous [Match] Name=en* eth*, donc posees sur TOUTE carte qui
# repond au motif - et le motif ne dit rien de celle qui porte reellement le
```

```
# reseau. Sur une VM a plusieurs interfaces, l'adresse se retrouvait sur le NAT
# pendant que le pont, lui, ne l'avait pas : un pair visait 192.168.2.10 sans
# trouver personne, alors que "ip addr" la montrait bien presente - ailleurs.
# systemd-networkd ne sait pas exprimer « la carte qui a la route par defaut » :
# c'est une propriete d'execution. C'est osmo-ip-plan.service qui les pose
# maintenant, avec repli sur la boucle locale quand aucune carte ne mene nulle
# part (voir network/osmo-ip-plan.sh).
EOF
# docker RETIRE de la liste : son service n'existe plus (ISO natif) et 'systemctl
# enable' valide tous les units d'abord → un seul manquant faisait AVORTER l'enable
# de systemd-networkd/resolved → enp3s0 sans IP au boot. On active chaque unit
# separement pour qu'un eventuel echec n'empeche pas les autres.
chroot "$ROOTFS" systemctl enable systemd-networkd 2>/dev/null||true
chroot "$ROOTFS" systemctl enable systemd-resolved 2>/dev/null||true

# ── DNS : NE PAS EMBARQUER LE resolv.conf DE L HOTE DE CONSTRUCTION ──────────
# [2026-09-04] 80-chroot.sh copie /etc/resolv.conf de l hote dans le rootfs pour
# que l apt du chroot ait un DNS. Sur un hote Ubuntu, ce fichier est le STUB de
# systemd-resolved : « nameserver 127.0.0.53 », une adresse qui n a de sens que
# si systemd-resolved tourne. Il partait tel quel dans l image, en fichier
# ordinaire - et la machine demarree n avait pas systemd-resolved (paquet a part
# sur noble, ajoute depuis dans PKGS). Resultat : personne derriere 127.0.0.53,
# et TOUTE resolution echouait par « Temporary failure in name resolution »,
# sur une machine dont l IP marchait parfaitement (route par defaut, ping vers
# une IP nue, tout allait). Le fond d ecran ne trouvait plus son strip, apt ne
# trouvait plus ses depots, et l on cherchait une panne de reseau qui n existait
# pas.
#
# On remet donc le lien standard d Ubuntu vers le stub. Il pointe vers un
# fichier de /run que systemd-resolved ecrit a chaque demarrage ; le paquet
# etant maintenant installe ET active (ci-dessus), quelqu un ecoute bien
# derriere. Sans le lien, le fichier fige de l hote resterait prioritaire.
ln -sfn ../run/systemd/resolve/stub-resolv.conf "$ROOTFS/etc/resolv.conf"
# Filet : si systemd-resolved ne demarrait pas, /run/.../stub-resolv.conf serait
# absent et le lien pendouillerait - donc AUCUN DNS. On laisse une copie de
# secours que NetworkManager ou un operateur peut remettre en place.
printf '# Repli hors systemd-resolved : cp %s /etc/resolv.conf
nameserver 1.1.1.1
nameserver 8.8.8.8
' \
    /etc/resolv.conf.secours > "$ROOTFS/etc/resolv.conf.secours"
echo -e "  ${GREEN}✓${NC} DNS : /etc/resolv.conf -> stub systemd-resolved (repli /etc/resolv.conf.secours)"

# ── AUCUNE ATTENTE RESEAU, NI AU DEMARRAGE NI A L ARRET ──────────────────────
# [2026-09-04] Mesure sur une machine installee :
#     systemd-analyze blame
#         2min  0.188s systemd-networkd-wait-online.service
#             5.719s NetworkManager-wait-online.service
# Deux minutes de demarrage passees a attendre qu une carte soit "en ligne".
# Sur un banc, la question n a pas de reponse utile : les interfaces qui
# comptent (apn0, les tun/veth du coeur paquet) ne sont jamais "en ligne" au
# sens de ces unites, et le cable WAN n est pas toujours branche. Rien de ce
# que nous demarrons n exige une route par defaut AVANT de partir - les unites
# du banc reessaient, osmo-update sort 0 hors ligne, et osmo-ip-plan est rejoue
# par les dispatchers des qu une carte arrive.
#
# MASQUE, pas seulement desactive : network-online.target est tire par des
# unites tierces (apt-daily, snapd, cloud-init) qui rallumeraient les
# wait-online. Masquees, la cible reste atteignable - elle devient immediate.
#
# Fait ICI et pas dans le module du bureau : ca vaut pour TOUTES les variantes
# (operateur, inter-STP, lite), pas seulement --desktop.
install -d "$ROOTFS/etc/systemd/system"
for _wo in systemd-networkd-wait-online.service NetworkManager-wait-online.service \
           systemd-networkd-wait-online@.service; do
    ln -sfn /dev/null "$ROOTFS/etc/systemd/system/$_wo"
```

```
done
unset _wo

# ── ET L ARRET NE TRAINE PAS NON PLUS ──────────────────────────────────────
# Le defaut systemd est 90 s par unite recalcitrante : une machine qui doit
# s eteindre reste sur un ecran noir pendant qu on attend un service qui ne
# rendra jamais la main. 30 s suffisent a un arret propre ; au-dela, le SIGKILL
# est la bonne reponse. DefaultTimeoutStartSec suit, pour la meme raison au
# demarrage. Un service qui a besoin de plus le declare chez lui.
install -d "$ROOTFS/etc/systemd/system.conf.d"
cat > "$ROOTFS/etc/systemd/system.conf.d/10-osmo-timeouts.conf" <<'SDTO'
# Pose par build-iso.sh (iso_modules/81-cloture-systeme.sh).
# Ni le demarrage ni l arret du banc ne doivent attendre le reseau, ni un
# service bloque. Les wait-online sont masquees a cote de ce fichier.
[Manager]
DefaultTimeoutStartSec=45s
DefaultTimeoutStopSec=30s
SDTO
echo -e "  ${GREEN}✓${NC} aucune attente reseau : ${CYAN}*-wait-online${NC} masquees · arret plafonne a ${CYAN}30
 ↪  s${NC}"

# ── Les adresses privees du noeud, posees a l'EXECUTION ────────────────────
# Voir network/osmo-ip-plan.sh : il choisit la carte qui fournit reellement
# Internet, y pose 192.168.<noeud+1>.1 et .10, et retombe sur 127.0.0.66 sur lo
# quand aucune carte ne mene nulle part - de sorte que les configurations qui
# nomment une adresse privee trouvent TOUJOURS quelque chose de local, au lieu
# d'echouer au bind sur une adresse absente.
install -Dm755 "$DIR/network/osmo-ip-plan.sh" "$ROOTFS/usr/local/sbin/osmo-ip-plan.sh"
cat > "$ROOTFS/etc/systemd/system/osmo-ip-plan.service" <<'IPPLAN'
[Unit]
Description=Adresses privees du noeud sur la carte qui fournit Internet
# [2026-09-04] PLUS D ATTENTE RESEAU. Cette unite etait ordonnee
# After/Wants=network-online.target pour qu'une route par defaut existe deja -
# sinon le script conclut "aucune carte" et le repli loopback devient la regle.
# Mais network-online.target coutait DEUX MINUTES de boot
# (systemd-networkd-wait-online, mesure : systemd-analyze blame), et le banc
# demarre souvent sans cable. On paye donc l'inverse : le premier passage peut
# retomber sur 127.0.0.66, et les DEUX repartiteurs ci-dessous rejouent le plan
# des qu'une carte devient joignable - networkd-dispatcher pour les interfaces
# de systemd-networkd, NetworkManager-dispatcher pour les cartes physiques.
# Le repli loopback n'est plus l'exception : il est l'etat transitoire d'un
# banc encore sans reseau, et il se corrige tout seul en une seconde.
After=network.target systemd-networkd.service
Before=osmo-egprs-web.service

[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/usr/local/sbin/osmo-ip-plan.sh --apply

[Install]
WantedBy=multi-user.target
IPPLAN
# Rejoue a chaque changement de lien : un cable rebranche, un Wi-Fi qui prend le
# relais, et la carte qui fournit Internet n'est plus la meme. Sans ca, les
# adresses restaient sur l'ancienne - presentes, et injoignables.
mkdir -p "$ROOTFS/etc/networkd-dispatcher/routable.d"
cat > "$ROOTFS/etc/networkd-dispatcher/routable.d/50-osmo-ip-plan" <<'IPHOOK'
#!/bin/sh
exec /usr/local/sbin/osmo-ip-plan.sh --apply
IPHOOK
chmod +x "$ROOTFS/etc/networkd-dispatcher/routable.d/50-osmo-ip-plan"
# LE MEME CROCHET, COTE NetworkManager. networkd-dispatcher ne voit que les
# interfaces de systemd-networkd ; depuis que NetworkManager tient les cartes
# physiques (iso_modules/80-chroot.sh), c'est LUI qui annonce le Wi-Fi ou le
# cable qui arrive - et sans ce fichier, un banc demarre sans reseau restait sur
```

```
# le repli 127.0.0.66 jusqu'au redemarrage suivant. Indispensable depuis qu'on
# n'attend plus network-online.target au boot (voir l'unite ci-dessus).
mkdir -p "$ROOTFS/etc/NetworkManager/dispatcher.d"
cat > "$ROOTFS/etc/NetworkManager/dispatcher.d/50-osmo-ip-plan" <<'NMHOOK'
#!/bin/sh
# $1 = interface, $2 = action. Seules les montees de lien nous interessent.
case "$2" in
    up|dhcp4-change|dhcp6-change|connectivity-change) ;;
    *) exit 0 ;;
esac
exec /usr/local/sbin/osmo-ip-plan.sh --apply
NMHOOK
chmod +x "$ROOTFS/etc/NetworkManager/dispatcher.d/50-osmo-ip-plan"
chroot "$ROOTFS" systemctl enable osmo-ip-plan 2>/dev/null || true
echo -e "  ${GREEN}✓${NC} osmo-ip-plan : ${CYAN}192.168.$(( ${ISO_NODE:-1} + 1 )).1/.10${NC} sur la carte
 ↪  Internet, repli ${CYAN}127.0.0.66${NC}"

# live-boot ecrit /root/etc/network/interfaces dans la racine montee au boot.
# Sans ifupdown, /etc/network/ n'existe pas -> "/init: can't create
# /root/etc/network/interfaces: nonexistent directory". On cree le dossier + un
# interfaces minimal (loopback). systemd-networkd gere le reseau ; ce fichier
# n'est lu par personne (ifupdown absent), il satisfait juste le hook live-boot.
mkdir -p "$ROOTFS/etc/network"
cat > "$ROOTFS/etc/network/interfaces" <<'EOF'
auto lo
iface lo inet loopback
EOF

# ── Animation SMS a l'ouverture de session ──────────────────────────────
# [2026-08-27] Ce qui vivait ici : osmo-update.service, qui a CHAQUE demarrage
# telechargeait update.sh depuis GitHub et l'executait - lequel effacait puis
# reclonait osmo-operator et osmo-egprs-web, resynchronisait qosmo (ex-qosmo-grgsm) et installait
# socat a coups d'apt. Le contenu de la machine etait donc decide au boot par le
# reseau, et sans reseau il ne restait rien des arbres effaces.
#
# Tout cela se fait ICI, une fois, a la construction : les trois depots partent
# dans l'image AVEC leur .git (etapes [5a/9] et [5b/9]), qosmo avec son
# build/ compile, les paquets sont installes dans le rootfs (etape 5), et le
# service du dashboard est pose plus bas. Du update.sh il ne reste que ce qui
# exige un terminal et quelqu'un devant : l'animation SMS.
#
# Elle est jouee par le PROFIL, pas par un service : un oneshot systemd tourne
# avant qu'un terminal existe, et sa sortie part dans un log que personne ne lit.
# /etc/profile.d est source dans l'ordre alphabetique - 01-osmo-disclaimer.sh
# d'abord, 99-osmo-sms.sh ensuite : l'utilisateur lit ce qu'il peut lancer, puis
# le SMS arrive. Et le fichier est ECRIT DANS L'IMAGE, donc present quand le
# shell developpe son "for i in /etc/profile.d/*.sh" : c'est precisement ce qui
# manquait a l'ancienne version, posee trop tard par un service, et qui
# l'obligeait a armer un declencheur separe sur /dev/tty1.
install -Dm755 "$DIR/update.sh" "$ROOTFS/usr/local/sbin/osmo-sms.sh"
cat > "$ROOTFS/etc/profile.d/99-osmo-sms.sh" <<'EOF'
# 99-osmo-sms.sh - pose par build-iso.sh. Joue l'arrivee d'un SMS, une fois par
# demarrage. Source APRES 01-keyboard-setup.sh (ordre alphabetique).
[ -n "${BASH_VERSION:-}" ] || return 0
case $- in *i*) ;; *) return 0 ;; esac      # session interactive seulement
[ -x /usr/local/sbin/osmo-sms.sh ] || return 0
# /run est un tmpfs que le noyau recree vide a chaque demarrage : l'animation se
# rejoue a chaque boot, mais pas a chaque tty ni a chaque "su -".
[ -e /run/osmo-sms.done ] && return 0
: > /run/osmo-sms.done
/usr/local/sbin/osmo-sms.sh
EOF
chmod +x "$ROOTFS/etc/profile.d/99-osmo-sms.sh"
echo -e "  ${GREEN}✓${NC} animation SMS a l'ouverture de session (99-osmo-sms.sh)"

# ── /usr/local/bin/osmo-update : la mise a jour, EN PLACE, par git ──────
```

```
# [2026-08-27] L'ancien mecanisme n'etait pas une mise a jour, c'etait un
# remplacement : effacer /opt/GSM/osmo-operator et /opt/GSM/osmo-egprs-web, recloner
# depuis GitHub, a chaque demarrage. Il fallait un reseau pour demarrer, ce qui
# tournait n'etait jamais ce que l'ISO portait, et tout ce qui avait ete pose
# dans un arbre disparaissait au boot suivant.
#
# Les trois depots partent maintenant dans l'image AVEC leur .git : il y a donc
# un HEAD auquel se comparer, et la mise a jour redevient ce qu'elle doit etre -
# un fetch et une avance rapide. Rien n'est efface, rien n'est reclone, et une
# machine sans reseau garde exactement ce avec quoi elle a ete gravee.
cat > "$ROOTFS/usr/local/bin/osmo-update" <<'OSMOUPD'
#!/bin/bash
# ═══════════════════════════════════════════════════════════════════════
# osmo-update - met a jour, en place, les depots embarques dans l'image.
#
#   osmo-update              les trois depots
#   osmo-update qosmo        un seul (osmo-operator | osmo-egprs-web | qosmo | c54x_exe | grgsm_exe)
#   osmo-update --check      dit ce qui est en retard, n'ecrit rien
#   osmo-update --quiet      sans couleurs ni fioritures (journal, cron)
#   osmo-update --boot       mode demarrage : --quiet, journalise, sort toujours 0
#
# Ce qu'il ne fait PAS, deliberement : effacer un arbre, recloner un depot,
# installer un paquet. Une machine qui demarre n'a rien a aller chercher.
# ═══════════════════════════════════════════════════════════════════════
set -u

G='\033[0;32m'; Y='\033[1;33m'; R='\033[0;31m'; C='\033[0;36m'; B='\033[1m'; N='\033[0m'
CHECK=0; QUIET=0; BOOT=0
while [ $# -gt 0 ]; do
    case "$1" in
        --check)   CHECK=1 ;;
        --quiet)   QUIET=1 ;;
        --boot)    BOOT=1; QUIET=1 ;;
        -h|--help) sed -n '3,12p' "$0" | sed 's/^# \{0,1\}//'; exit 0 ;;
        -*)        echo "option inconnue : $1" >&2; exit 2 ;;
        *)         break ;;
    esac
    shift
done
[ "$QUIET" = "1" ] && { G=''; Y=''; R=''; C=''; B=''; N=''; }

# Au demarrage personne ne lit l'ecran : la sortie part dans le journal, et le
# code de retour ne doit jamais retarder ni bloquer multi-user.target.
if [ "$BOOT" = "1" ]; then
    exec >>/var/log/osmo-update.log 2>&1
    echo "===== osmo-update (boot) $(date '+%F %T') ====="
fi

[ "$(id -u)" -eq 0 ] || { echo "Root requis." >&2; exit 1; }

# nom|chemin - les chemins que cherchent deja start-native.sh, le dashboard et
# environnement/paths.env. En changer un ici ne deplacerait pas ceux qui les lisent.
REPOS="osmo-operator|/opt/GSM/osmo-operator
osmo-egprs-web|/opt/GSM/osmo-egprs-web
qosmo|/opt/GSM/qosmo
c54x_exe|/opt/GSM/c54x_exe
grgsm_exe|/opt/GSM/grgsm_exe"

WANT="${1:-}"
rc=0; web_moved=0; forks_moved=""

while IFS='|' read -r name dir; do
    [ -n "$name" ] || continue
    [ -z "$WANT" ] || [ "$WANT" = "$name" ] || continue
    found=1
    printf "  ${B}%-16s${N} ${C}%s${N}\n" "$name" "$dir"
```

```
    if [ ! -d "$dir" ]; then
        printf "    ${R}✗${N} absent - l'image ne le portait pas\n"; rc=1; continue
    fi
    if [ ! -d "$dir/.git" ]; then
        # On ne reclone pas par-dessus : ce serait effacer un arbre dont on ne
        # sait pas ce qu'il contient. On le dit, et on passe.
        printf "    ${Y}⚠${N} pas de depot (.git absent) - laisse tel quel\n"; rc=1; continue
    fi

    br="$(git -C "$dir" symbolic-ref --quiet --short HEAD 2>/dev/null)" || br=""
    [ -n "$br" ] || br=main

    # Un depot livre en --depth 1 est GREFFE : son unique commit n'a pas de
    # parent, donc rien de ce que le serveur renvoie n'a d'ancetre commun avec
    # lui. Le refetcher en --depth 1 garde cette propriete (et le depot reste
    # leger) ; un depot complet, lui, se fetch complet - sinon on lui ferait
    # perdre l'ancestralite qui permet justement l'avance rapide.
    if git -C "$dir" rev-parse --is-shallow-repository 2>/dev/null | grep -q true; then
        fetch_ok=$(git -C "$dir" fetch --depth 1 --quiet origin "$br" 2>/dev/null && echo 1)
    else
        fetch_ok=$(git -C "$dir" fetch --quiet origin "$br" 2>/dev/null && echo 1)
    fi
    if [ -z "${fetch_ok:-}" ]; then
        printf "    ${Y}⚠${N} fetch impossible (reseau ?) - copie locale conservee\n"; rc=1; continue
    fi

    local_h="$(git -C "$dir" rev-parse HEAD 2>/dev/null)"
    remote_h="$(git -C "$dir" rev-parse FETCH_HEAD 2>/dev/null)"
    if [ "$local_h" = "$remote_h" ]; then
        printf "    ${G}✓${N} deja a jour - %s\n" "$(git -C "$dir" log -1 --format='%h %s')"
        continue
    fi
    if [ "$CHECK" = "1" ]; then
        printf "    ${Y}→${N} en retard : %s -> %s\n" "${local_h:0:7}" "${remote_h:0:7}"
        continue
    fi

    # Trois cas, et un seul refus. Le refus porte sur le TRAVAIL LOCAL, jamais
    # sur l'historique : c'est la difference avec l'ancien "rm -rf puis clone",
    # qui effacait sans distinguer.
    if git -C "$dir" merge-base --is-ancestor HEAD FETCH_HEAD 2>/dev/null \
       && git -C "$dir" merge --ff-only FETCH_HEAD >/dev/null 2>&1; then
        # 1. Avance rapide : on est en retard sur la meme branche.
        printf "    ${G}✓${N} %s\n" "$(git -C "$dir" log -1 --format='%h %s')"
        [ "$name" = "osmo-egprs-web" ] && web_moved=1
        case "$name" in qosmo) forks_moved="$forks_moved $name" ;; esac
    elif [ -z "$(git -C "$dir" status --porcelain 2>/dev/null)" ]; then
        # 2. Pas d'ancetre commun (depot greffe par --depth 1) mais arbre propre :
        #    il n'y a rien a perdre, on aligne sur le serveur.
        if git -C "$dir" reset --hard FETCH_HEAD >/dev/null 2>&1; then
            printf "    ${G}✓${N} aligne sur origin/%s - %s\n" "$br" "$(git -C "$dir" log -1 --format='%h %s')"
            [ "$name" = "osmo-egprs-web" ] && web_moved=1
            case "$name" in qosmo) forks_moved="$forks_moved $name" ;; esac
        else
            printf "    ${Y}⚠${N} alignement impossible - arbre inchange\n"; rc=1
        fi
    else
        # 3. Des fichiers ont ete modifies ici : on ne touche a rien, on le dit.
        printf "    ${Y}⚠${N} modifications locales - rien n'a ete ecrase\n"
        printf "       a la main : ${C}git -C %s status${N}\n" "$dir"
        rc=1
    fi
done <<REPOEOF
$REPOS
REPOEOF
```

```
if [ -z "${found:-}" ]; then
    echo "depot inconnu : $WANT  (osmo-operator | osmo-egprs-web | qosmo | c54x_exe | grgsm_exe)" >&2
    exit 2
fi

# Un fork qui avance peut changer tools/qosmo-launch : le lanceur installe
# (/usr/local/bin/<fork>, celui que 40-qemu.sh appelle) est recompile sur place.
# C pur, libc seule, quelques secondes ; sans gcc on le dit et on laisse
# l'ancien binaire, qui reste valide (40-qemu.sh retombe sinon sur QEMU_BIN).
# [2026-09-09] LA VOIX, REPOSEE APRES CHAQUE FETCH. « git reset --hard
# FETCH_HEAD » ci-dessus efface deux reglages vitaux des forks :
# « io-tch-format rtp » dans cfgs/mobile_group1.cfg (sans lui le mobile parle
# la disposition Texas Instruments quand le pont code en RTP : voix robotisee)
# et CALYPSO_PULSE_LATENCY_MSEC (un tampon trop court fait remonter le flux
# PulseAudio du mobile 50 fois par seconde). Le pourquoi complet et le comment
# sont dans osmo-operator/scripts/voix-forks.sh - ici on ne fait que l appeler.
[ -x /opt/GSM/osmo-operator/scripts/voix-forks.sh ] \
    && /opt/GSM/osmo-operator/scripts/voix-forks.sh

for f in $forks_moved; do
    src="/opt/GSM/$f/tools/qosmo-launch"
    [ -f "$src/qosmo-launch.c" ] || continue
    if command -v gcc >/dev/null 2>&1 && make -s -C "$src" ALIAS="$f" install >/dev/null 2>&1; then
        printf "  ${G}✓${N} lanceur %s recompile (/usr/local/bin/%s)\n" "$f" "$f"
    else
        printf "  ${Y}⚠${N} lanceur %s non recompile (gcc/make ?) - l'ancien reste en place\n" "$f"
    fi
done

# Le dashboard tourne en service : un depot avance ne change rien tant que le
# demon fait tourner l'ancien server.js.
if [ "$web_moved" = "1" ]; then
    [ -f /opt/GSM/osmo-egprs-web/package.json ] && \
        (cd /opt/GSM/osmo-egprs-web && npm install --production >/dev/null 2>&1 || true)
    systemctl try-restart osmo-egprs-web >/dev/null 2>&1 || true
    printf "  ${G}✓${N} dashboard relance\n"
fi

[ "$BOOT" = "1" ] && exit 0
exit $rc
OSMOUPD
chmod +x "$ROOTFS/usr/local/bin/osmo-update"

# Au demarrage : apres le reseau, sans le bloquer. Type=oneshot + un ExecStart
# qui sort toujours 0 en mode --boot : une machine hors ligne demarre pareil.
cat > "$ROOTFS/etc/systemd/system/osmo-update.service" <<'EOF'
[Unit]
Description=osmo-operator - mise a jour des depots embarques (git, en place)
# [2026-09-04] PLUS D ATTENTE RESEAU. C etait Wants/After=network-online.target
# + systemd-networkd-wait-online.service, soit jusqu a 2 min de boot (mesure :
# systemd-analyze blame) pour une garantie dont cette unite n a pas besoin.
# network.target suffit : les interfaces sont configurees, la route par defaut
# peut arriver apres - ce service le supporte (voir son ExecStart).
After=network.target
[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/usr/local/bin/osmo-update --boot
[Install]
WantedBy=multi-user.target
EOF
chroot "$ROOTFS" systemctl enable osmo-update 2>/dev/null || true
echo -e "  ${GREEN}✓${NC} osmo-update (/usr/local/bin, + service au demarrage : git fetch, jamais de reclone)"

# ── QEMU_BIN apres le reclone : build/qemu-system-arm dans l'arbre qosmo ──
```

```
# [2026-08-27] Deux decisions justes, prises separement, se contredisent :
#
# L'arbre qosmo part maintenant entier - .git et build/ compris - donc
# QEMU_BIN est resolu des la gravure, et ce service n'a rien a faire. Il est la
# pour le seul cas ou l'arbre perdrait son build/ : quelqu'un qui le reclone a
# la main, ou qui remplace /opt/GSM/qosmo par un checkout frais. Sans build/,
# environnement/paths.env resout QEMU_BIN a un chemin inexistant et la pile
# s'arrete des le premier module :
#     [FAIL] Prerequisite checks (dépendances introuvables : QEMU_BIN)
# — alors que le binaire est la, dans /usr/local/bin.
#
# On recree donc le seul chemin que paths.env cherche, apres le reclone. Un lien
# symbolique, pas une copie : le binaire fait ~30 Mo et l'ISO tient en RAM.
# "build/" est la premiere ligne du .gitignore de qemu : le "git clean -fd" des
# synchronisations suivantes (wipe=0, incremental) ne l'efface pas - seul un
# clone frais le ferait, et ce service repasse a chaque demarrage.
cat > "$ROOTFS/usr/local/sbin/osmo-qemu-link.sh" <<'QLINK'
#!/bin/bash
# osmo-qemu-link.sh - rend QEMU_BIN resolvable apres le reclone de qosmo.
# Voir build-iso.sh, etape [6/9], pour le pourquoi.
set -u
SRC="${OSMO_QEMU_BIN:-/usr/local/bin/qemu-system-arm}"
TREE="${OSMO_QEMU_SRC:-/opt/GSM/qosmo}"
LNK="$TREE/build/qemu-system-arm"

# Pas de binaire (image inter-STP) ou pas d'arbre (reclone impossible, reseau
# coupe) : il n'y a rien a relier, et ce n'est pas ce service qui le dira.
[ -x "$SRC" ] || { echo "osmo-qemu-link: $SRC absent - rien a faire"; exit 0; }
[ -d "$TREE" ] || { echo "osmo-qemu-link: $TREE absent - rien a faire"; exit 0; }

# Un VRAI build compile sur place gagne toujours : on ne remplace qu'un lien
# (le notre) ou un chemin vide.
if [ -e "$LNK" ] && [ ! -L "$LNK" ]; then
    echo "osmo-qemu-link: $LNK est un vrai fichier - laisse tel quel"
    exit 0
fi

mkdir -p "$TREE/build"
ln -sfn "$SRC" "$LNK"
echo "osmo-qemu-link: $LNK -> $SRC"

# ── Lanceurs C (tools/qosmo-launch) : ce que 40-qemu.sh appelle ──────────────
# [2026-09-03] /usr/local/bin/<fork> est recompile s'il manque ou si la source
# de l'arbre est plus recente (reclone, osmo-update). [2026-09-25] Un seul
# arbre, qosmo ; le DSP (--dsp) est c54x_exe, hors QEMU : on dit s'il manque.
for fork in qosmo; do
    src="/opt/GSM/$fork/tools/qosmo-launch"
    [ -f "$src/qosmo-launch.c" ] || continue
    if [ ! -x "/usr/local/bin/$fork" ] || [ "$src/qosmo-launch.c" -nt "/usr/local/bin/$fork" ]; then
        if command -v gcc >/dev/null 2>&1 && make -s -C "$src" ALIAS="$fork" install >/dev/null 2>&1; then
            echo "osmo-qemu-link: lanceur /usr/local/bin/$fork (re)compile"
        else
           echo "osmo-qemu-link: lanceur $fork non compile (gcc/make absents ?) - 40-qemu.sh retombe sur QEMU_BIN"
        fi
    fi
done
if [ -d /opt/GSM/c54x_exe ] && [ ! -x /opt/GSM/c54x_exe/c54x_exe ]; then
    echo "osmo-qemu-link: c54x_exe non compile - --dsp indisponible (make -C /opt/GSM/c54x_exe)"
fi
if [ ! -f /opt/GSM/calypso_dsp.PROM0.bin ]; then
    echo "osmo-qemu-link: ROM DSP absente de /opt/GSM - --dsp indisponible (copie dans /opt/GSM/c54x_exe/rom)"
fi
QLINK
chmod +x "$ROOTFS/usr/local/sbin/osmo-qemu-link.sh"

cat > "$ROOTFS/etc/systemd/system/osmo-qemu-link.service" <<'EOF'
```

```
[Unit]
Description=osmo-operator - QEMU_BIN dans l'arbre qosmo + lanceur qosmo
After=local-fs.target
[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/usr/local/sbin/osmo-qemu-link.sh
[Install]
WantedBy=multi-user.target
EOF
chroot "$ROOTFS" systemctl enable osmo-qemu-link 2>/dev/null || true
echo -e "  ${GREEN}✓${NC} osmo-qemu-link (QEMU_BIN relie apres le reclone de qosmo ; lanceur qosmo recompile si
 ↪  besoin)"
  ×3
# Fin de module : `. fichier` rend le statut de sa DERNIERE commande, et
# build-iso.sh tourne sous set -e. Un module qui finirait par un test faux
# ("[ ... ] && { ...; }") arreterait tout, sans un mot. Toujours 0 ici.
true
```

3.206 /opt/GSM/osmo-operator/iso_modules/82-arm-natif.sh

13001 octets, 209 lignes → 209 lignes

```
#!/bin/bash
# iso_modules/82-arm-natif.sh - --arm : ce que l hote x86 ne peut pas produire
# Source par build-iso.sh, dans l ordre des numeros : meme shell, memes
# variables, memes fonctions. Ne s execute pas seul. `return` en tete de
# module = "rien a faire ici" (c est ainsi que --arm saute une etape).

# Sur la cible amd64 tout ceci est fait ailleurs, nativement (52-qemu, 71-debs-banc).
if [ "${ISO_ARCH:-amd64}" = "amd64" ]; then return 0; fi

# ═══════════════════════════════════════════════════════════════════════
#  Etape 8e : COMPILATION NATIVE DANS LE CHROOT arm64
# ═══════════════════════════════════════════════════════════════════════
# Trois choses que l ISO amd64 fabrique SUR L HOTE avec le gcc de l hote :
# toast (codec GSM 06.10), le lanceur C qosmo, c54x_exe/grgsm_exe, et les .deb du
# banc (build-debs.sh compile le lanceur et lit les binaires par ldd). Sur un
# hote x86 ca donnerait du x86 dans un rootfs arm64 - silencieusement, puisque
# aucune de ces etapes n est fatale. Ici tout tourne DANS le chroot, sous
# qemu-user : le gcc est celui du rootfs (build-essential, pose a l etape 8),
# le resultat est de l aarch64. Plus lent qu en natif, mais juste.
#
# Puis ce que le Pi exige et que l ISO ne connait pas : la partition firmware
# (/boot/firmware : noyau, initrd, device-trees, start4.elf, config.txt,
# cmdline.txt), posee par un hook qui rejouera a chaque mise a jour du noyau,
# et l agrandissement de la racine a la carte au premier demarrage.
echo -e "${GREEN}[8e/9] Compilation native dans le chroot ${ISO_ARCH} (toast, lanceurs, .deb du banc)...${NC}"
_arm_chroot() {
    chroot "$ROOTFS" env PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin \
        LC_ALL=C.UTF-8 LANG=C.UTF-8 LANGUAGE= DEBIAN_FRONTEND=noninteractive "$@"
}

# ── toast ──────────────────────────────────────────────────────────────
if [ "$ISO_ROLE" != "interstp" ]; then
    if [ -x "$ROOTFS/usr/local/bin/toast" ]; then
        echo -e "  ${GREEN}✓${NC} toast deja present (image)"
    elif _arm_chroot bash -c '
        set -e; cd /opt/GSM
        wget -qO gsm-1.0.24.tar.gz https://www.quut.com/gsm/gsm-1.0.24.tar.gz
        tar xzf gsm-1.0.24.tar.gz && rm -f gsm-1.0.24.tar.gz && cd gsm-1.0-pl24
        make >/dev/null 2>&1 || true
        [ -x bin/toast ] || make toast >/dev/null 2>&1
        for b in toast untoast tcat; do [ -e "bin/$b" ] && install -m755 "bin/$b" "/usr/local/bin/$b"; done
```

```
        [ -x /usr/local/bin/toast ]' >"$WORK/toast-arm.log" 2>&1; then
        echo -e "  ${GREEN}✓${NC} toast compile dans le chroot (sources : /opt/GSM/gsm-1.0-pl24)"
    else
        echo -e "  ${YELLOW}!${NC} toast non compile dans le chroot (voir $WORK/toast-arm.log) - on continue" >&2
    fi
fi

# ── Lanceur C qosmo ──────────────────────────────────────────────
if [ "$ISO_ROLE" != "interstp" ]; then
    for fork in qosmo; do
        if [ -x "$ROOTFS/usr/local/bin/$fork" ]; then
            echo -e "  ${GREEN}✓${NC} lanceur $fork deja present (image)"
        elif [ -f "$ROOTFS/opt/GSM/$fork/tools/qosmo-launch/qosmo-launch.c" ]; then
            if _arm_chroot make -s -C "/opt/GSM/$fork/tools/qosmo-launch" ALIAS="$fork" "$fork"
             ↪  >"$WORK/launch-$fork.log" 2>&1 \
              && _arm_chroot install -Dm755 "/opt/GSM/$fork/tools/qosmo-launch/$fork" "/usr/local/bin/$fork"; then
                echo -e "  ${GREEN}✓${NC} lanceur ${CYAN}/usr/local/bin/$fork${NC} compile dans le chroot"
            else
                echo -e "  ${YELLOW}!${NC} lanceur $fork : compilation echouee (voir $WORK/launch-$fork.log) -
                 ↪  run.sh appellera qemu-system-arm directement" >&2
            fi
        else
            echo -e "  ${YELLOW}!${NC} lanceur $fork : pas de source dans le rootfs" >&2
        fi
    done
    # [2026-09-25] c54x_exe et grgsm_exe : les binaires venus de l hote sont du
    # x86 ; on les recompile ici, sur les sources de /opt/GSM/qosmo du rootfs.
    for _e in c54x_exe grgsm_exe; do
        [ -f "$ROOTFS/opt/GSM/$_e/Makefile" ] || continue
        if _arm_chroot make -s -B -C "/opt/GSM/$_e" >"$WORK/make-$_e.log" 2>&1; then
            echo -e "  ${GREEN}✓${NC} ${CYAN}/opt/GSM/$_e/$_e${NC} compile dans le chroot"
        else
            echo -e "  ${YELLOW}!${NC} $_e : compilation echouee (voir $WORK/make-$_e.log)" >&2
        fi
    done
    # Le binaire arm64 doit etre executable ici, sous qemu-user : c est le test
    # qui aurait manque a une compilation croisee silencieuse.
    if [ -x "$ROOTFS/usr/local/bin/qemu-system-arm" ]; then
        if _arm_chroot /usr/local/bin/qemu-system-arm --version >/dev/null 2>&1; then
            echo -e "  ${GREEN}✓${NC} qemu-system-arm (aarch64) s execute dans le chroot"
        else
            echo -e "${RED}qemu-system-arm du rootfs ne s execute pas sous qemu-user : binaire ou bibliotheques d
             ↪  une autre architecture ?${NC}" >&2
            file "$ROOTFS/usr/local/bin/qemu-system-arm" >&2 || true
            exit 1
        fi
    fi
fi

# ── Les .deb du banc (etape 7c), fabriques dans le chroot ────────
if [ -x "$ROOTFS/opt/GSM/osmo-operator/packaging/build-debs.sh" ]; then
    install -d "$ROOTFS/var/cache/osmo-debs"
    if _arm_chroot env OSMO_OPERATOR_SRC=/opt/GSM/osmo-operator QOSMO_SRC=/opt/GSM/qosmo \
         FIRMWARE_SRC=/opt/GSM/firmware TMPDIR=/var/tmp \
         bash /opt/GSM/osmo-operator/packaging/build-debs.sh --out /var/cache/osmo-debs \
         >"$WORK/build-debs.log" 2>&1; then
        echo -e "  ${GREEN}✓${NC} $(ls "$ROOTFS"/var/cache/osmo-debs/*.deb 2>/dev/null | wc -l) paquets dans
         ↪  ${CYAN}/var/cache/osmo-debs${NC}"
    else
        echo -e "  ${YELLOW}!${NC} paquets .deb du banc non construits (voir $WORK/build-debs.log) - l image sort
         ↪  sans eux"
    fi
fi

# ═══════════════════════════════════════════════════════════════
#  La partition firmware du Pi : /boot/firmware, comme Armbian la peuple
```

```
# ═══════════════════════════════════════════════════════════════════
# Le Pi 4 ne demarre pas par GRUB ni par u-boot : son GPU lit une partition FAT
# (la premiere, etiquette RPICFG), y charge start4.elf, lit config.txt, puis le
# noyau et l initrd designes. Armbian monte cette partition sur /boot/firmware
# et la tient a jour par trois hooks de son paquet BSP (armbian-bsp-cli-rpi4b) :
#   /etc/kernel/postinst.d/z51-raspi-firmware   start4.elf, fixup4.dat... depuis
#                                               /usr/lib/linux-firmware-raspi
#   /etc/kernel/postinst.d/zzz-copy-new-files   vmlinuz, *.dtb, overlays/
#   /etc/initramfs/post-update.d/zzz-update-initramfs   initrd.img
# Le noyau s installe AVANT le BSP dans un meme apt : ses hooks n ont pas
# tourne. On les rejoue ici, comme le fait le build Armbian lui-meme
# (post_family_tweaks__populate_boot_firmware_directory), puis update-initramfs
# pour que l initrd parte par le troisieme hook. Toute mise a jour du noyau
# par apt sur la machine refera la meme chose, sans nous.
_KVER="$(ls "$ROOTFS"/boot/vmlinuz-*-bcm2711 2>/dev/null | sort -V | tail -1 | sed 's|.*/vmlinuz-||' || true)"
[ -n "$_KVER" ] || { echo -e "${RED}aucun noyau Armbian bcm2711 dans /boot du rootfs (linux-image-current-bcm2711
 ↪  non installe ?)${NC}" >&2; exit 1; }
install -d "$ROOTFS/boot/firmware"
for _h in /etc/kernel/postinst.d/z51-raspi-firmware /etc/kernel/postinst.d/zzz-copy-new-files; do
    [ -x "$ROOTFS$_h" ] || { echo -e "${RED}hook Armbian absent : $_h (armbian-bsp-cli-rpi4b-current non installe
     ↪  ?)${NC}" >&2; exit 1; }
    _arm_chroot "$_h" "$_KVER" >>"$WORK/rpi-firmware.log" 2>&1 \
        || { echo -e "${RED}$_h a echoue (voir $WORK/rpi-firmware.log)${NC}" >&2; tail -5 "$WORK/rpi-firmware.log"
         ↪  >&2; exit 1; }
done
_arm_chroot update-initramfs -u -k "$_KVER" >>"$WORK/rpi-firmware.log" 2>&1 \
    || { echo -e "${RED}update-initramfs -k $_KVER a echoue (voir $WORK/rpi-firmware.log)${NC}" >&2; exit 1; }
[ -f "$ROOTFS/boot/firmware/initrd.img" ] || cp -f "$ROOTFS/boot/initrd.img-$_KVER"
 ↪  "$ROOTFS/boot/firmware/initrd.img"

# config.txt et cmdline.txt : ceux qu Armbian ecrit pour rpi4b (bcm2711.conf,
# pre_umount_final_image__write_raspi_config/cmdline), a deux details pres :
# la console serie est activee (enable_uart=1 : un banc a souvent un cable
# serie, et l autologin root y est pose par 84-comptes), et loglevel n est pas
# force a 1 - on veut lire ce que dit le noyau. Ecrits une fois, jamais
# reecrits ensuite : modifiez librement sur la carte.
cat > "$ROOTFS/boot/firmware/config.txt" <<'CFG'
# osmo-operator sur Armbian - Raspberry Pi 4 (arm64)
# For more options and information see http://rptl.io/configtxt

# Enable audio (loads snd_bcm2835)
dtparam=audio=on

# Additional overlays and parameters are documented /boot/firmware/overlays/README

# Automatically load overlays for detected cameras / DSI displays
camera_auto_detect=1
display_auto_detect=1

# Automatically load initramfs files, if found
auto_initramfs=1

# Enable DRM VC4 V3D driver
dtoverlay=vc4-kms-v3d
max_framebuffers=2

# Don't have the firmware create an initial video= setting in cmdline.txt.
disable_fw_kms_setup=1

# Disable compensation for displays with overscan
disable_overscan=1

# Run as fast as firmware / board allows
arm_boost=1

# Console serie (serial0 -> ttyS0), autologin root (osmo-operator)
```

```
enable_uart=1

[cm4]
otg_mode=1

[all]
kernel=vmlinuz
initramfs initrd.img followkernel
arm_64bit=1
CFG
printf '%s\n' "console=serial0,115200 console=tty1 root=LABEL=armbi_root rootfstype=ext4 fsck.repair=yes rootwait
 ↪  logo.nologo cgroup_enable=cpuset cgroup_memory=1 cgroup_enable=memory" \
    > "$ROOTFS/boot/firmware/cmdline.txt"

for _f in vmlinuz initrd.img start4.elf fixup4.dat bcm2711-rpi-4-b.dtb config.txt cmdline.txt
 ↪  overlays/vc4-kms-v3d.dtbo; do
    [ -e "$ROOTFS/boot/firmware/$_f" ] || { echo -e "${RED}/boot/firmware/$_f absent apres les hooks Armbian (voir
     ↪  $WORK/rpi-firmware.log)${NC}" >&2; ls -la "$ROOTFS/boot/firmware" >&2; exit 1; }
done
echo -e "  ${GREEN}✓${NC} /boot/firmware : noyau ${CYAN}${_KVER}${NC}, $(ls "$ROOTFS/boot/firmware" | wc -l)
 ↪  entrees ($(du -sh "$ROOTFS/boot/firmware" | cut -f1))"

# ── Les services Armbian du premier demarrage ──────────────────────────
# armbian-resize-filesystem : etend la partition racine a la carte et
# resize2fs, une fois (c est lui, pas nous, qui sait le faire sur MBR ET GPT).
# armbian-firstrun : cles ssh regenerees, reglages materiels. Le BSP enable
# zram, hardware-optimize, led-state, monitor par son postinst ; ces deux-la
# non (le build Armbian le fait a la fabrication de l image, comme ici).
# armbian-ramlog (log2ram) : COUPE. Il tient /var/log dans 50 Mo de RAM et
# synchronise sur la carte : les journaux osmocom (logrotate a 32 Mo x 3), les
# pcap en anneau du tcpdump enveloppe (5 x 32 Mo) le rempliraient en une heure
# et les ecritures echoueraient sans un mot. Sur un banc, /var/log sur la carte.
chroot "$ROOTFS" systemctl enable armbian-resize-filesystem armbian-firstrun >/dev/null 2>&1 \
    || echo -e "  ${YELLOW}!${NC} armbian-resize-filesystem / armbian-firstrun : enable a echoue" >&2
install -d "$ROOTFS/etc/default"
if [ -f "$ROOTFS/etc/default/armbian-ramlog" ] || [ -f "$ROOTFS/etc/default/armbian-ramlog.dpkg-dist" ]; then
    cp -f "$ROOTFS/etc/default/armbian-ramlog.dpkg-dist" "$ROOTFS/etc/default/armbian-ramlog" 2>/dev/null || true
    sed -i 's/^ENABLED=.*/ENABLED=false/' "$ROOTFS/etc/default/armbian-ramlog"
    chroot "$ROOTFS" systemctl disable armbian-ramlog >/dev/null 2>&1 || true
fi
# Pas d assistant Armbian au premier login : root:osmo est deja pose, le
# marqueur /root/.not_logged_in_yet n existe que dans les images Armbian.
rm -f "$ROOTFS/root/.not_logged_in_yet"
echo -e "  ${GREEN}✓${NC} armbian-resize-filesystem et armbian-firstrun actives, armbian-ramlog coupe"

# Fin de module : `. fichier` rend le statut de sa DERNIERE commande, et
# build-iso.sh tourne sous set -e. Un module qui finirait par un test faux
# ("[ ... ] && { ...; }") arreterait tout, sans un mot. Toujours 0 ici.
true
```

3.207 /opt/GSM/osmo-operator/iso_modules/82-services.sh

32371 octets, 556 lignes → 556 lignes

```
#!/bin/bash
# iso_modules/82-services.sh - role, os-release, asterisk, dashboard, audio, pulse
# Source par build-iso.sh, dans l ordre des numeros : meme shell, memes
# variables, memes fonctions. Ne s execute pas seul. `return` en tete de
# module = "rien a faire ici" (c est ainsi que --arm saute une etape).

# ── Marqueur de role : ce que CETTE image est ──────────────────────────
# Lu par start-native.sh et par la banniere. Sans lui, deux ISO issues de la
# meme chaine sont indiscernables une fois demarrees - et on lance le mauvais
# script sur la mauvaise machine.
```

```
{
    printf '# /etc/osmo-role - genere par build-iso.sh\n'
    printf 'OSMO_ROLE=%s\n' "$ISO_ROLE"
    [ -n "$ISO_NODE" ] && printf 'OSMO_WAN_NODE=%s\n' "$ISO_NODE"
    # Pour la meme raison que le role : deux ISO issues de la meme chaine sont
    # indiscernables une fois demarrees. Celle-ci n'a pas les arbres de
    # compilation de /opt/GSM - autant que la machine puisse le dire elle-meme
    # quand quelque chose y sera cherche en vain.
    printf 'OSMO_LITE=%s\n' "$ISO_LITE"
    printf 'OSMO_HUB_IP=%s\n' "$ISO_HUB_IP"
} > "$ROOTFS/etc/osmo-role"

# ── /etc/os-release : l'image se nomme elle-meme ──────────────────────────
# [2026-08-27] Les trois ISO se presentaient toutes comme "Ubuntu 22.04 LTS".
# /etc/osmo-role dit deja ce que l'image est, mais lui ne s'affiche nulle part :
# la banniere de login, `hostnamectl`, les rapports de bug et le tableau de bord
# lisent os-release. Sur trois machines demarrees cote a cote, rien ne
# distinguait le hub d'un noeud, ni le noeud complet de sa variante elaguee.
#
# ON NE TOUCHE QU'AUX CHAMPS D'AFFICHAGE. ID, VERSION_ID, VERSION_CODENAME et
# UBUNTU_CODENAME restent ceux d'Ubuntu : apt, add-apt-repository et la moitie
# des scripts de paquets s'en servent pour choisir leurs depots. Renommer ID
# casserait l'image bien au-dela de sa banniere.
# VARIANT / VARIANT_ID sont les champs prevus par os-release(5) pour exactement
# cette distinction ; IMAGE_ID / IMAGE_VERSION, ceux prevus pour une image
# construite. On les remplit plutot que d'inventer des noms a nous.
case "$ISO_ROLE" in
    interstp) OS_NAME="osmo-operator interstp"; OS_VARIANT_ID="interstp" ;;
    *)        if [ "$ISO_LITE" = "1" ]; then
                  OS_NAME="osmo-operator-lite"; OS_VARIANT_ID="operator-lite"
              else
                  OS_NAME="osmo-operator";      OS_VARIANT_ID="operator"
              fi ;;
esac
# Le numero de noeud fait partie de l'identite quand il est fige dans l'image :
# c'est la seule chose qui distingue osmo-operator-1.iso de osmo-operator-2.iso.
OS_PRETTY="$OS_NAME"
[ -n "$ISO_NODE" ] && OS_PRETTY="$OS_NAME (noeud $ISO_NODE)"

# /etc/os-release est un lien vers ../usr/lib/os-release : on ecrit la cible et
# on laisse le lien tranquille - le remplacer par un fichier ferait diverger les
# deux chemins, que differents outils lisent indifferemment.
_osrel="$ROOTFS/usr/lib/os-release"
sed -i -e '/^NAME=/d' -e '/^PRETTY_NAME=/d' \
       -e '/^VARIANT=/d' -e '/^VARIANT_ID=/d' \
       -e '/^IMAGE_ID=/d' -e '/^IMAGE_VERSION=/d' \
       -e '/^HOME_URL=/d' -e '/^SUPPORT_URL=/d' -e '/^BUG_REPORT_URL=/d' \
       "$_osrel"
{
    printf 'NAME="%s"\n'          "$OS_NAME"
    printf 'PRETTY_NAME="%s"\n'   "$OS_PRETTY"
    printf 'VARIANT="%s"\n'       "$OS_PRETTY"
    printf 'VARIANT_ID="%s"\n'    "$OS_VARIANT_ID"
    printf 'IMAGE_ID="%s"\n'      "$OS_NAME"
    printf 'IMAGE_VERSION="%s"\n' "$LABEL"
    printf 'HOME_URL="https://github.com/bbaranoff/osmo-operator"\n'
    printf 'SUPPORT_URL="https://github.com/bbaranoff/osmo-operator"\n'
    printf 'BUG_REPORT_URL="https://github.com/bbaranoff/osmo-operator/issues"\n'
} >> "$_osrel"
echo -e "  ${GREEN}✓${NC} os-release : ${CYAN}${OS_PRETTY}${NC}"

# ── Asterisk : UN SEUL proprietaire, et c'est run_modules/19-asterisk.sh ────
# [2026-08-27] Le paquet asterisk installe son unite `enabled` : systemd
# demarrait donc un Asterisk au boot, pendant que 19-asterisk.sh lancait le sien
# en direct. Deux proprietaires pour un seul /etc/asterisk et une seule socket
# /var/run/asterisk/asterisk.ctl - la console finissait par ne plus repondre a
```

```
# personne et la pile s'arretait sur "console Asterisk : toujours pas pret".
# Le module ecarte deja systemd a chaque demarrage ; on le fait AUSSI ici pour
# que le premier boot d'une ISO neuve parte propre, sans le coup de balai.
chroot "$ROOTFS" systemctl disable asterisk >/dev/null 2>&1 || true
echo -e "  ${GREEN}✓${NC} asterisk.service desactive - le PBX est lance par ${CYAN}19-asterisk.sh${NC}"

if [ "$ISO_ROLE" = "interstp" ]; then
    # Le hub, lui, DOIT demarrer seul : les noeuds s'attachent a lui au boot, et
    # un hub qu'il faut lancer a la main transforme un demarrage simultane en
    # course perdue d'avance.
    cat > "$ROOTFS/etc/systemd/system/osmo-interstp.service" <<EOF
[Unit]
Description=osmo-operator inter-STP - hub SS7 du WAN (PC 0.0.0)
# [2026-09-04] PLUS D ATTENTE RESEAU. C etait Wants/After=network-online.target
# + systemd-networkd-wait-online.service, soit jusqu a 2 min de boot (mesure :
# systemd-analyze blame) pour une garantie dont cette unite n a pas besoin.
# network.target suffit : les interfaces sont configurees, la route par defaut
# peut arriver apres - ce service le supporte (voir son ExecStart).
After=network.target
[Service]
Type=forking
PIDFile=/run/osmo-interstp.pid
ExecStart=/opt/GSM/osmo-operator/launch/start-interstp.sh --ip ${ISO_HUB_IP}
ExecStop=/opt/GSM/osmo-operator/launch/start-interstp.sh --stop
Restart=on-failure
RestartSec=5
[Install]
WantedBy=multi-user.target
EOF
    chroot "$ROOTFS" systemctl enable osmo-interstp 2>/dev/null || true
    echo -e "  ${GREEN}✓${NC} osmo-interstp.service - hub lance au demarrage sur ${CYAN}${ISO_HUB_IP}${NC}"
fi


# Autologin root : pose plus bas (8d, getty@tty1.service.d/10-autologin-root.conf).

# ── Service web dashboard : DEJA POSE, NE PAS LE REECRIRE ──────────────
# [2026-08-31] Ici vivait un SECOND heredoc qui reecrivait
# osmo-egprs-web.service par-dessus celui que l'etape 6 venait de copier depuis
# services/ - et il en etait une version APPAUVRIE : ni TLS_CERT/TLS_KEY (donc
# pas de listener HTTPS, donc pas de contexte securise, donc pas de micro), ni
# PULSE_SERVER (donc pas de pont audio vers gsm_mic), ni CAP_IFACE=any (donc
# l'onglet trafic muet). L'ISO partait avec cette version-la ; ce n'est qu'au
# premier boot que install-web-service.sh recopiait la bonne par-dessus, et
# seulement s'il allait au bout - ce qu'il ne faisait pas, puisqu'il echouait
# justement sur le service qui refusait de demarrer.
#
# Deux fichiers pour une seule unite, c'est un de trop : la source unique est
# services/osmo-egprs-web.service, copiee a l'etape 6. On ne garde ici que
# l'activation.
# Le hub n'a ni VTY d'operateur a afficher ni radio a tracer : le tableau de
# bord n'aurait rien a montrer. On ne l'active pas.
[ "$ISO_ROLE" = "interstp" ] || chroot "$ROOTFS" systemctl enable osmo-egprs-web 2>/dev/null||true

# ── LA CONSOLE DES ABONNES 4G (Open5GS WebUI, :9999) ────────────────────
# [2026-09-10] Meme regle que le tableau de bord 2G : posee ET activee par
# defaut. Elle ne demarre pas pour autant sur une image sans WebUI construit -
# son ConditionPathExists (services/open5gs-webui.service) la saute alors, sans
# echec ni service rouge. Le hub n a pas de coeur 4G : on ne l active pas.
if [ "$ISO_ROLE" != "interstp" ]; then
    if [ -f "$DIR/services/open5gs-webui.service" ]; then
        install -m644 "$DIR/services/open5gs-webui.service" \
                "$ROOTFS/etc/systemd/system/open5gs-webui.service"
        mkdir -p "$ROOTFS/etc/systemd/system/multi-user.target.wants"
        chroot "$ROOTFS" systemctl enable open5gs-webui 2>/dev/null || \
            ln -sf /etc/systemd/system/open5gs-webui.service \
```

```
                "$ROOTFS/etc/systemd/system/multi-user.target.wants/open5gs-webui.service"
        echo -e "  ${GREEN}✓${NC} ${CYAN}open5gs-webui.service${NC} pose et active au boot (console abonnes 4G,
         ↪  :9999)"
    else
        echo -e "  ${RED}✗ services/open5gs-webui.service introuvable${NC}" >&2; exit 1
    fi
fi

# ── LE BANC EST POSE EN SERVICE, MAIS NE PART PAS TOUT SEUL ──────────────
# [2026-09-04] services/osmo-banc.service (standalone : start-native.sh) et
# services/osmo-multi.service (multi-operateur : start-multi.sh). Les deux
# tiennent leur pile dans tmux : `tmux attach -t calypso` depuis n'importe quel
# terminal de root reprend la main, exactement comme la fin de run.sh le
# faisait dans le terminal du lanceur. launch.sh et start-multi.sh detectent
# l'unite et passent par elle.
#
# [2026-09-05] AUCUNE DES DEUX N'EST PLUS ACTIVEE AU BOOT. osmo-banc l'etait
# depuis le debut, osmo-multi depuis la veille. Ce que ca donnait sur une
# machine fraichement installee : la pile radio entiere (QEMU/Calypso, le
# coeur, les conteneurs du multi) montait AVANT que l'operateur ait pu voir
# l'ecran de session - plusieurs minutes de demarrage, un banc deja lance sur
# une configuration qu'on n'avait pas encore choisie (noeud, --dsp, topologie),
# et rien a faire d'autre que l'arreter pour le relancer autrement.
# Un banc se DEMARRE : c'est le geste de l'operateur, pas un effet de bord du
# boot.
#   systemctl start osmo-banc          une session (ou l'icone du bureau)
#   systemctl enable --now osmo-banc   pour qu'il reparte a chaque demarrage
# Au build : OSMO_ISO_BANC=1 (ou --banc) et OSMO_ISO_MULTI=1 (ou --multi)
# rendent l'activation au boot, image par image.
# [2026-09-08] osmo-lte (services/osmo-lte.service : Open5GS + srsENB + srsUE,
# l icone « 4G du banc ») suit la meme regle : posee, OSMO_ISO_LTE=1 pour
# l activer au boot. Sans srsenb (interstp, arm64) sa Condition la saute.
if [ "$ISO_ROLE" != "interstp" ]; then
    # [2026-10-24] osmo-supplements : posee, PAS activee ; le job Calamares osmosupplements
    # l active dans la cible si une case de la page « Supplements » est cochee.
    for _u in osmo-banc osmo-multi osmo-lte osmo-supplements; do
        if [ -f "$DIR/services/$_u.service" ]; then
            install -m644 "$DIR/services/$_u.service" "$ROOTFS/etc/systemd/system/$_u.service"
        else
            echo -e "  ${RED}✗ services/$_u.service introuvable${NC}" >&2; exit 1
        fi
    done
    # `systemctl disable` NE SUFFIT PAS SEUL ICI. Le rootfs vient d'une image
    # qui a pu etre construite quand ces unites etaient activees : le lien
    # multi-user.target.wants/ y est deja, et un `disable` dans le chroot
    # echoue en silence quand systemd n'y tourne pas (2>/dev/null || true).
    # On efface donc le lien nous-memes, ce qui est de toute facon ce que
    # `disable` fait - le symetrique exact du `ln -sf` qui sert de repli a
    # l'`enable` dans l'autre branche.
    mkdir -p "$ROOTFS/etc/systemd/system/multi-user.target.wants"
    for _u in osmo-banc osmo-multi osmo-lte; do
        # osmo-banc -> OSMO_ISO_BANC, osmo-multi -> OSMO_ISO_MULTI ; ${!_var}
        # est l'expansion indirecte de bash (la valeur de la variable NOMMEE
        # par $_var), pas un eval.
        _var="OSMO_ISO_${_u#osmo-}"; _var="${_var^^}"
        if [ "${!_var:-0}" = "1" ]; then
            chroot "$ROOTFS" systemctl enable "$_u" 2>/dev/null || \
                ln -sf "/etc/systemd/system/$_u.service" \
                       "$ROOTFS/etc/systemd/system/multi-user.target.wants/$_u.service"
            echo -e "  ${GREEN}✓${NC} ${CYAN}$_u.service${NC} pose ET active au boot (${_var}=1)"
        else
            chroot "$ROOTFS" systemctl disable "$_u" 2>/dev/null || true
            rm -f "$ROOTFS/etc/systemd/system/multi-user.target.wants/$_u.service"
            echo -e "  ${GREEN}✓${NC} ${CYAN}$_u.service${NC} pose, ${BOLD}non active${NC} au boot ·
             ↪  ${CYAN}systemctl start $_u${NC} pour le lancer, ${CYAN}${_var}=1${NC} au build pour l'activer"
        fi
```

```
    done
fi

# ── Audio : PulseAudio systeme (sink gsm_audio) au boot ──────────────────
# Chaine : osmo-gapk → ALSA gsm_out → sink null gsm_audio → monitor → loopback
# → carte. system.pa autorise l'acces anonyme + prepare le sink ; le service
# lance le demon au boot (ensure_pulse de start-native.sh devient un no-op).
if [ -f "$ROOTFS/etc/pulse/system.pa" ]; then
    sed -i 's|^load-module module-native-protocol-unix.*|load-module module-native-protocol-unix auth-anonymous=1
     ↪  socket=/var/run/pulse/native|' \
        "$ROOTFS/etc/pulse/system.pa"
    # [2026-08-14] gsm_mic MANQUAIT ICI. Seul gsm_audio etait declare, alors que
    # lib/audio.sh traite les deux sinks comme SOLIDAIRES (GSM_SINKS) et que
    # configs/asound.conf fait pointer `pcm.gsm_in` sur `gsm_mic.monitor`.
    # Consequence mesuree dans la VM : `pactl list short sources` sans gsm_mic
    # → gapk_io n'initialise pas la capture et ABANDONNE LES DEUX SENS
    #   (pq_alsa.c:168 "Couldn't init ALSA device 'gsm_in'" puis
    #    gapk_io.c:468 "Failed to initialize GAPK I/O")
    # → appel parfaitement etabli et TOTALEMENT MUET. Les deux sinks doivent
    # etre declares ensemble, ici, comme le dit deja lib/audio.sh.
    for _s in "gsm_audio:GSM_Audio" "gsm_mic:GSM_Mic"; do
        _n="${_s%%:*}"; _d="${_s##*:}"
        grep -q "sink_name=${_n}\b" "$ROOTFS/etc/pulse/system.pa" || \
            echo "load-module module-null-sink sink_name=${_n} format=s16le rate=8000 channels=1
             ↪  sink_properties=device.description=${_d}" \
            >> "$ROOTFS/etc/pulse/system.pa"
    done
fi
# [2026-08-14] /etc/asound.conf N'ETAIT DEPLOYE NULLE PART sur l'ISO. Il l'est
# par ensure_pulse() de lib/audio.sh - mais lib/audio.sh n'est source par
# personne (son appelant annonce, run_modules/25-audio.sh, n'existe pas). Sans
# ce fichier les PCM `gsm_out`/`gsm_in` que `mobile` ouvre n'existent pas, donc
# la voix TCH n'atteint jamais gsm_audio. Verifie absent dans la VM au boot.
if [ -f "$DIR/configs/asound.conf" ]; then
    cp -f "$DIR/configs/asound.conf" "$ROOTFS/etc/asound.conf"
    echo -e "  ${GREEN}✓${NC} /etc/asound.conf (PCM gsm_out/gsm_in → sinks PulseAudio)"
fi
cat > "$ROOTFS/usr/local/sbin/osmo-audio-chain.sh" <<'ACHAIN'
#!/bin/bash
# osmo-audio-chain.sh - ferme la chaine audio locale apres le demarrage de
# PulseAudio. Appele en ExecStartPost par osmo-pulse.service.
#   1. /etc/asound.conf present (PCM gsm_out/gsm_in)
#   2. les DEUX null-sinks gsm_audio + gsm_mic charges
#   3. le module-loopback gsm_audio.monitor → carte son
# Sans (2), gapk_io abandonne LES DEUX SENS ; sans (3), la voix descendante est
# jetee par le null-sink. Toujours exit 0 : l'audio ne doit jamais empecher la
# pile de monter. AUDIO=0 ou AUDIO_LOCAL_LOOPBACK=0 neutralisent.
set -u
for r in /opt/GSM/osmo-operator /etc/osmocom/osmo-operator; do
    [ -x "$r/scripts/audio-chain.sh" ] && exec "$r/scripts/audio-chain.sh" "${1:-30}"
done
exit 0
ACHAIN
chmod +x "$ROOTFS/usr/local/sbin/osmo-audio-chain.sh"

cat > "$ROOTFS/usr/local/sbin/osmo-pulse-link.sh" <<'PLINK'
#!/bin/sh
# osmo-pulse-link.sh [uid] - rend le PulseAudio SYSTEME visible des applications
# qui cherchent un socket par utilisateur.
#
# En mode systeme, PulseAudio n'ecoute que sur /run/pulse/native. Les clients,
# eux, regardent $XDG_RUNTIME_DIR/pulse/native (soit /run/user/<uid>/pulse).
# Sans ce chemin, le navigateur ne voit AUCUN peripherique : « NotFoundError »
# sur le micro, pendant que pactl liste deux entrees en RUNNING.
#
# Toujours exit 0 : l'audio ne doit jamais empecher la pile de monter.
```

```
#
# ── APPELE DEUX FOIS, ET LES DEUX COMPTENT ──────────────────────────────
#   1. sans argument, par osmo-pulse.service (ExecStartPost) : les sessions
#      DEJA ouvertes au (re)demarrage du demon.
#   2. avec un uid, par le drop-in de user@.service (ExecStartPre=+, donc en
#      root) : CHAQUE session qui s ouvre ensuite.
#
# [2026-09-04] L appel (2) N EXISTAIT PAS, et c est ce qui rendait le systeme
# INSTALLE muet alors que la cle live avait du son. Au boot, osmo-pulse.service
# part bien avant le premier login : /run/user/ ne contient alors QUE les
# repertoires deja crees. Sur la cle, la session est root (autologin root) et
# /run/user/0 est la - le lien se posait. Sur le disque, l utilisateur est
# celui de l installeur (uid >= 1000) et son /run/user/<uid> n existe pas
# encore : aucun lien, la session tombait sur le socket de pipewire-pulse (ou
# sur rien du tout), et le bureau restait muet sans un message.
#
# ── LE PROPRIETAIRE DU SOCKET, PAS SEULEMENT LE CHEMIN ──────────────────
# [2026-08-30] Un client confine par AppArmor peut se voir autoriser le chemin
# avec le qualificateur `owner`, qui exige proprietaire du fichier == fsuid du
# processus. Le socket appartient a `pulse` (le compte du demon systeme) et la
# session tourne sous un autre uid : deux nombres differents, et la connexion
# est refusee alors que tout le reste marche - haut-parleurs audibles, carte en
# RUNNING, et un navigateur muet. C'est exactement l'ecart qu'on cherche
# pendant des heures cote « permission micro » ou « pilote son ».
#
# ⚠ Une telle regle porte sur le CHEMIN REEL : le lien symbolique ci-dessous
# ne masque rien, c'est bien /run/pulse/native qui est evalue.
#
# On donne donc le socket a l'uid de la session graphique. Un seul socket ne
# peut avoir qu'un proprietaire : c'est la DERNIERE session humaine ouverte qui
# l'obtient. Les comptes de service (gdm, uid < 1000) sont exclus - sans quoi
# gdm, dont le user@.service part AVANT celui de l operateur, gardait le socket.
# OSMO_PULSE_UID fige la valeur si une installation veut en decider autrement.
# Le demon, lui, continue de tourner en `pulse` : accept() ne demande pas
# d'etre proprietaire, et le mode srwxrwxrwx laisse tout le monde se connecter.
set -u

# Le lien, pour l uid demande ou pour toutes les sessions ouvertes.
if [ "$#" -ge 1 ] && [ -n "${1:-}" ]; then
    _dirs="/run/user/$1"
else
    _dirs=$(echo /run/user/*)
fi
for d in $_dirs; do
    [ -d "$d" ] || continue
    mkdir -p "$d/pulse" 2>/dev/null || continue
    # -f : le socket d un pipewire-pulse qui aurait survecu doit ceder la
    # place. Sans -f, ln echoue en silence et la session reste muette.
    ln -sfn /run/pulse/native "$d/pulse/native" 2>/dev/null || true
done

# Le proprietaire. Argument fourni : c est cette session-la, si c est un humain.
# Sinon on cherche la session graphique active, et root en dernier recours
# (c est le compte de la cle live).
PUID="${OSMO_PULSE_UID:-}"
if [ -z "$PUID" ] && [ "$#" -ge 1 ] && [ -n "${1:-}" ]; then
    [ "$1" -ge 1000 ] 2>/dev/null && PUID="$1"
    [ "$1" = "0" ] && PUID=0
fi
if [ -z "$PUID" ]; then
    PUID=$(loginctl list-sessions --no-legend 2>/dev/null \
           | awk '$2 == 0 || $2 >= 1000 {print $2; exit}')
fi
[ -n "$PUID" ] || PUID=0
[ -S /run/pulse/native ] && chown "$PUID" /run/pulse/native 2>/dev/null || true
exit 0
```

```
PLINK
chmod +x "$ROOTFS/usr/local/sbin/osmo-pulse-link.sh"

# ── LE LIEN EST POSE A CHAQUE OUVERTURE DE SESSION ──────────────────────────
# [2026-09-04] Voir l en-tete d osmo-pulse-link.sh : au boot, osmo-pulse.service
# ne peut PAS poser le lien d une session qui n existe pas encore. systemd-logind
# cree /run/user/<uid> puis demarre user@<uid>.service ; ce drop-in accroche le
# script juste apres, une fois par session.
#
# `ExecStartPre=+` : le `+` fait tourner la commande EN ROOT malgre le User= de
# user@.service. Sans lui, le script tournerait sous le compte de la session et
# le chown du socket (qui appartient a `pulse`) echouerait - c est justement la
# moitie du correctif AppArmor de 2026-08-30.
install -d "$ROOTFS/etc/systemd/system/user@.service.d"
cat > "$ROOTFS/etc/systemd/system/user@.service.d/10-osmo-pulse-link.conf" <<'UAT'
[Service]
ExecStartPre=+/usr/local/sbin/osmo-pulse-link.sh %i
UAT

# ── ET LE SERVEUR PAR DEFAUT, POUR LES CLIENTS QUI IGNORENT LE LIEN ─────────
# libpulse lit /etc/pulse/client.conf et son repertoire .d avant de deduire un
# chemin depuis XDG_RUNTIME_DIR. `default-server` est donc la garantie qui ne
# depend d aucune course, d aucun uid et d aucun ordre de demarrage : GNOME,
# Firefox, VLC et gapk arrivent au demon systeme meme si le lien manque.
#
# `autospawn = no` avec : sans lui, un client qui ne trouve pas de serveur
# lance SON PROPRE pulseaudio, lequel echoue a prendre les cartes (le demon
# systeme les tient) et laisse la session avec un serveur vide - un deuxieme
# faux "pas de son" par-dessus le premier.
install -d "$ROOTFS/etc/pulse/client.conf.d"
cat > "$ROOTFS/etc/pulse/client.conf.d/50-osmo-system-server.conf" <<'PCLIENT'
# Pose par build-iso.sh (iso_modules/82-services.sh).
# Le banc tourne un PulseAudio SYSTEME (osmo-pulse.service) : tous les clients,
# quel que soit leur uid, s y connectent. Sans cette ligne, chaque session
# cherche un serveur dans son propre /run/user/<uid> et n en trouve pas.
default-server = unix:/run/pulse/native
autospawn = no
PCLIENT
echo -e "  ${GREEN}✓${NC} audio session : ${CYAN}client.conf.d/50-osmo-system-server.conf${NC} + lien pose a
 ↪  chaque ouverture (user@.service)"

cat > "$ROOTFS/etc/systemd/system/osmo-pulse.service" <<'EOF'
[Unit]
Description=osmo-operator PulseAudio system daemon (GSM audio)
# sound.target seul ne garantit RIEN : c'est une cible passive, atteinte des que
# systemd a fini de traiter les regles udev deja connues - pas quand les cartes
# sont la. Au premier demarrage d'un disque fraichement installe, les modules
# snd_hda_* sont encore en cours de chargement quand cette unite part, et
# osmo-audio-chain.sh concluait "aucune sortie materielle - loopback local
# ignore" : l'appel montait, et il etait muet. module-udev-detect rattrape les
# cartes qui arrivent APRES le demarrage du demon (les sources et sinks ALSA
# apparaissent tout seuls), mais le loopback vers la carte, lui, n'est pose
# qu'une fois. On ne tire donc PAS systemd-udev-settle ici - il est obsolete et
# retarderait tout le boot pour un seul service : l'attente est dans
# scripts/audio-chain.sh, qui a deja un delai en parametre et ne fait patienter
# que lui-meme.
After=sound.target
Wants=sound.target
[Service]
# ── Type=notify ET --daemonize=no, ENSEMBLE OU PAS DU TOUT ──────────────────
# [2026-08-31] Ce couple valait "Type=forking" + "--daemonize=yes", et c'est ce
# qui rendait le son INDISPONIBLE sur le systeme installe alors qu'un
# "systemctl start osmo-pulse" a la main marchait a tous les coups.
#
# En Type=forking, systemd attend la mort du processus lance, puis DEVINE lequel
# des survivants du cgroup est le demon (GuessMainPID). pulseaudio --daemonize
```

```
# fait deux forks et laisse, le temps de la mise en place, ses fils de travail
# dans le cgroup : la devinette tombe sur un PID deja mort, systemd conclut que
# le service s'est termine, et son KillMode=control-group emporte le vrai demon
# qui venait juste de finir de charger ses modules. Le journal en garde la
# trace, et elle se lit a l'envers :
#     osmo-pulse.service: Deactivated successfully.
#     Started osmo-operator PulseAudio system daemon (GSM audio).
# "Deactivated" AVANT "Started" - le service est annonce demarre alors qu'il est
# deja mort. Rien n'echoue, rien n'est reessaye (Restart=on-failure ne voit
# qu'une sortie 0), et /run/pulse/native n'existe simplement jamais : pactl rend
# "Connection refused", et Firefox, qui cherche ce socket, enumere ZERO micro.
# Au demarrage manuel la course se joue autrement et la devinette tombe juste -
# d'ou un bug qui ne se reproduit qu'au boot, le seul moment ou personne ne
# regarde.
#
# La devinette disparait si le demon ne se detache pas : en --daemonize=no le
# processus lance EST le demon, son PID n'est plus a deviner. Et pulseaudio 15.x
# de jammy est lie a libsystemd : il appelle sd_notify(READY=1) une fois ses
# modules charges, ce que Type=notify attend. C'est aussi ce que fait l'unite
# fournie par le paquet (/usr/lib/systemd/user/pulseaudio.service), qui le dit
# dans son propre commentaire : "notify will only work if --daemonize=no".
#
# BENEFICE COLLATERAL, et il compte : en Type=notify, ExecStartPost ne part
# qu'apres READY=1. Avant, osmo-pulse-link.sh et osmo-audio-chain.sh couraient
# contre un demon qui n'avait pas fini d'ouvrir son socket.
Type=notify
ExecStart=/usr/bin/pulseaudio --system --daemonize=no --disallow-exit --exit-idle-time=-1 --log-target=journal
ExecStartPre=/bin/mkdir -p /var/log/osmocom /var/run/pulse
# ── LE SOCKET LA OU LES APPLICATIONS LE CHERCHENT ──────────────────────────
# PulseAudio tourne ici en mode SYSTEME : il n'ecoute que sur /run/pulse/native.
# Or une application cherche $XDG_RUNTIME_DIR/pulse/native, soit
# /run/user/<uid>/pulse/native. Sans ce chemin-la, elle ne trouve AUCUN serveur
# audio, enumere ZERO entree, et getUserMedia rend « NotFoundError — The object
# can not be found here ». Le diagnostic part alors invariablement sur une
# permission micro refusee, alors que la machine a deux entrees bien reelles.
# Un lien suffit ; il est pose par le demon lui-meme, donc il survit a un
# restart du service.
ExecStartPost=/usr/local/sbin/osmo-pulse-link.sh
# [2026-08-14] Sans ceci, gsm_audio (module-null-sink) n'a AUCUN consommateur :
# la voix descendante y est jetee par construction, la sortie ALSA reste
# SUSPENDED et l'appel est muet. Le loopback est le maillon qui manquait - il
# est pose ici, par le demon lui-meme, donc il survit a un restart du service.
# Non fatal (le script sort 0 quoi qu'il arrive) : l'audio ne doit jamais
# empecher la pile de monter. AUDIO_LOCAL_LOOPBACK=0 le neutralise.
# Passe par un wrapper /usr/local/sbin (meme patron que osmo-sms.sh) : une
# directive `ExecStartPost=/bin/sh -c "... \" ... \" ..."` avec guillemets
# imbriques est ACCEPTEE par `systemctl cat` mais rejetee par le parseur -
# `systemctl show -p ExecStartPost` revient alors VIDE et rien ne s'execute.
ExecStartPost=/usr/local/sbin/osmo-audio-chain.sh 30
Restart=on-failure
RestartSec=5
[Install]
WantedBy=multi-user.target
EOF
# Idem pour l'audio : le hub ne porte aucun appel, il route de la signalisation.
[ "$ISO_ROLE" = "interstp" ] || chroot "$ROOTFS" systemctl enable osmo-pulse 2>/dev/null||true

# Modules noyau
mkdir -p "$ROOTFS/etc/modules-load.d"
printf 'sctp\ntun\n' > "$ROOTFS/etc/modules-load.d/osmocom.conf"

# Variables d'environnement
cat > "$ROOTFS/etc/environment" <<'EOF'
PATH="/usr/local/sbin:/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin"
PKG_CONFIG_PATH="/usr/local/lib/pkgconfig"
LD_LIBRARY_PATH="/usr/local/lib"
```

```
EOF

# bashrc pour root
cat >> "$ROOTFS/root/.bashrc" <<'BASH'
# Active par defaut l'environnement python (gr-gsm + bridges) utilise par
# /opt/GSM/osmo-operator/start-native.sh. VIRTUAL_ENV_DISABLE_PROMPT pour garder le PS1.
export VIRTUAL_ENV_DISABLE_PROMPT=1
[ -f /root/.env/bin/activate ] && source /root/.env/bin/activate
# coeur.env est pose dans /etc/osmocom pour survivre au reclone de boot, mais
# environment/load.env ne va le chercher QUE dans son propre repertoire : sorti
# de l'arbre, personne ne le lit. On le charge donc ici, ou les deux arbres en
# heritent - l'arbre fige /opt/GSM/osmo-operator, qui n'embarque pas environment/, et
# l'arbre reclone /opt/GSM/osmo-operator, ou il ne survivrait pas. set -a : sans
# export, la valeur ne franchirait pas le fork vers start-native.sh. L'idiome
# ":=" du fichier laisse gagner N_MS=3 ./start-native.sh.
if [ -f /etc/osmocom/coeur.env ]; then set -a; . /etc/osmocom/coeur.env; set +a; fi
# Trois annonces designaient trois arbres differents. Le MOTD et le message de
# login pointent /opt/GSM/osmo-operator (l'arbre fige, present meme sans reseau) ;
# l'alias visait /opt/GSM/osmo-operator (l'arbre reclone au demarrage). Les deux
# fonctionnent, mais un utilisateur qui suit l'un puis l'autre ne travaille pas
# au meme endroit. On prend le premier chemin qui existe, dans l'ordre ou ils
# sont les plus complets.
alias osmo-lab='cd /opt/GSM/osmo-operator; ./start-native.sh'
alias osmo-web='systemctl status osmo-egprs-web'
alias osmo-status='/etc/osmocom/status.sh status'
export PATH="$HOME/.local/bin:$PATH"

### calypso-prompt ###
export PS1='\[\033[1;31m\]\u\[\033[0m\]@\[\033[1;34m\]\h\[\033[0m\]:\[\033[1;32m\]\w\[\033[0m\]☎ # '
### end calypso-prompt ###
BASH

# Message du jour - banniere couleur + boite alignee. Contenu de la boite en
# ASCII (pas de ←/e/• multi-octets) + padding printf => bords parfaitement
# alignes. Genere a chaud pour injecter les couleurs ANSI dans /etc/motd.
{
  B=$'\033[1;36m'; T=$'\033[1;37m'; G=$'\033[1;32m'; Y=$'\033[0;33m'; N=$'\033[0m'
  W=58
  printf '\n%b' "$B"
  cat <<'LOGO'
    ___  ___ _ __ ___   ___    ___  __ _ _ __  _ __ ___
   / _ \/ __| '_ ` _ \ / _ \  / _ \/ _` | '_ \| '__/ __|
  | (_) \__ \ | | | | | (_) ||  __/ (_| | |_) | |  \__ \
   \___/|___/_| |_| |_|\___/  \___|\__, | .__/|_|  |___/
                                   |___/|_|
LOGO
  printf '%b' "$N"
  printf "${B}  ╔"; printf '═%.0s' $(seq 1 $W); printf "╗${N}\n"
  printf "${B}  ║${N} ${T}%-*s${N} ${B}║${N}\n" $((W-2)) "GSM / EGPRS  Multi-PLMN  Live System"
  printf "${B}  ║${N} %-*s ${B}║${N}\n"         $((W-2)) "SS7/SIGTRAN  -  Osmocom  -  Calypso/QEMU"
  printf "${B}  ╠"; printf '═%.0s' $(seq 1 $W); printf "╣${N}\n"
  # Le chemin annonce ici est celui de l'arbre FIGE, comme le message de login
  # et comme le lien osmo-start-direct. Il nommait /opt/GSM/osmo-operator, que
  # osmo-update.service effacait et reclonait au demarrage : sans reseau au boot,
  # la premiere chose que lisait l'utilisateur designait un arbre qui pouvait ne
  # pas etre la. Le reclone a disparu, l'arbre fige reste - il ne depend de rien.
  printf "${B}  ║${N} ${G}%-*s${N} ${B}║${N}\n" $((W-2)) "/opt/GSM/osmo-operator/start-native.sh"
  printf "${B}  ║${N} %-*s ${B}║${N}\n"         $((W-2)) "    -> lance le lab Calypso/QEMU (A5/1)"
  printf "${B}  ║${N} ${G}%-*s${N} ${B}║${N}\n" $((W-2)) "Dashboard web  ->  http://<vm-ip>:8080"
  printf "${B}  ║${N} ${G}%-*s${N} ${B}║${N}\n" $((W-2)) "FFT spectres   ->  http://<vm-ip>:8081"
  printf "${B}  ║${N} ${Y}%-*s${N} ${B}║${N}\n" $((W-2)) "Wiki / docs        ->  pl4y.store"
  printf "${B}  ║${N} ${G}%-*s${N} ${B}║${N}\n" $((W-2)) "ssh root@<vm-ip>   -> mot de passe : osmo"
  printf "${B}  ║${N} ${Y}%-*s${N} ${B}║${N}\n" $((W-2)) "loadkeys fr   -> changer le clavier (apres boot)"
  printf "${B}  ║${N} ${G}%-*s${N} ${B}║${N}\n" $((W-2)) "osmo-update   -> met a jour les depots (git en place)"
  printf "${B}  ╚"; printf '═%.0s' $(seq 1 $W); printf "╝${N}\n\n"
} > "$ROOTFS/etc/motd"
```

```
# Mot de passe root = "osmo" (autologin console + login SSH). On NE vide PAS le
# mot de passe (sinon sshd refuse le login root).
echo 'root:osmo' | chroot "$ROOTFS" chpasswd 2>/dev/null || true

# ── POLITIQUE PAR DEFAUT : AUCUN SERVICE TELECOM AU BOOT (2026-10-23) ──────────
# Le preset services/80-osmo-operator.preset est pose dans le rootfs, et toute
# unite telecom deja activee (paquet asterisk/ofono qui s active a
# l installation, unites osmo-* amont, image reprise d un ancien build) est
# desactivee. Restent actives les seules demandes EXPLICITES du build :
# OSMO_ISO_BANC / _MULTI / _LTE=1, et le hub osmo-interstp, raison d etre du
# role interstp.
_garder=()
[ "$ISO_ROLE" = "interstp" ] && _garder+=(osmo-interstp)
[ "${OSMO_ISO_BANC:-0}" = 1 ]  && _garder+=(osmo-banc)
[ "${OSMO_ISO_MULTI:-0}" = 1 ] && _garder+=(osmo-multi)
[ "${OSMO_ISO_LTE:-0}" = 1 ]   && _garder+=(osmo-lte)
bash "$DIR/tools/osmo-telecom-boot.sh" --root "$ROOTFS" ${_garder[@]+--garder "${_garder[@]}"} || true

# Fin de module : `. fichier` rend le statut de sa DERNIERE commande, et
# build-iso.sh tourne sous set -e. Un module qui finirait par un test faux
# ("[ ... ] && { ...; }") arreterait tout, sans un mot. Toujours 0 ici.
true
```

3.208 /opt/GSM/osmo-operator/iso_modules/83-espace-ram.sh

19397 octets, 411 lignes → 411 lignes

```
#!/bin/bash
# iso_modules/83-espace-ram.sh - fstab, tmp, journal, logrotate, tcpdump, purge, ssh, clavier
# Source par build-iso.sh, dans l ordre des numeros : meme shell, memes
# variables, memes fonctions. Ne s execute pas seul. `return` en tete de
# module = "rien a faire ici" (c est ainsi que --arm saute une etape).

# ── Espace writable du live : /dev/shm + /tmp, en POURCENTAGE de la RAM ──────
# Le live boote en 'toram' → racine = overlay tmpfs (RAM). Les gros writers de la
# stack sont les cfiles I/Q dans /dev/shm (FFT/record, plusieurs centaines de Mo)
# et /tmp. systemd applique ces entrees au boot.
#
# POURQUOI PLUS 2 Go EN DUR
# Ces caps ne reservent rien, mais ils AUTORISENT : 2 + 2 Go, sur une machine
# ou le squashfs occupe deja ~2,5 Go de RAM et ou la racine elle-meme est un
# tmpfs, c'est plus que ce dont dispose une VM de 8 Go. Deux writers un peu
# gourmands suffisaient alors a saturer la memoire - et le symptome n'est pas
# "tmpfs plein" mais une machine exsangue : "No space left on device" sur la
# racine, puis un sshd qui n'arrive meme plus a envoyer sa banniere.
#
# En pourcentage, le plafond suit la taille de la machine : 20 % + 15 % laissent
# toujours les deux tiers de la RAM au squashfs, a la racine et aux processus.
# Une VM a 16 Go y gagne autant qu'une VM a 8 Go cesse de se noyer.
# Idempotent + anti-doublon : on purge d'abord toute entree tmpfs /tmp ou /dev/shm
# preexistante (y compris la variante 'nosuid,nodev' sans size=) et l'ancien
# commentaire de bloc, PUIS on (re)ecrit le bloc canonique size en pourcentage. Garantit
# exactement une entree /tmp et une entree /dev/shm dans le fstab du squashfs.
touch "$ROOTFS/etc/fstab"
sed -i -E \
    -e '/^[[:space:]]*tmpfs[[:space:]]+\/tmp[[:space:]]/d' \
    -e '/^[[:space:]]*tmpfs[[:space:]]+\/dev\/shm[[:space:]]/d' \
    -e '/^# osmo-operator live - espace writable/d' \
    "$ROOTFS/etc/fstab"
# /dev/shm : sizing via fstab (sans risque de doublon generateur).
cat >> "$ROOTFS/etc/fstab" <<'FSTAB'
# osmo-operator live - espace writable (cfiles I/Q FFT)
tmpfs   /dev/shm   tmpfs   defaults,nosuid,nodev,size=20%   0 0
```

```
FSTAB
# --arm : la racine et la partition firmware sont de VRAIES partitions de la
# carte SD (92-rpi-image les nomme ainsi). Le live n en a pas : live-boot lui
# donne sa racine. Etiquettes, pas UUID : l image est fabriquee sans monter.
if [ "${ISO_ARCH:-amd64}" = "arm64" ]; then
    # Les etiquettes d Armbian (armbi_root, RPICFG) : c est ce que son
    # armbian-resize-filesystem et sa doc attendent.
    sed -i -E -e '/^LABEL=armbi_root[[:space:]]/d' -e '/^LABEL=RPICFG[[:space:]]/d' "$ROOTFS/etc/fstab"
    cat >> "$ROOTFS/etc/fstab" <<'FSTAB'
LABEL=armbi_root    /              ext4    defaults,noatime,commit=120,errors=remount-ro   0 1
LABEL=RPICFG        /boot/firmware  vfat    defaults                                        0 2
FSTAB
    install -d "$ROOTFS/boot/firmware"
fi
# /tmp : PAS dans fstab. Une entree fstab /tmp entre en collision avec l'unite
# systemd tmp.mount -> "systemd-fstab-generator: tmp.mount already exists,
# Duplicate entry in /etc/fstab" (generateur en exit 1) ; et l'ancien update.sh
# la reinjectait au boot, ce qui obligeait a la retirer apres coup. On gere /tmp
# en natif systemd via un drop-in size=15% : une seule source, zero doublon -
# et plus rien, au demarrage, qui reecrive fstab.
mkdir -p "$ROOTFS/etc/systemd/system/tmp.mount.d"
cat > "$ROOTFS/etc/systemd/system/tmp.mount.d/size.conf" <<'EOF'
[Mount]
Options=mode=1777,strictatime,nosuid,nodev,size=15%
EOF
chroot "$ROOTFS" systemctl enable tmp.mount 2>/dev/null || true

# ── Ce qui remplit la RAM : les ECRITURES DE LA PILE ──────────────────────
# En 'toram' la racine est un overlay tmpfs : TOUT ce qui s'ecrit a l'execution
# reste en RAM, rien n'atteint un disque. Le squashfs y est deja recopie, /tmp
# et /dev/shm en reservent 2 Go chacun - le reste, quelques Go, est tout ce dont
# dispose la racine.
#
# Trois writers non bornes suffisaient a la remplir, et le symptome n'apparait
# qu'apres des heures : "No space left on device" sur une machine qui n'a
# pourtant aucun disque plein.
#   1. le journal systemd, sans plafond ;
#   2. /var/log/osmocom/*.log - la pile journalise en 'filter all 1', et mobile
#      tourne avec une vingtaine de categories de debug ;
#   3. les captures pcap GSMTAP, ecrites en continu et sans limite de taille.
# On les borne ici, dans l'image : un cap pose au build vaut pour toutes les VM,
# alors qu'un nettoyage manuel est a refaire apres chaque boot.

# 1. Journal : volatile (il est de toute facon perdu au reboot d'un live) et
#    plafonne. Sans RuntimeMaxUse, journald s'autorise 10 % de la RAM.
mkdir -p "$ROOTFS/etc/systemd/journald.conf.d"
# --arm : la racine est persistante, le journal aussi (Storage=persistent,
# plafonne) ; volatile jetterait les logs a chaque reboot d une carte SD.
if [ "${ISO_ARCH:-amd64}" = "arm64" ]; then
    cat > "$ROOTFS/etc/systemd/journald.conf.d/osmo-live.conf" <<'EOF'
[Journal]
Storage=persistent
SystemMaxUse=256M
RuntimeMaxUse=64M
EOF
else
cat > "$ROOTFS/etc/systemd/journald.conf.d/osmo-live.conf" <<'EOF'
# osmo-operator live : la racine est en RAM, le journal ne doit pas la manger.
[Journal]
Storage=volatile
RuntimeMaxUse=64M
RuntimeKeepFree=256M
EOF
fi

# 2. Logs Osmocom : rotation a la TAILLE, pas a la date - une pile bavarde
```

```
#    remplit en une heure ce qu'une rotation quotidienne ne verrait jamais.
#    copytruncate : les demons gardent leur descripteur ouvert ; sans lui la
#    rotation leur laisse un fichier supprime, et l'espace n'est pas rendu.
mkdir -p "$ROOTFS/etc/logrotate.d"
cat > "$ROOTFS/etc/logrotate.d/osmocom" <<'EOF'
/var/log/osmocom/*.log {
    size 32M
    rotate 3
    compress
    delaycompress
    missingok
    notifempty
    copytruncate
}
EOF
# logrotate.timer ne passe qu'une fois par jour : trop tard pour un tmpfs.
cat > "$ROOTFS/etc/systemd/system/osmo-logrotate.service" <<'EOF'
[Unit]
Description=osmo-operator - rotation des journaux Osmocom (racine en RAM)
[Service]
Type=oneshot
ExecStart=/usr/sbin/logrotate /etc/logrotate.d/osmocom --state /run/osmo-logrotate.state
EOF
cat > "$ROOTFS/etc/systemd/system/osmo-logrotate.timer" <<'EOF'
[Unit]
Description=osmo-operator - rotation des journaux Osmocom toutes les 15 min
[Timer]
OnBootSec=10min
OnUnitActiveSec=15min
[Install]
WantedBy=timers.target
EOF
chroot "$ROOTFS" systemctl enable osmo-logrotate.timer 2>/dev/null || true

# 3. Captures pcap : purge de celles de plus d'une heure. La capture GSMTAP
#    tourne en continu ; elle sert a regarder ce qui vient de se passer, pas a
#    constituer un historique - qu'aucun live ne pourrait de toute facon garder.
mkdir -p "$ROOTFS/etc/tmpfiles.d"
cat > "$ROOTFS/etc/tmpfiles.d/osmo-captures.conf" <<'EOF'
# osmo-operator : les captures vivent en RAM, on ne les garde pas plus d'une heure.
d /run/user/0/osmo-nitb/captures 0755 root root 1h
EOF

# 3bis. Le ring, plutot qu'un fichier qui gonfle
# Purger toutes les heures ne protege de rien : entre deux passages, UNE
# capture continue peut a elle seule remplir la RAM - et sur un lien charge,
# c'est l'affaire de quelques minutes, pas d'une nuit. Un fichier unique en -w
# croit sans limite ; -C <Mo> -W <n> lui substitue un ANNEAU de n fichiers qui
# se recyclent : la capture ne s'arrete jamais, l'empreinte reste bornee.
#
# Par un wrapper plutot qu'en corrigeant les appelants : la capture GSMTAP est
# lancee depuis le dashboard web (autre depot, clone au build) et depuis des
# outils qui ne vivent pas dans ce depot-ci. Un wrapper vaut pour tous, y
# compris ceux qu'on ajoutera. /usr/local/bin precede /usr/bin dans le PATH :
# l'appel "tcpdump" passe par lui sans que rien n'ait a etre reecrit.
#
# Il ne force rien quand l'appelant a deja choisi (-C ou -W presents), et sans
# -w il n'y a rien a borner.
cat > "$ROOTFS/usr/local/bin/tcpdump" <<'TCPDUMPEOF'
#!/bin/sh
# tcpdump - wrapper osmo-operator : impose un anneau aux captures sur fichier.
#
# La racine du live est un tmpfs : une capture non bornee finit par remplir la
# RAM, et l'erreur ("No space left on device") tombe des heures plus tard, sur
# une machine qui n'a pourtant aucun disque plein.
#
```

```
# DEUX PIEGES, DEUX PARADES
#  -Z root : avec -C/-W, tcpdump cree les membres suivants de l'anneau APRES
#            avoir abandonne ses privileges (utilisateur "tcpdump"). Sans -Z il
#            echoue des le premier : "Permission denied" - et aucune capture.
#            Sans -C il ouvrait le fichier AVANT, d'ou un fonctionnement qui ne
#            cassait qu'en ajoutant l'anneau.
#  lien    : avec -C/-W, tcpdump n'ecrit pas le nom demande mais numerote les
#            membres (capture.pcap0, .pcap1...). Le nom exact n'existe jamais,
#            et l'appelant qui l'attend conclut a un echec. On maintient donc
#            <nom exact> -> membre courant : la barriere le suit, qui ouvre le
#            fichier lit la capture en cours, et rien n'a a etre reecrit.
#
# Reglable : OSMO_PCAP_RING_MB (32), OSMO_PCAP_RING_FILES (5).
mkdir -p "$ROOTFS/usr/bin"
# [2026-09-04] SANS ECRASER LES BINAIRES DE DPKG. Ce `cp -a` recopiait TOUT
# /usr/local/bin - dont le wrapper tcpdump (docker cp de l'image en 50, puis
# 83-espace-ram.sh) - par-dessus /usr/bin/tcpdump, le vrai binaire. Le wrapper
# appelle /usr/bin/tcpdump : il s'appelait donc lui-meme a l'infini. Sur la
# machine demarree : "tcpdump n'a pas cree <pcap>", 1 module en echec, run.sh
# sort en 1, et osmo-banc.service (oneshot) tue toute la pile. On ne copie que
# ce qui n'existe pas deja dans /usr/bin (-n) ; les lanceurs qemu/qosmo, eux,
# n'ont pas d'homonyme dpkg et passent comme avant.
find "$ROOTFS/usr/local/bin" -mindepth 1 -maxdepth 1 ! -name tcpdump \
    -exec cp -an {} "$ROOTFS/usr/bin/" \; 2>/dev/null || true
 has_ring=0; has_z=0; wfile=""; next_is_w=0
for a in "$@"; do
    if [ "$next_is_w" = 1 ]; then wfile="$a"; next_is_w=0; continue; fi
    case "$a" in
        -w)            has_w=1; next_is_w=1 ;;
        -w?*)          has_w=1; wfile="${a#-w}" ;;
        -C|-C*|-W|-W*) has_ring=1 ;;
        -Z|-Z*)        has_z=1 ;;
    esac
done

# Rien a borner, ou l'appelant a deja choisi son anneau : on s'efface.
if [ "$has_w" != 1 ] || [ "$has_ring" = 1 ] || [ -z "$wfile" ]; then
    exec "$REAL" "$@"
fi

# Le veilleur du lien. Lance AVANT l'exec : apres, ce processus EST tcpdump.
# $$ reste le meme a travers l'exec, donc il suit exactement sa vie et s'arrete
# avec lui - aucun processus orphelin a nettoyer.
( ppid=$$
  n=0
  while [ "$n" -lt 300 ]; do
      [ -e "${wfile}0" ] && break
      kill -0 "$ppid" 2>/dev/null || exit 0
      sleep 0.2; n=$((n + 1))
  done
  while kill -0 "$ppid" 2>/dev/null; do
      newest=$(ls -t "${wfile}"[0-9]* 2>/dev/null | head -1)
      if [ -n "$newest" ] && [ "$(readlink "$wfile" 2>/dev/null)" != "$newest" ]; then
          ln -sfn "$newest" "$wfile"
      fi
      sleep 2
  done ) >/dev/null 2>&1 &

if [ "$has_z" = 1 ]; then
    exec "$REAL" -C "${OSMO_PCAP_RING_MB:-32}" -W "${OSMO_PCAP_RING_FILES:-5}" "$@"
fi
exec "$REAL" -Z root -C "${OSMO_PCAP_RING_MB:-32}" -W "${OSMO_PCAP_RING_FILES:-5}" "$@"
TCPDUMPEOF
chmod +x "$ROOTFS/usr/local/bin/tcpdump"
# Le vrai binaire doit etre reste un ELF : si un module l'a remplace par un
# script, l'image part avec un tcpdump qui boucle. On arrete la construction
```

```
# plutot que de decouvrir ca sur la machine demarree.
if [ "$(head -c 2 "$ROOTFS/usr/bin/tcpdump" 2>/dev/null)" = "#!" ]; then
    echo -e "  ${RED}✗ /usr/bin/tcpdump du rootfs est un script, pas le binaire dpkg${NC}" >&2
    exit 1
fi
echo -e "  ${GREEN}✓${NC} wrapper tcpdump (anneau pcap) dans /usr/local/bin, binaire dpkg intact dans /usr/bin"

# ── Purge complete a chaque relance ──────────────────────────────
# Les caps ci-dessus empechent la derive PENDANT une session ; celui-ci repart
# d'une racine propre A CHAQUE DEMARRAGE. Sur un live c'est sans perte : ces
# fichiers ne survivraient pas au reboot de toute facon. Avec persistance, en
# revanche, ils s'accumuleraient d'un boot a l'autre jusqu'a remplir le medium -
# c'est precisement le cas ou la purge devient indispensable.
#
# Avant la pile (Before=osmo-*.service) : purger APRES le demarrage effacerait
# les journaux de la session en cours, et le premier incident serait invisible.
cat > "$ROOTFS/usr/local/sbin/osmo-purge.sh" <<'PURGEEOF'
#!/bin/bash
# osmo-purge.sh - repart d'une racine propre. Appele au boot par osmo-purge.service.
# Ne touche NI aux configs (/etc/osmocom), NI aux bases (HLR) : seulement ce qui
# se regenere - journaux, captures, fichiers de travail.
set -u

purge_dir() {   # $1=repertoire  $2=motif
    [ -d "$1" ] || return 0
    find "$1" -maxdepth 1 -type f -name "$2" -delete 2>/dev/null || true
}

# Journaux de la pile
purge_dir /var/log/osmocom '*.log'
purge_dir /var/log/osmocom '*.log.*'
purge_dir /var/log/osmocom '*.gz'

# Captures pcap (GSMTAP et autres)
rm -rf /run/user/0/osmo-nitb/captures/* 2>/dev/null || true
purge_dir /var/log/osmocom '*.pcap'
find /tmp /var/tmp -maxdepth 2 -type f -name '*.pcap*' -delete 2>/dev/null || true

# Fichiers de travail : I/Q FFT (plusieurs centaines de Mo piece)
find /dev/shm -maxdepth 1 -type f \( -name '*.cfile' -o -name '*.raw' \) -delete 2>/dev/null || true
# Les MEMES fichiers HORS /dev/shm - ce sont eux qui ont rempli la RAM de la VM
# (4,6 Go mesures). Le mode pont ecrit /root/record.cfile et /root/record_ul.cfile
# en continu et empile /root/osmo-rec/*.cfile jusqu'a son propre plafond de 64 Go ;
# sur un live la racine EST un tmpfs, donc ce plafond n'en est pas un. Ne purger
# que /dev/shm laissait passer la totalite de ce qui se remplit vraiment.
find /root /tmp /var/tmp -maxdepth 2 -type f \( -name '*.cfile' -o -name '*.raw' \) -delete 2>/dev/null || true

# Repertoire d'execution du live, recree par la pile au demarrage
rm -rf /run/user/0/osmo-nitb/logs/* 2>/dev/null || true

# Journal systemd volatile
command -v journalctl >/dev/null 2>&1 && journalctl --rotate --vacuum-time=1s >/dev/null 2>&1

exit 0
PURGEEOF
chmod +x "$ROOTFS/usr/local/sbin/osmo-purge.sh"
cat > "$ROOTFS/etc/systemd/system/osmo-purge.service" <<'EOF'
[Unit]
Description=osmo-operator - purge des journaux, captures et fichiers de travail
DefaultDependencies=no
After=local-fs.target
Before=osmo-stp.service osmo-interstp.service osmo-msc.service osmo-bsc.service
Before=osmo-egprs-web.service shutdown.target
Conflicts=shutdown.target
[Service]
Type=oneshot
```

```
RemainAfterExit=yes
ExecStart=/usr/local/sbin/osmo-purge.sh
[Install]
WantedBy=multi-user.target
EOF
chroot "$ROOTFS" systemctl enable osmo-purge.service 2>/dev/null || true

# SSH : autorise le login root par mot de passe + active le service au boot.
if [ -f "$ROOTFS/etc/ssh/sshd_config" ]; then
    sed -i \
        -e 's/^#\?PermitRootLogin.*/PermitRootLogin yes/' \
        -e 's/^#\?PasswordAuthentication.*/PasswordAuthentication yes/' \
        "$ROOTFS/etc/ssh/sshd_config"
    grep -q '^PermitRootLogin yes' "$ROOTFS/etc/ssh/sshd_config" || \
        echo 'PermitRootLogin yes' >> "$ROOTFS/etc/ssh/sshd_config"
fi
chroot "$ROOTFS" systemctl enable ssh 2>/dev/null || true

# ── Clavier : fige DANS l'image, plus demande au premier boot ────────────
# Le choix se fait au debut de cette construction (voir "LES QUESTIONS"). Ici on
# ne fait que l'ecrire. L'ancienne version posait la question dans
# /etc/profile.d au premier login : chaque machine du banc s'arretait alors sur
# un menu, et une VM demarree sans console attendait une reponse que personne
# n'allait donner.
cat > "$ROOTFS/etc/default/keyboard" <<KBCONF
XKBMODEL="pc105"
XKBLAYOUT="${OSMO_ISO_KB}"
XKBVARIANT=""
XKBOPTIONS=""
BACKSPACE="guess"
KBCONF
chroot "$ROOTFS" setupcon --force 2>/dev/null || true
chroot "$ROOTFS" dpkg-reconfigure -f noninteractive keyboard-configuration 2>/dev/null || true

# Ce qui reste au login : le rappel, qui n'attend rien.
# La commande DEPEND du role : le hub n'a pas de coeur GSM a demarrer, et
# start-native.sh y chercherait un BSC, un MSC, une BTS qui n'existent pas.
# Lui dicter start-native.sh, c'est envoyer droit dans une erreur.
if [ "$ISO_ROLE" = "interstp" ]; then
    OSMO_START_HINT='Pour demarrer le hub SS7 : /opt/GSM/osmo-operator/launch/start-interstp.sh'
    OSMO_START_HINT2='  (etat des noeuds attaches : ./launch/start-interstp.sh --status)'
else
    OSMO_START_HINT='Pour demarrer la stack : /opt/GSM/osmo-operator/start-native.sh --node N'
    OSMO_START_HINT2='  (N de 1 a 9 : il fixe les point codes 1.<N>1.x du noeud)'
fi
cat > "$ROOTFS/etc/profile.d/01-osmo-disclaimer.sh" <<KBSCRIPT
#!/bin/bash
[ "\$(id -u)" -ne 0 ] && return 0
[ -n "\${OSMO_DISCLAIMER_SHOWN:-}" ] && return 0
export OSMO_DISCLAIMER_SHOWN=1
echo ""
echo -e "  \033[1;33mDisclaimer\033[0m - banc d'essai GSM/SS7 Osmocom."
echo -e "  A n'utiliser que sur un reseau radio \033[1mISOLE\033[0m (cage/attenuateur) ou"
echo -e "  sur une bande sous licence : emettre sur le spectre public est illegal."
echo -e "  Aucun service Osmocom n'est lance automatiquement sur cette ISO."
echo -e "  \033[1;33m${OSMO_START_HINT}\033[0m"
echo -e "  \033[0;36m${OSMO_START_HINT2}\033[0m"
echo -e "  clavier : \033[1;32m\$(awk -F\\" '/^XKBLAYOUT/{print \$2}' /etc/default/keyboard 2>/dev/null)\033[0m
 ↪  \033[0;36m(changer : osmo-keyboard)\033[0m"
echo ""
KBSCRIPT

chmod +x "$ROOTFS/etc/profile.d/01-osmo-disclaimer.sh"
rm -f "$ROOTFS/etc/profile.d/01-keyboard-setup.sh"

# Le choix du clavier reste offert - mais QUAND ON LE DEMANDE. C'est le meme
```

```
# menu qu'avant ; ce qui change, c'est qu'il ne s'interpose plus entre le login
# et le shell : une VM sans console ne peut plus rester bloquee dessus.
cat > "$ROOTFS/usr/local/bin/osmo-keyboard" <<'KBCMD'
#!/bin/bash
# osmo-keyboard - change la disposition clavier du systeme, a la demande.
[ "$(id -u)" -ne 0 ] && { echo "Root requis."; exit 1; }

if [ -n "$1" ]; then
    KB_LAYOUT="$1"
else
    echo ""
    echo -e "\033[1;36m== Configuration clavier ==\033[0m"
    echo "  1) fr    2) us    3) de    4) es    5) it"
    echo "  6) pt    7) gb    8) be    9) ch    0) autre"
    echo ""
    read -rp "  Choix [1] : " KB_CHOICE
    case "${KB_CHOICE:-1}" in
        1|"") KB_LAYOUT="fr" ;;
        2) KB_LAYOUT="us" ;;  3) KB_LAYOUT="de" ;;
        4) KB_LAYOUT="es" ;;  5) KB_LAYOUT="it" ;;
        6) KB_LAYOUT="pt" ;;  7) KB_LAYOUT="gb" ;;
        8) KB_LAYOUT="be" ;;  9) KB_LAYOUT="ch" ;;
        0) read -rp "  Layout (fr, us, ru, ar...) : " KB_LAYOUT
           KB_LAYOUT="${KB_LAYOUT:-us}" ;;
        *) KB_LAYOUT="fr" ;;
    esac
fi

loadkeys "$KB_LAYOUT" 2>/dev/null || true
cat > /etc/default/keyboard <<KBCONF
XKBMODEL="pc105"
XKBLAYOUT="${KB_LAYOUT}"
XKBVARIANT=""
XKBOPTIONS=""
BACKSPACE="guess"
KBCONF
setupcon --force 2>/dev/null || true
dpkg-reconfigure -f noninteractive keyboard-configuration 2>/dev/null || true
echo -e "  \033[1;32mClavier : ${KB_LAYOUT}\033[0m   (sans persistance, revient au reboot)"
KBCMD
chmod +x "$ROOTFS/usr/local/bin/osmo-keyboard"

# Fin de module : `. fichier` rend le statut de sa DERNIERE commande, et
# build-iso.sh tourne sous set -e. Un module qui finirait par un test faux
# ("[ ... ] && { ...; }") arreterait tout, sans un mot. Toujours 0 ici.
true
```

3.209 /opt/GSM/osmo-operator/iso_modules/84-comptes.sh

7849 octets, 122 lignes → 122 lignes

```
#!/bin/bash
# iso_modules/84-comptes.sh - etape 8d : comptes, autologin console/bureau, linphone
# Source par build-iso.sh, dans l ordre des numeros : meme shell, memes
# variables, memes fonctions. Ne s execute pas seul. `return` en tete de
# module = "rien a faire ici" (c est ainsi que --arm saute une etape).

# ═══════════════════════════════════════════════════════════════════
# Etape 8d : comptes, session, installeur - TOUT CE QUI PORTE DES APOSTROPHES
# ═══════════════════════════════════════════════════════════════════
# Ecrit ICI et pas dans le chroot de l etape 8 : ce chroot est un bash -c en
# QUOTES SIMPLES. Une apostrophe de plus et tout ce qui suit change de sens -
# et l erreur ne se voit qu au build suivant, sur une ligne sans rapport. Les
# configurations Calamares, les sudoers et les unites systemd en sont pleins.
```

```
# On ecrit donc dans "$ROOTFS" directement, avec le quoting normal du script.
echo -e "${GREEN}[8d/9] Comptes, session et installeur...${NC}"

# ── LE MODELE DE COMPTES ──────────────────────────────────────────
# root est le compte de TRAVAIL : le banc se pilote en root (start-native.sh,
# les VTY, tcpdump, les netns), et tout le depot le suppose. La session s ouvre
# donc sur root, et les terminaux qu on y ouvre sont root sans rien demander.
#
# osmocom est un SECOND compte, non privilegie, sudoer - le bac a sable pour ce
# qui n a pas besoin des pleins pouvoirs : un navigateur, une session de bureau
# ordinaire. On y va EXPLICITEMENT (se deconnecter, le choisir dans GDM, ou
# "su - osmocom"), jamais par defaut.
#
# CE QUE CE N EST PLUS. Ce compte a longtemps ete un ALIAS D UID 0
# (usermod -o -u 0 osmocom) : un compte qui portait un nom d utilisateur
# ordinaire et les pleins pouvoirs, ce qui est le pire des deux mondes - on
# croit travailler sans privileges et on est root. C etait aussi la raison pour
# laquelle Chromium refusait de demarrer avec son bac a sable. Ici, osmocom est
# un vrai compte non privilegie, avec son propre UID.
chroot "$ROOTFS" env LC_ALL=C.UTF-8 LANGUAGE= bash -c "
set -u
id -u osmocom >/dev/null 2>&1 || useradd -m -s /bin/bash -c 'Compte osmocom (non privilegie)' osmocom
echo 'osmocom:osmo' | chpasswd
echo 'root:osmo'    | chpasswd
passwd -u root >/dev/null 2>&1 || true
for g in sudo adm dialout audio video plugdev netdev cdrom; do
    getent group \$g >/dev/null && usermod -aG \$g osmocom
done
" 2>/dev/null || echo -e "  ${YELLOW}!${NC} creation des comptes incomplete"
echo -e "  ${GREEN}✓${NC} comptes : ${CYAN}root${NC} (travail, mdp osmo) et ${CYAN}osmocom${NC} (non privilegie,
 ↪  sudoer, mdp osmo)"

# ── LA CONSOLE OUVRE SUR ROOT ─────────────────────────────────────
# Sur la cle live il n y a personne a authentifier : demander un mot de passe
# sur tty1 ne protege rien (qui tient le medium tient la machine) et empeche
# juste de travailler. --noclear garde a l ecran les messages du demarrage,
# qui sont souvent la seule trace d un module qui a echoue.
install -d "$ROOTFS/etc/systemd/system/getty@tty1.service.d"
cat > "$ROOTFS/etc/systemd/system/getty@tty1.service.d/10-autologin-root.conf" <<'GETTY'
[Service]
ExecStart=
ExecStart=-/sbin/agetty --autologin root --noclear %I $TERM
GETTY
install -d "$ROOTFS/etc/systemd/system/serial-getty@ttyS0.service.d"
cat > "$ROOTFS/etc/systemd/system/serial-getty@ttyS0.service.d/10-autologin-root.conf" <<'GETTYS'
[Service]
ExecStart=
ExecStart=-/sbin/agetty --autologin root --keep-baud 115200,57600,38400,9600 %I $TERM
GETTYS
# Le Pi 4 : console=serial0 -> ttyS0 (mini-UART, enable_uart=1) ou ttyAMA0
# (PL011) selon l overlay ; les deux unites recoivent le meme drop-in.
if [ "${ISO_ARCH:-amd64}" = "arm64" ]; then
    install -d "$ROOTFS/etc/systemd/system/serial-getty@ttyAMA0.service.d"
    cp "$ROOTFS/etc/systemd/system/serial-getty@ttyS0.service.d/10-autologin-root.conf" \
       "$ROOTFS/etc/systemd/system/serial-getty@ttyAMA0.service.d/10-autologin-root.conf"
fi
echo -e "  ${GREEN}✓${NC} tty1 et console serie : ouverture automatique sur ${CYAN}root${NC}"

# ── LE BUREAU AUSSI ───────────────────────────────────────────────
# L etape 8 pose deja AutomaticLogin=root dans /etc/gdm3/custom.conf. On le
# reecrit ici sans condition : cette etape tourne meme quand ISO_DESKTOP vaut 0
# (le fichier est alors sans effet, GDM n est pas installe), et surtout elle
# garantit que le reglage est le meme des deux cotes - la cle live et le
# systeme installe, ou c est shellprocess-osmo.conf qui l ecrit.
if [ -d "$ROOTFS/etc/gdm3" ] || [ "${ISO_DESKTOP:-0}" = "1" ]; then
    install -d "$ROOTFS/etc/gdm3"
```

```
    cat > "$ROOTFS/etc/gdm3/custom.conf" <<'GDMCONF'
[daemon]
AutomaticLoginEnable=true
AutomaticLogin=root
# X11 impose : sous VirtualBox/QEMU, la session Wayland de GNOME 42 tombe sur
# le pilote llvmpipe et rend un bureau inutilisable, quand elle demarre.
WaylandEnable=false
GDMCONF
    sed -i "/pam_succeed_if.so user != root quiet_success/s/^/#/" \
        "$ROOTFS/etc/pam.d/gdm-password" "$ROOTFS/etc/pam.d/gdm-autologin" 2>/dev/null || true
    install -d "$ROOTFS/root/.config" "$ROOTFS/home/osmocom/.config"
    echo yes > "$ROOTFS/root/.config/gnome-initial-setup-done"
    echo yes > "$ROOTFS/home/osmocom/.config/gnome-initial-setup-done"
    chroot "$ROOTFS" chown -R osmocom:osmocom /home/osmocom 2>/dev/null || true
    echo -e "  ${GREEN}✓${NC} session graphique : ouverture automatique sur ${CYAN}root${NC} (osmocom au choix,
     ↪  apres deconnexion)"
fi

# ── LINPHONE : COMPTE PRE-PROVISIONNE ──────────────────────────────
# Place ICI, et pas ailleurs : le compte doit exister AVANT le premier
# lancement du client. Linphone ne relit pas linphonerc a chaud - l instance
# qui tourne garde son etat en memoire et REECRIT le fichier en sortant. Un
# compte pose pendant que Linphone tourne n emet donc jamais de REGISTER, et
# le symptome est muet des DEUX cotes : cote client rien dans les journaux,
# cote Asterisk endpoint "Unavailable" sans le moindre 401 - puisque aucun
# paquet n arrive. Diagnostic du 2026-08-31, une capture de 90 s pour le voir.
# Le detail des deux pieges (relecture a chaud, publish de presence) est en
# tete de configs/linphonerc.
if [ "${ISO_DESKTOP:-0}" = "1" ] && [ -f "$DIR/configs/linphonerc" ]; then
    # Les DEUX comptes plus /etc/skel : la session s ouvre sur root (gdm3
    # ci-dessus), osmocom reste disponible apres deconnexion, et skel couvre
    # les comptes crees par l installeur sur le systeme cible.
    for _lh in "$ROOTFS/root" "$ROOTFS/home/osmocom" "$ROOTFS/etc/skel"; do
        install -d "$_lh/.config/linphone"
        cp -a "$DIR/configs/linphonerc" "$_lh/.config/linphone/linphonerc"
    done
    chroot "$ROOTFS" chown -R osmocom:osmocom /home/osmocom 2>/dev/null || true
    echo -e "  ${GREEN}✓${NC} linphone : compte ${CYAN}linphone_A${NC} pre-provisionne (poste ${CYAN}100${NC},
     ↪  UDP vers 127.0.0.1:5060)"
fi


# Fin de module : `. fichier` rend le statut de sa DERNIERE commande, et
# build-iso.sh tourne sous set -e. Un module qui finirait par un test faux
# ("[ ... ] && { ...; }") arreterait tout, sans un mot. Toujours 0 ici.
true
```

3.210 /opt/GSM/osmo-operator/iso_modules/85-extensions-gnome.sh

799 octets, 15 lignes → 15 lignes

```
#!/bin/bash
# iso_modules/85-extensions-gnome.sh - extensions GNOME Shell actives par defaut
# Source par build-iso.sh, dans l ordre des numeros. `return` en tete = rien a faire.
#
# [2026-10-23] EasyScreenCast, pose et active pour tous les comptes par
# tools/gnome-extensions-install.sh (zip de extensions.gnome.org, sha256
# verifie). chrome-gnome-shell vient d apt (liste du bureau, 80-chroot.sh).
# Bureau seulement (--desktop), amd64. Un echec (reseau) n arrete pas le build.
if [ "${ISO_ARCH:-amd64}" = "arm64" ] || [ "${ISO_DESKTOP:-0}" != "1" ]; then return 0; fi

if _EXT_LOG=$("$DIR/tools/gnome-extensions-install.sh" "$ROOTFS" 2>&1); then
    echo -e "  ${GREEN}✓${NC} ${_EXT_LOG##*$'\n'}"
else
```

```
    echo -e "  ${YELLOW}!${NC} extensions GNOME non posees :\n$_EXT_LOG" | tail -4 >&2
fi
```

3.211 /opt/GSM/osmo-operator/iso_modules/85-glava.sh

1465 octets, 27 lignes → 27 lignes

```
#!/bin/bash
# iso_modules/85-glava.sh - le visualiseur audio du bureau (GLava + NCS Spectrum)
# Source par build-iso.sh, dans l ordre des numeros. `return` en tete = rien a faire.
#
# [2026-10-23] GLava n a pas de paquet dans noble : tools/glava-build.sh le
# compile sur l HOTE (meme Ubuntu que l image) avec le patch du module NCS, et
# l installe dans le rootfs par DESTDIR - binaire, shaders dans /etc/xdg/glava,
# lanceur /usr/local/bin/osmo-glava, service utilisateur glava.service. Les
# bibliotheques d execution (libpulse0, libx11-6, libxext6, libxrender1,
# libxcomposite1, libgl1) viennent deja avec le bureau GNOME.
#
# Bureau seulement (--desktop), amd64. Desactivable : OSMO_ISO_GLAVA=0.
# Un echec (reseau, paquets -dev absents de l hote) n arrete pas le build :
# l ISO sort sans visualiseur, avec un avertissement.
if [ "${ISO_ARCH:-amd64}" = "arm64" ] || [ "${ISO_DESKTOP:-0}" != "1" ]; then return 0; fi
if [ "${OSMO_ISO_GLAVA:-1}" != "1" ]; then
    echo -e "  ${YELLOW}!${NC} GLava desactive (OSMO_ISO_GLAVA=0)"
    return 0
fi

if _GLV_LOG=$("$DIR/tools/glava-build.sh" "$ROOTFS" 2>&1); then
    echo -e "  ${GREEN}✓${NC} ${_GLV_LOG##*$'\n'}"
    _GLV_MANQUE=$(chroot "$ROOTFS" ldd /usr/bin/glava 2>/dev/null | grep 'not found' || true)
    [ -z "$_GLV_MANQUE" ] || echo -e "  ${YELLOW}!${NC} GLava : bibliotheques absentes du rootfs :\n$_GLV_MANQUE"
     ↪  >&2
else
    echo -e "  ${YELLOW}!${NC} GLava non installe :\n$_GLV_LOG" | tail -4 >&2
fi
```

3.212 /opt/GSM/osmo-operator/iso_modules/85-installeur-bureau.sh

59592 octets, 1047 lignes → 1047 lignes

```
#!/bin/bash
# iso_modules/85-installeur-bureau.sh - calamares, osmo-install, bureau (icones, lanceurs)
# Source par build-iso.sh, dans l ordre des numeros : meme shell, memes
# variables, memes fonctions. Ne s execute pas seul. `return` en tete de
# module = "rien a faire ici" (c est ainsi que --arm saute une etape).

# --arm : calamares, osmo-install et le bureau : pas de --desktop en arm64, rien a poser.
if [ "${ISO_ARCH:-amd64}" = "arm64" ]; then return 0; fi

# ── L INSTALLEUR ──────────────────────────────────────────────────────
# La configuration vit dans le depot (installer/calamares/) plutot qu en
# heredocs ici : elle est relisible, versionnee, et validable hors build
# (c est du YAML, "python3 -c import yaml" suffit a la verifier).
_CAL_SRC="$DIR/installer/calamares"
if [ "${ISO_DESKTOP:-0}" = "1" ] && [ -d "$_CAL_SRC" ]; then
    install -d "$ROOTFS/etc/calamares"
    cp -a "$_CAL_SRC/settings.conf" "$ROOTFS/etc/calamares/"
    cp -a "$_CAL_SRC/modules"       "$ROOTFS/etc/calamares/"
    cp -a "$_CAL_SRC/branding"      "$ROOTFS/etc/calamares/"
    # [2026-10-24] Le job python osmosupplements (page « Supplements ») : dans le repertoire
    # des modules SYSTEME, celui de modules-search que Calamares parcourt a coup sur.
```

```
    _CAL_MODS="$ROOTFS/usr/lib/$(chroot "$ROOTFS" dpkg-architecture -qDEB_HOST_MULTIARCH 2>/dev/null || echo
     ↪  x86_64-linux-gnu)/calamares/modules"
    install -d "$_CAL_MODS/osmosupplements"
    install -m644 "$_CAL_SRC/jobs/osmosupplements/module.desc" "$_CAL_SRC/jobs/osmosupplements/main.py" \
        "$_CAL_MODS/osmosupplements/"

    # Les images de l habillage : on reprend le fond d ecran deja fige au build
    # plutot que d ajouter des binaires au depot. Calamares les met a l echelle.
    _WPI="$DIR/configs/gsm-lab-wallpaper.png"
    if [ -f "$_WPI" ]; then
        cp "$_WPI" "$ROOTFS/etc/calamares/branding/osmo/logo.png"
    else
        # Sans image, Calamares journalise une erreur par ecran. On retire les
        # cles plutot que de laisser pointer vers des fichiers absents.
        sed -i '/^images:/,/^slideshow:/{/productLogo\|productIcon/d}' \
            "$ROOTFS/etc/calamares/branding/osmo/branding.desc"
    fi
    # [2026-10-24] L accueil et la premiere diapo : le fond "Telecom Operator /
    # No RF" (gsm-lab-ville.jpg), le meme que celui du bureau live.
    _VILLE="$DIR/configs/gsm-lab-ville.jpg"
    if [ -f "$_VILLE" ]; then
        cp "$_VILLE" "$ROOTFS/etc/calamares/branding/osmo/welcome.jpg"
    else
        sed -i '/^images:/,/^slideshow:/{/productWelcome/d}' \
            "$ROOTFS/etc/calamares/branding/osmo/branding.desc"
    fi

    # ── LE LANCEUR ──────────────────────────────────────────────────────
    # pkexec et non sudo : l installeur est lance depuis le bureau, ou il n y a
    # pas de terminal pour taper un mot de passe. La session tourne deja en
    # root, mais osmocom doit pouvoir lancer l installeur aussi - et c est la
    # que pkexec sert vraiment.
    cat > "$ROOTFS/usr/local/bin/osmo-install" <<'INSTALLER'
#!/bin/bash
# Lance l installeur du systeme. Sur la cle live uniquement.
set -u
if [ ! -r /etc/calamares/settings.conf ]; then
    echo "Calamares n est pas installe sur cette image (ISO_DESKTOP=0 ?)." >&2
    exit 1
fi

# On repasse root TOUT DE SUITE, et sur ce script - pas sur calamares. Ce qui
# suit (monter le medium, demonter une cible restee ouverte) demande root ;
# le faire apres pkexec, c etait le faire en simple utilisateur, donc pas du
# tout. pkexec transmet DISPLAY et XAUTHORITY, l interface s ouvre quand meme.
if [ "$(id -u)" -ne 0 ]; then
    exec pkexec --disable-internal-agent "$0" "$@"
fi

# ── Retrouver le squashfs, sans jamais supposer OU il est ──────────────
# LE SYMPTOME QUI A AMENE CE BLOC A SA FORME ACTUELLE. L entree "en RAM" du
# menu demarre parfaitement, et depuis elle - et depuis elle seule -
# l installeur repondait "Medium live introuvable ... Rien a installer".
#
# POURQUOI. En mode normal, live-boot monte le medium sur /run/live/medium et
# le squashfs y est a sa place d origine : live/filesystem.squashfs. En
# "toram=filesystem.squashfs", live-boot ne recopie PAS l arborescence du
# medium : il copie LE FICHIER, seul, a la racine d un tmpfs, puis deplace ce
# tmpfs sur /run/live/medium (cp -a "${MODULETORAMFILE}" "${copyto}", puis
# mount -r -o move). Le squashfs se retrouve donc en
#     /run/live/medium/filesystem.squashfs
# et non plus en
#     /run/live/medium/live/filesystem.squashfs
# Le sous-repertoire "live/" a disparu au passage. live-boot s en moque - il
# sait ou il l a mis - mais tout ce qui ecrit ce chemin en dur le rate.
#
```

```
# Et la version precedente de ce bloc le ratait DEUX FOIS : son repli prenait
# le fichier de backing du loop (donc le bon chemin), puis lui appliquait deux
# dirname pour "remonter a la racine du medium". Deux dirname sur
# /run/live/medium/filesystem.squashfs donnent /run/live - qui n a pas plus de
# sous-repertoire "live/". Le bind reussissait, le test suivant echouait, et le
# message accusait le medium d etre absent alors qu il etait en RAM, monte, et
# parfaitement lisible.
#
# CE QU ON FAIT A LA PLACE. On ne reconstitue plus une arborescence supposee :
# on trouve LE FICHIER, par trois voies de plus en plus larges, et on le
# presente a Calamares a un chemin qui, lui, ne depend d aucun mode de
# demarrage. C est ce chemin que nomme unpackfs.conf.
SQ=""

# 1. Le chemin canonique du demarrage normal. S il est la, rien a faire.
[ -e /run/live/medium/live/filesystem.squashfs ] \
    && SQ=/run/live/medium/live/filesystem.squashfs

# 2. Le loop qui porte la racine en cours d execution. C est la source la plus
#    sure qui soit : ce n est pas "un" squashfs trouve quelque part, c est
#    CELUI sur lequel ce systeme tourne. Vrai en normal, en toram et en
#    persistant.
if [ -z "$SQ" ]; then
    _loop=$(findmnt -no SOURCE /run/live/rootfs/filesystem.squashfs 2>/dev/null || true)
    if [ -n "$_loop" ]; then
        _bf=$(losetup -nO BACK-FILE "$_loop" 2>/dev/null || true)
        [ -n "$_bf" ] && [ -e "$_bf" ] && SQ="$_bf"
    fi
fi

# 3. Dernier recours : n importe quel loop dont le fichier de backing est un
#    squashfs. Couvre le cas ou live-boot aurait nomme le point de montage
#    autrement (persistance : /run/live/persistence/sdX).
if [ -z "$SQ" ]; then
    SQ=$(losetup -anO BACK-FILE 2>/dev/null | grep -m1 '\.squashfs$' || true)
    [ -n "$SQ" ] && [ -e "$SQ" ] || SQ=""
fi

if [ -z "$SQ" ]; then
    echo "Squashfs introuvable - ce systeme ne tourne pas depuis une cle live," >&2
    echo "ou l image n est plus lisible. Rien a installer." >&2
    exit 1
fi

# ── LE CHEMIN STABLE, CELUI QUE CALAMARES LIT ──────────────────────────
# Un bind sur le FICHIER, pas sur son repertoire : le repertoire d origine
# change de forme d un mode de demarrage a l autre (c est tout le probleme
# ci-dessus), le fichier non. /run est un tmpfs sur un systeme live, donc
# inscriptible - contrairement a /run/live/medium en toram, que live-boot
# deplace en lecture seule et dans lequel on ne pourrait pas creer le "live/"
# qui manque.
SRC=/run/osmo-install-src
mkdir -p "$SRC/live"
if ! mountpoint -q "$SRC/live/filesystem.squashfs"; then
    : > "$SRC/live/filesystem.squashfs"
    mount --bind "$SQ" "$SRC/live/filesystem.squashfs" || {
        echo "Impossible de presenter $SQ a l installeur." >&2
        exit 1
    }
fi
echo "Image source : $SQ  ->  $SRC/live/filesystem.squashfs"

# Le FICHIER, pas le repertoire. Tester "-d" ne prouvait rien : le repertoire
# existe meme vide, le test passait, et l echec tombait plus loin dans
# Calamares - au pire endroit, le disque deja repartitionne.
if [ ! -s "$SRC/live/filesystem.squashfs" ]; then
```

```
    echo "Le squashfs presente a l installeur est vide - rien a installer." >&2
    exit 1
fi

# ── Nettoyer la cible d un essai precedent ──────────────────────────────
# Quand une etape echoue, Calamares saute tous les jobs suivants, "umount"
# compris : la cible reste montee sous /tmp/calamares-root-*. Au lancement
# d apres il voit un disque monte, retire "Effacer le disque" de la liste et ne
# laisse que le partitionnement manuel. Chaque echec degradait l essai suivant.
for _t in /tmp/calamares-root-*; do
    [ -d "$_t" ] || continue
    if mountpoint -q "$_t"; then
        umount -R "$_t" 2>/dev/null || {
            echo "Cible $_t encore montee et non demontable - fermez ce qui l occupe" >&2
            echo "(un terminal, un gestionnaire de fichiers) ou redemarrez." >&2
            exit 1
        }
    fi
    rmdir "$_t" 2>/dev/null || true
done

# ── GARDE-FOU : une page generee invalide ne doit pas bloquer l installation ─
# [2026-09-04] Un seul antislash de trop dans contextualprocess-mirror.conf et
# Calamares s ouvrait sur "Calamares Initialization Failed - the following
# modules could not be loaded : contextualprocess@mirror" : plus d installeur
# du tout, pour une QUESTION DE MIROIR. Les deux pages ci-dessous sont
# ECRITES a chaque lancement, donc verifiees a chaque lancement : si le YAML
# ne se relit pas, on repose la version minimale (celle qui ne fait rien) et
# l operateur installe quand meme. yaml.safe_load lit ce que lit yaml-cpp.
# Sans python3-yaml sous la main, on ne bloque rien : le test s efface.
_yamlok() {
    python3 -c "import yaml" 2>/dev/null || return 0
    python3 - "$1" <<'YAMLCHECK'
import io, sys, yaml
lines = io.open(sys.argv[1], encoding="utf-8").read().splitlines()
i = next((n for n, l in enumerate(lines) if l.strip() == "---"), -1)
d = yaml.safe_load("\n".join(lines[i + 1:]))
if not isinstance(d, dict):
    raise SystemExit("pas une table YAML")
YAMLCHECK
}

# ── MIROIR : la page "Miroir Ubuntu" de l installeur suit le reseau ─────────
# packaging/apt-mirror.sh --list mesure les miroirs d ici (5 s max chacun) ;
# la page propose ceux qui repondent, du plus rapide au plus lent, le premier
# pre-selectionne, plus "celui de la cle" (aucun changement). Le module
# contextualprocess@mirror ecrit le choix dans la cible, apres unpackfs.
# [2026-09-04] AUCUN ANTISLASH DANS CES COMMANDES. Elles sont posees dans une
# chaine YAML entre guillemets, ou seule une poignee d echappements existe :
# \s et \S (les classes de sed) n en font pas partie et yaml-cpp refusait le
# FICHIER ENTIER - Calamares s ouvrait sur "contextualprocess@mirror could not
# be loaded". D ou les classes POSIX ([[:space:]], [^[:space:]]) et l expression
# sed entre guillemets echappes (\" est, lui, un echappement YAML valide), ce
# qui dispense aussi d echapper l espace du remplacement.
_pm=/etc/calamares/modules/packagechooser-mirror.conf
_cm=/etc/calamares/modules/contextualprocess-mirror.conf
_am=/opt/GSM/osmo-operator/packaging/apt-mirror.sh
if [ -f "$_pm" ] && [ -f "$_cm" ] && [ -f "$_am" ]; then
    _cur="$(sed -nE 's|^deb (http[^ ]+) .*|\1|p' /etc/apt/sources.list 2>/dev/null | head -1)"
    _suite="$(. /etc/os-release 2>/dev/null; echo "${VERSION_CODENAME:-noble}")"
    _mlist="$(bash "$_am" --list "$_suite" 2>/dev/null || true)"
    _n=0
    {
        echo "---"
        echo "mode: required"
        echo "method: legacy"
```

```
        echo "labels:"
        echo "    step: \"Miroir Ubuntu\""
        if [ -n "$_mlist" ]; then echo "default: m1"; else echo "default: keep"; fi
        echo "items:"
        echo "    - id: keep"
        echo "      name: \"Celui de la cle live\""
        echo "      description: \"Le systeme installe garde le miroir de la cle : ${_cur:-archive.ubuntu.com}.\""
        printf '%s\n' "$_mlist" | while IFS="$(printf '\t')" read -r _ms _url; do
            [ -n "$_url" ] || continue
            _n=$((_n + 1))
            _tag=""; [ "$_n" = 1 ] && _tag=" - le plus rapide d ici"
            echo "    - id: m${_n}"
            echo "      name: \"${_url#http://}\""
            echo "      description: \"${_url} - InRelease en ${_ms} ms${_tag}.\""
        done
    } > "$_pm"
    {
        echo "---"
        echo "dontChroot: false"
        echo "timeout: 60"
        echo "packagechooser_mirror:"
        echo "    \"keep\":"
        echo "        - \"-/bin/true\""
        _n=0
        printf '%s\n' "$_mlist" | while IFS="$(printf '\t')" read -r _ms _url; do
            [ -n "$_url" ] || continue
            _n=$((_n + 1))
            echo "    \"m${_n}\":"
            echo "        - \"-/bin/bash -c 'sed -i -E \\\"s,^deb[[:space:]]+https?://[^[:space:]]+,deb
↪  ${_url},\\\" /etc/apt/sources.list; [ -f /etc/apt/sources.list.d/ubuntu.sources ] && sed -i -E
↪  \\\"s,^URIs:[[:space:]]+https?://[^[:space:]]+,URIs: ${_url},\\\" /etc/apt/sources.list.d/ubuntu.sources;
↪  echo [mirror] ${_url}'\""
        done
    } > "$_cm"
    if ! _yamlok "$_pm" || ! _yamlok "$_cm"; then
        echo "Miroir Ubuntu : page generee ILLISIBLE - retour au choix minimal" >&2
        printf '%s\n' "---" "mode: required" "method: legacy" "labels:" \
            "    step: \"Miroir Ubuntu\"" "default: keep" "items:" \
            "    - id: keep" "      name: \"Celui de la cle live\"" \
            "      description: \"Le systeme installe garde le miroir de la cle.\"" > "$_pm"
        printf '%s\n' "---" "dontChroot: false" "timeout: 60" \
            "packagechooser_mirror:" "    \"keep\":" "        - \"-/bin/true\"" > "$_cm"
    elif [ -n "$_mlist" ]; then
        echo "Miroir Ubuntu : $(printf '%s\n' "$_mlist" | wc -l) miroirs mesures, le plus rapide : $(printf '%s\n'
↪  "$_mlist" | head -1 | cut -f2)"
    else
        echo "Miroir Ubuntu : aucune mesure (pas de reseau ?) - la cible garde ${_cur:-archive.ubuntu.com}"
    fi
fi

# ── NVIDIA : la page "Pilotes graphiques" de l installeur suit la machine ────
# Calamares ne sait pas griser une entree selon le materiel : on ECRIT sa page
# de choix et le module qui installe a CHAQUE lancement, d apres lspci :
#   - carte NVIDIA vue -> l entree "nvidia-driver-610" est nommee avec la carte
#     et pre-selectionnee ;
#   - sans carte NVIDIA, la page le dit et tout choix reste sans effet
#     (contextualprocess refait le test lspci dans la cible).
# Sur le systeme installe, tools/osmo-drivers.sh (icone "Pilotes graphiques")
# montre le meme etat et laisse installer ou mettre a jour plus tard.
#
# [2026-09-04] UNE SEULE VERSION, FIGEE : nvidia-driver-610. La page listait
# auparavant tout ce que `ubuntu-drivers list` proposait, plus une entree
# "recommande" qui deleguait le choix a `ubuntu-drivers install`. Resultat :
# une page differente d une machine et d une semaine a l autre, et une
# installation qui dependait d un outil absent des que le live n avait pas
# ubuntu-drivers-common. Le banc demande nvidia-driver-610 - il est dans
```

```
# noble-updates/multiverse et s installe comme n importe quel paquet :
#     sudo apt install nvidia-driver-610
# OSMO_NVIDIA_DRIVER permet d en figer un autre au build.
_pc=/etc/calamares/modules/packagechooser-nvidia.conf
_cp=/etc/calamares/modules/contextualprocess-nvidia.conf
_nvdrv="${OSMO_NVIDIA_DRIVER:-nvidia-driver-610}"
if [ -f "$_pc" ] && [ -f "$_cp" ]; then
    _nv="$(lspci -d 10de: 2>/dev/null | head -1 | cut -d: -f3- | sed 's/^ //; s/ (rev.*//')"
    # Aucun guillemet dans cette commande : elle est posee entre apostrophes
    # (bash -c) dans une chaine YAML entre guillemets - l un comme l autre y
    # seraient une fin de chaine. /run de la cible est un tmpfs vide (mount.conf)
    # et /etc/resolv.conf y pointe : on y ecrit des resolveurs le temps de l apt.
    _apt_cmd='mkdir -p /run/systemd/resolve; [ -s /run/systemd/resolve/stub-resolv.conf ] || { echo nameserver
↪   1.1.1.1 > /run/systemd/resolve/stub-resolv.conf; echo nameserver 8.8.8.8 >>
↪   /run/systemd/resolve/stub-resolv.conf; }; export DEBIAN_FRONTEND=noninteractive; lspci -n 2>/dev/null | grep
↪   -qi 10de: || { echo [nvidia] aucune carte NVIDIA : rien a installer; exit 0; }; apt-get update -qq || { echo
↪   [nvidia] pas de reseau : pilote non installe; exit 0; }'
    {
        echo "---"
        echo "mode: optional"
        echo "method: legacy"
        echo "labels:"
        echo "    step: \"Pilotes graphiques\""
        if [ -n "$_nv" ]; then echo "default: nvidia"; else echo "default: none"; fi
        echo "items:"
        echo "    - id: none"
        echo "      name: \"Aucun pilote supplementaire\""
        if [ -n "$_nv" ]; then
            _state=""
            dpkg -s "$_nvdrv" >/dev/null 2>&1 && _state=" (deja installe sur la cle)"
            echo "      description: \"Carte detectee : ${_nv}. Le pilote libre (nouveau) du noyau, comme sur la
↪   cle live.\""
            echo "    - id: nvidia"
            echo "      name: \"${_nvdrv} (proprietaire)\""
           echo "      description: \"Carte ${_nv} : installe ${_nvdrv}${_state} depuis les depots Ubuntu (reseau
↪   requis).\""
        else
           echo "     description: \"AUCUNE carte NVIDIA detectee (lspci) : rien a installer sur cette machine.\""
        fi
    } > "$_pc"
    {
        echo "---"
        echo "dontChroot: false"
        echo "timeout: 1800"
        echo "packagechooser_nvidia:"
        echo "    \"none\":"
        echo "        - \"-/bin/true\""
        echo "    \"nvidia\":"
        echo "        - \"-/bin/bash -c '${_apt_cmd}; apt-get install -y ${_nvdrv} || echo [nvidia] ${_nvdrv} non
↪   installable - le pilote libre nouveau reste en place; update-initramfs -u >/dev/null 2>&1 || true'\""
    } > "$_cp"
    if ! _yamlok "$_pc" || ! _yamlok "$_cp"; then
        echo "NVIDIA : page generee ILLISIBLE - retour au choix minimal" >&2
        printf '%s\n' "---" "mode: optional" "method: legacy" "labels:" \
            "    step: \"Pilotes graphiques\"" "default: none" "items:" \
            "    - id: none" "      name: \"Aucun pilote supplementaire\"" \
            "      description: \"Le pilote libre (nouveau) du noyau.\"" > "$_pc"
        printf '%s\n' "---" "dontChroot: false" "timeout: 1800" \
            "packagechooser_nvidia:" "    \"none\":" "        - \"-/bin/true\"" > "$_cp"
    elif [ -n "$_nv" ]; then
        echo "NVIDIA : $_nv - pilote propose : $_nvdrv"
    else
        echo "NVIDIA : aucune carte detectee - la page le dira, sans effet"
    fi
fi
exec /usr/bin/calamares -d "$@"
```

```
INSTALLER
    chmod +x "$ROOTFS/usr/local/bin/osmo-install"

    # ── CONKY : le tableau de bord du banc, dans TOUTE session GNOME ──────────
    # [2026-09-03] /etc/xdg/autostart vaut pour tous les comptes : root sur la
    # cle live, l utilisateur cree par Calamares sur le disque (le fichier
    # voyage dans le squashfs, donc dans la copie installee). La config et le
    # script d etat vivent dans le depot (/opt/GSM/osmo-operator, present sur
    # les deux) : configs/conky/osmo-conky.conf, tools/conky-osmo-status.sh.
    # sleep 6 : conky doit trouver le bureau GNOME deja peint (own_window_type
    # desktop), sinon il reste derriere le fond d ecran. --daemonize : la
    # session n attend pas. Une seule instance : pkill avant.
    # ── UN VEILLEUR, PAS UN LANCEMENT UNIQUE ──────────────────────────────────
    # [2026-09-04] L autostart lancait osmo-panel.py UNE FOIS, en tache de fond.
    # osmo-panel.py est la fenetre qui AFFICHE /run/osmo-fft/panel.png : sans
    # elle, l encart n existe pas a l ecran, quoi que fabrique osmo-fft-snap.
    # Or c est un python3 - et le demontage du banc tuait tous les python3 (voir
    # killall_python dans start-native.sh). Un seul arret du banc, et l encart
    # disparaissait de l ecran POUR LE RESTE DE LA SESSION : plus de FFT, plus
    # de mobile.log, et rien dans aucun journal pour le dire.
    # Le teardown les epargne desormais, mais un plantage, un OOM ou un kill a
    # la main auraient le meme effet. On les surveille donc : ce script relance
    # ce qui manque, toutes les cinq secondes, sans jamais doubler.
    cat > "$ROOTFS/usr/local/bin/osmo-desktop-panel" <<'PANEL'
#!/bin/sh
# osmo-desktop-panel - tient l encart et Conky en vie pour toute la session.
# Lance par /etc/xdg/autostart/osmo-conky.desktop. Voir build-iso.sh (85).
REPO=/opt/GSM/osmo-operator
export OSMO_REPO="$REPO"
# La carte LAB GSM n est plus cuite dans le fond : c est osmo-topzone.py qui la
# rend, vivante. On le dit au rendu du fond.
export OSMO_LIVE_BANNER=1

# ── UN WIDGET RESCAPE DE LA SESSION D AVANT N EST PAS UN WIDGET ────────────
# [2026-09-10] Le Conky avait disparu du coin haut droit, et le gardien ne le
# relancait JAMAIS. Pourquoi : un conky de la session X precedente tournait
# encore, avec DISPLAY=:0, alors que GNOME etait reparti sur :1 (le socket
# /tmp/.X11-unix/X0 avait disparu avec l ancien serveur). Il ne peut plus rien
# peindre - « Invalid MIT-MAGIC-COOKIE-1 key / can't open display: :0 » dans
# /tmp/osmo-conky.log - mais il EXISTE, et « pgrep -x conky » ne sait dire que
# ca. Le gardien le comptait pour vivant et n en relancait pas d autre : plus
# de tableau de bord pour toute la session. Meme piege pour les widgets python
# (osmo-ts-probe.py etait dans le meme etat).
#
# On ne compte donc que les processus qui regardent NOTRE display, et on tue
# les autres : ils ne peuvent plus servir personne.
osmo_vivant() {
    _trouve=1
    for _p in $(pgrep "$@" 2>/dev/null); do
        _d="$(tr '\0' '\n' < "/proc/$_p/environ" 2>/dev/null | sed -n 's/^DISPLAY=//p' | head -1)"
        # On ne tue QUE ce qu on a lu et qui regarde ailleurs. Un environ
        # illisible, ou sans DISPLAY, ne prouve rien : ce processus compte
        # pour vivant - une sonde qui confond « absent » et « invisible »
        # ferait ici pousser un doublon de chaque widget a chaque tour.
        if [ -n "$_d" ] && [ "$_d" != "$DISPLAY" ]; then kill "$_p" 2>/dev/null
        else _trouve=0
        fi
    done
    return $_trouve
}
# Et le gardien ne survit pas a son propre serveur X : sans cette sortie,
# l ancien tourne en parallele du neuf et relance dans le vide, pour toujours
# (on en avait deux, l un sur :0 et l autre sur :1).
osmo_display_vivant() {
    [ -n "$DISPLAY" ] || return 0          # pas de display nomme : on ne juge pas
    _n="${DISPLAY#*:}"; _n="${_n%%.*}"
```

```
    [ -S "/tmp/.X11-unix/X$_n" ]
}

# ── LES FLECHES DU CONKY, AU CLAVIER : Ctrl+AltGr+Gauche / Droite ──────────
# [2026-09-10] Le raccourci est cuit dans un override de schema
# (/usr/share/glib-2.0/schemas/98-osmo-keys.gschema.override, pose par
# iso_modules/80-chroot.sh) - sauf qu un override ne prend PAS sur un schema
# RELOCALISABLE : « gsettings get ...custom-keybinding:/...osmo-op-next/
# binding » rendait '' sur un systeme ou le fichier etait pourtant bien
# compile. Le raccourci n a donc jamais rien lie. On l ecrit ici, dans la
# session, une fois par demarrage : c est le seul endroit qui tourne SOUS le
# compte de l utilisateur, et gsettings ecrit dans son dconf.
# <Mod5> est ISO_Level3_Shift, c est-a-dire AltGr (xmodmap -pm le confirme).
osmo_raccourci() {
    command -v gsettings >/dev/null 2>&1 || return 0
    _k=org.gnome.settings-daemon.plugins.media-keys.custom-keybinding
    _p=/org/gnome/settings-daemon/plugins/media-keys/custom-keybindings
    gsettings set org.gnome.settings-daemon.plugins.media-keys custom-keybindings \
        "['$_p/osmo-op-next/','$_p/osmo-op-prev/']" 2>/dev/null || return 0
    gsettings set "$_k:$_p/osmo-op-next/" name 'osmo-operator : arret suivant (Conky)' 2>/dev/null
    gsettings set "$_k:$_p/osmo-op-next/" command '/usr/local/bin/osmo-op --next' 2>/dev/null
    gsettings set "$_k:$_p/osmo-op-next/" binding '<Control><Mod5>Right' 2>/dev/null
    gsettings set "$_k:$_p/osmo-op-prev/" name 'osmo-operator : arret precedent (Conky)' 2>/dev/null
    gsettings set "$_k:$_p/osmo-op-prev/" command '/usr/local/bin/osmo-op --prev' 2>/dev/null
    gsettings set "$_k:$_p/osmo-op-prev/" binding '<Control><Mod5>Left' 2>/dev/null
}
osmo_raccourci

# Conky doit trouver le bureau GNOME deja peint (own_window_type desktop),
# sinon il se pose derriere le fond d ecran.
sleep 6
while osmo_display_vivant; do
    osmo_vivant -f "$REPO/tools/osmo-panel.py" || \
        "$REPO/tools/osmo-panel.py" >>/tmp/osmo-panel.log 2>&1 &
    # [2026-09-13] LE DINO N'EST PLUS SUR LE BUREAU. Il etait relance ici en
    # boucle, en fenetre de type BUREAU collee sous le Conky. Il est maintenant
    # une APPLICATION : data/desktop/osmo-dino.desktop, dans le menu, qui lance
    # osmo-dino.py --app (fenetre ordinaire, page entiere, plein ecran par son
    # propre bouton). Le bureau ne porte plus que le Conky et ses encarts.
    # la sonde d activite des timeslots (conf osmo-bsc + VTY) pour la banniere
    osmo_vivant -f "$REPO/tools/osmo-ts-probe.py" || \
        "$REPO/tools/osmo-ts-probe.py" >>/tmp/osmo-ts-probe.log 2>&1 &
    # la zone haute LAB GSM vivante (timeslots clignotants + accueil d appli)
    osmo_vivant -f "$REPO/tools/osmo-topzone.py" || \
        "$REPO/tools/osmo-topzone.py" >>/tmp/osmo-topzone.log 2>&1 &
    # [2026-10-12] LA BARRE DU BAS N EST PLUS SUR LE BUREAU (choix du banc :
    # « enleve les choix en bas »). Le 2G/4G se lancent par les icones et
    # launch.sh ; OSMO_LAUNCHER=1 dans l environnement de session la remet.
    if [ "${OSMO_LAUNCHER:-0}" = 1 ]; then
        osmo_vivant -f "$REPO/tools/osmo-launcher.py" || \
            "$REPO/tools/osmo-launcher.py" >>/tmp/osmo-launcher.log 2>&1 &
    fi
    osmo_vivant -x conky || \
        conky --daemonize -c "$REPO/configs/conky/osmo-conky.conf" >>/tmp/osmo-conky.log 2>&1
    sleep 5
done
PANEL
    chmod 755 "$ROOTFS/usr/local/bin/osmo-desktop-panel"

    cat > "$ROOTFS/etc/xdg/autostart/osmo-conky.desktop" <<'CONKY'
[Desktop Entry]
Type=Application
Name=Conky osmo-operator
Comment=Tableau de bord du banc GSM (coeur, radio, abonnes, services)
Exec=/usr/local/bin/osmo-desktop-panel
Icon=utilities-system-monitor
```

```
Terminal=false
NoDisplay=true
OnlyShowIn=GNOME;
X-GNOME-Autostart-enabled=true
X-GNOME-Autostart-Delay=6
CONKY
    chmod 644 "$ROOTFS/etc/xdg/autostart/osmo-conky.desktop"
    chmod +x "$ROOTFS/opt/GSM/osmo-operator/tools/osmo-dino.py" \
             "$ROOTFS/opt/GSM/osmo-operator/tools/conky-osmo-status.sh" \
             "$ROOTFS/opt/GSM/osmo-operator/tools/osmo-drivers.sh" \
             "$ROOTFS/opt/GSM/osmo-operator/tools/osmo-fft-snap.py" \
             "$ROOTFS/opt/GSM/osmo-operator/tools/osmo-panel.py" \
             "$ROOTFS/opt/GSM/osmo-operator/tools/wallpaper-render.py" \
             "$ROOTFS/opt/GSM/osmo-operator/tools/osmo-wallpaper.sh" 2>/dev/null || true
    echo -e "  ${GREEN}✓${NC} Conky du banc : autostart GNOME (live et disque), config
     ↪  ${CYAN}configs/conky/osmo-conky.conf${NC}"

    # Le fichier vit dans le depot (data/desktop/), pas en heredoc ici :
    # l install native (install_modules/80-bureau.sh) et le paquet .deb posent
    # le MEME.
    # [2026-09-03] TOUS les raccourcis du depot, pas seulement osmo-install.
    # Ils etaient poses par l install native (install_modules/80-bureau.sh) et
    # par le .deb, mais PAS dans l ISO : la cle live n avait qu une icone sur
    # neuf, et les huit autres n existaient meme pas dans le menu des
    # applications — donc le dock ne pouvait pas les afficher non plus.
    # [2026-10-12] SAUF L ICONE multi-operator : elle n a de sens qu une fois
    # les Supplements poses (docker, l image operateur, osmo-multi.conf), et
    # c est addition.sh --multi qui la pose a ce moment-la. Copiee ici, elle
    # etait au dock des le premier boot et son clic finissait en « il manque
    # ... » (launch.sh --multi). Le dock la montre quand elle existe
    # (favorite-apps la nomme deja, iso_modules/80-chroot.sh).
    for _d in "$DIR"/data/desktop/*.desktop; do
        [ -f "$_d" ] || continue
        [ "$(basename "$_d")" = osmo-multi.desktop ] && continue
        install -m644 "$_d" "$ROOTFS/usr/share/applications/$(basename "$_d")"
    done

    # ── PLUS D ICONES SUR LE BUREAU : LE DOCK SUFFIT ──────────────────
    # [2026-09-04] Les raccourcis ne sont plus copies dans ~/Desktop ni
    # ~/Bureau : ils vivent dans /usr/share/applications (ci-dessus) et dans
    # les favoris du dock (favorite-apps, iso_modules/80-chroot.sh). Cela
    # retire aussi tout le mecanisme d approbation (metadata::trusted,
    # osmo-trust-desktop au login) qui n existait que pour DING.
    # Le bureau reste vide ; le fond d ecran du jour s y voit en entier.
    rm -f "$ROOTFS/root/Desktop"/*.desktop "$ROOTFS/root/Bureau"/*.desktop \
          "$ROOTFS/home/osmocom/Desktop"/*.desktop "$ROOTFS/home/osmocom/Bureau"/*.desktop \
          "$ROOTFS/usr/local/bin/osmo-trust-desktop" \
          "$ROOTFS/etc/xdg/autostart/osmo-trust-desktop.desktop" 2>/dev/null || true
    chroot "$ROOTFS" chown -R osmocom:osmocom /home/osmocom 2>/dev/null || true

    # ── LANCEUR ET TUTORIEL : ICONES DU BUREAU ────────────────────────
    # Icones du depot (data/*.svg) et pas des noms du theme : "call-start" est
    # VERT dans Adwaita et n a pas de variante rouge, et un nom d icone absent
    # se remplace EN SILENCE par un rectangle gris - le lanceur devient alors
    # introuvable sur le bureau qu il est cense ouvrir.
    install -d "$ROOTFS/usr/share/icons/hicolor/scalable/apps" \
              "$ROOTFS/usr/share/osmo-operator/icons" \
              "$ROOTFS/usr/share/osmo-operator"
    for _ic in osmo-launch osmo-multi osmo-tutorial claude osmo-paint osmo-lte osmo-dino; do
        [ -f "$DIR/data/$_ic.svg" ] || continue
        cp -f "$DIR/data/$_ic.svg" \
              "$ROOTFS/usr/share/icons/hicolor/scalable/apps/$_ic.svg"
        # LA COPIE QUI COMPTE POUR LE BUREAU. Voir le bloc juste dessous.
        cp -f "$DIR/data/$_ic.svg" \
              "$ROOTFS/usr/share/osmo-operator/icons/$_ic.svg"
        chmod 644 "$ROOTFS/usr/share/osmo-operator/icons/$_ic.svg"
```

```
    done

    # ⚠ LES .desktop POINTENT LE FICHIER, PAS LE NOM DE L ICONE.
    # [2026-08-31] Les trois raccourcis s affichaient en PAGE BLANCHE generique
    # sur le bureau, SVG valides et bien installes. Cause : "Icon=osmo-launch"
    # n est pas un chemin, c est un nom a resoudre dans le thème, et cette
    # resolution passe par /usr/share/icons/hicolor/icon-theme.cache. Le cache
    # etait construit AVANT que les icones n arrivent - constate sur le banc :
    #     strings .../icon-theme.cache | grep -c osmo   ->  0
    #     cache 17:22:12   ·   icones 17:28:07
    # Zero entree sur trois, et rien ne le signale : un nom d icone qui ne
    # resout pas se remplace EN SILENCE par la page blanche. "Supplements" s en
    # sortait seul parce que son "system-software-install" vient de Yaru, deja
    # dans le cache depuis l installation du systeme.
    # D ou les deux mesures, et pas une seule :
    #   - Icon= en CHEMIN ABSOLU (plus bas) : court-circuite thème et cache,
    #     c est ce qui garantit l icone sur le bureau ;
    #   - le cache reconstruit quand meme, pour le menu des applications, qui
    #     lui continue de resoudre par nom.
    if chroot "$ROOTFS" sh -c 'command -v gtk-update-icon-cache' >/dev/null 2>&1; then
        chroot "$ROOTFS" gtk-update-icon-cache -f -q /usr/share/icons/hicolor 2>/dev/null || true
    fi
    [ -f "$DIR/data/tutorial.html" ] && \
        cp -f "$DIR/data/tutorial.html" "$ROOTFS/usr/share/osmo-operator/tutorial.html"

    # Un exemplaire dans /usr/share sert de repli hors ligne : c est le seul
    # chemin qui reste lisible quand le dashboard ne repond pas.
    cat > "$ROOTFS/usr/local/bin/osmo-tutorial" <<'TUTO'
#!/bin/bash
# osmo-tutorial - ouvre le quick-start.
#
# PAR HTTP, PAS PAR file://. Le dashboard sert deja ce fichier sur 8080 : en
# passant par lui, la page affichee est CELLE QUI FAIT FOI (celle du depot,
# recopiee a chaque build), la meme que depuis une autre machine du banc, et
# les liens relatifs vers le tableau de bord marchent. Le file:// reste en
# repli, pour le cas ou le dashboard ne repond pas.
set -u
URL="${OSMO_TUTORIAL_URL:-http://127.0.0.1:8080/tutorial.html}"
PORT="${URL##*:}"; PORT="${PORT%%/*}"
if (exec 3<>"/dev/tcp/127.0.0.1/${PORT}") 2>/dev/null; then
    exec 3>&- 2>/dev/null
    exec xdg-open "$URL"
fi
echo "Dashboard injoignable sur $PORT - repli file:// sur la copie de /usr/share" >&2
exec xdg-open "file:///usr/share/osmo-operator/tutorial.html"
TUTO
    chmod +x "$ROOTFS/usr/local/bin/osmo-tutorial"
    # [2026-10-23] le meme lanceur pour la page d aide (le « ? »), servie par le dashboard
    install -m755 "$DIR/tools/osmo-info" "$ROOTFS/usr/local/bin/osmo-info"

    # Le fichier vit dans le depot (data/desktop/), pas en heredoc ici :
    # l install native (install_modules/80-bureau.sh) et le paquet .deb posent
    # le MEME.
    install -m644 "$DIR/data/desktop/osmo-launch.desktop" "$ROOTFS/usr/share/applications/osmo-launch.desktop"
    # [2026-09-04] Plus d'icone DSP a part. [2026-10-01] Le DSP est le defaut
    # du banc ; le clic droit du telephone propose l'autre chaine (« Lancer en
    # mode gr-gsm »).
    rm -f "$ROOTFS/usr/share/applications/osmo-dsp.desktop"

    # osmo-multi (antenne, multi-operator) N EST PLUS POSEE ICI. Son lanceur
    # start-multi.sh suppose docker + l image + la topologie SS7, qui n existent
    # qu apres le supplement (addition.sh). L icone apparaissait donc au premier
    # boot pour ne rien faire au clic ; addition.sh la pose desormais LUI-MEME,
    # a la fin d une install SS7 reussie. Son SVG reste installe (bloc icones
    # ci-dessus) pour que cette pose differee y trouve l image.
```

```
    # ── SUPPLEMENTS : LA FENETRE A COCHER ──────────────────────────────────
    # [2026-10-01] Meme facture que osmo-update-anim : une fenetre GTK du banc
    # avec un terminal VTE dedans (tools/osmo-gtk-run.py --tty), et la main
    # rendue seulement quand on a lu la fin. addition.sh ouvre lui-meme sa
    # liste a cocher (zenity) quand DISPLAY est la ; le terminal reste utile
    # pour la suite, longue et bavarde (apt, puis compilation Osmocom). Aucun
    # favori du dock n ouvre plus de gnome-terminal.
    cat > "$ROOTFS/usr/local/bin/osmo-addition-anim" <<'ADDGUI'
#!/bin/bash
# osmo-addition-anim - les supplements (addition.sh) dans une fenetre GTK du
# banc : un terminal VTE dedans, pour la liste a cocher puis la suite, longue
# et bavarde (apt, compilation) ; la fenetre reste ouverte a la fin, avec le
# statut. Root par pkexec, DISPLAY/XAUTHORITY et le proxy de la session
# reportes (osmo-gtk-run --root). [2026-10-01] Plus de gnome-terminal.
set -u
REPO=/opt/GSM/osmo-operator
SCRIPT=$REPO/addition.sh
if [ ! -x "$SCRIPT" ]; then
    command -v zenity >/dev/null 2>&1 && \
        zenity --error --text="addition.sh introuvable : $SCRIPT" 2>/dev/null
    exit 1
fi
# [2026-10-12] --progress : barre + etape, la console (VTE) repliee derriere « console »
exec python3 "$REPO/tools/osmo-gtk-run.py" --progress --root --title "osmo-operator - supplements" -- "$SCRIPT"
 ↪  "$@"

ADDGUI
    chmod +x "$ROOTFS/usr/local/bin/osmo-addition-anim"

    # Le fichier vit dans le depot (data/desktop/), pas en heredoc ici :
    # l install native (install_modules/80-bureau.sh) et le paquet .deb posent
    # le MEME.
    install -m644 "$DIR/data/desktop/osmo-addition.desktop" "$ROOTFS/usr/share/applications/osmo-addition.desktop"

    # Le fichier vit dans le depot (data/desktop/), pas en heredoc ici :
    # l install native (install_modules/80-bureau.sh) et le paquet .deb posent
    # le MEME.
    install -m644 "$DIR/data/desktop/osmo-tutorial.desktop" "$ROOTFS/usr/share/applications/osmo-tutorial.desktop"
    # [2026-10-01] La page Info (anglais) : chaque module du banc, par icone,
    # barre de lancement ou CLI, avec le code derriere. Icone dans les favoris.
    install -d "$ROOTFS/usr/share/osmo-operator"
    install -m644 "$DIR/configs/info.html" "$ROOTFS/usr/share/osmo-operator/info.html"
    install -d "$ROOTFS/usr/share/osmo-operator/icons"
    install -m644 "$DIR/data/osmo-info.svg" "$ROOTFS/usr/share/osmo-operator/icons/osmo-info.svg"
    install -m644 "$DIR/data/desktop/osmo-info.desktop" "$ROOTFS/usr/share/applications/osmo-info.desktop"

    # ── CLAUDE : lanceur + entree de menu ──────────────────────────────────
    # Claude Code n est PAS dans l ISO (il s installe via le supplement) : le
    # lanceur enchaine donc l installation (addition.sh --claude) au premier
    # clic si claude manque, puis l ouvre. L icone est la des le boot, mais elle
    # ne ment pas - elle sait s installer elle-meme.
    cat > "$ROOTFS/usr/local/bin/osmo-claude-anim" <<'CLA'
#!/bin/bash
# osmo-claude-anim - Claude Code dans une fenetre GTK du banc (terminal VTE :
# claude est interactif de bout en bout). Pas dans l ISO : le premier clic
# l installe (addition.sh --claude, en root) puis l ouvre, dans la meme
# fenetre. [2026-10-01] Plus de gnome-terminal.
set -u
exec python3 /opt/GSM/osmo-operator/tools/osmo-gtk-run.py --tty --title "Claude" -- bash -lc '
if ! command -v claude >/dev/null 2>&1; then
    ADD=/opt/GSM/osmo-operator/addition.sh
    if [ -x "$ADD" ]; then
        if [ "$(id -u)" -ne 0 ] && command -v pkexec >/dev/null 2>&1; then
            pkexec env DISPLAY="${DISPLAY:-}" XAUTHORITY="${XAUTHORITY:-}" "$ADD" --claude
        else
            "$ADD" --claude
```

```
        fi
    fi
fi
if command -v claude >/dev/null 2>&1; then exec claude; fi
echo "Claude non installe - lancez le supplement (addition.sh --claude)."
exit 1'

CLA
    chmod +x "$ROOTFS/usr/local/bin/osmo-claude-anim"
    # Le fichier vit dans le depot (data/desktop/) ; Icon= en chemin absolu.
    install -m644 "$DIR/data/desktop/claude.desktop" "$ROOTFS/usr/share/applications/claude.desktop"
    sed -i "s|^Icon=.*|Icon=/usr/share/osmo-operator/icons/claude.svg|" \
        "$ROOTFS/usr/share/applications/claude.desktop"

    # Terminal=false : launch.sh ouvre LUI-MEME son terminal et demande les
    # privileges (pkexec). [2026-09-04] Plus de copie sur le bureau : les
    # raccourcis sont dans le menu et les favoris du dock.
    echo -e "  ${GREEN}✓${NC} menu/dock : ${CYAN}telephone${NC} (lancer) · ${CYAN}livre${NC} (tutoriel) ·
     ↪  ${CYAN}supplements${NC} · ${CYAN}Claude${NC}"

    # Le paquet calamares pose SA propre entree de menu, qui lance
    # /usr/bin/calamares directement. Elle court-circuite osmo-install : ni le
    # medium remis a sa place, ni la cible d un essai precedent demontee - donc
    # exactement la panne qu on vient de corriger, a un clic de la bonne icone.
    # NoDisplay la retire des menus sans toucher au paquet.
    if [ -f "$ROOTFS/usr/share/applications/calamares.desktop" ]; then
        grep -q '^NoDisplay=' "$ROOTFS/usr/share/applications/calamares.desktop" \
            || echo 'NoDisplay=true' >> "$ROOTFS/usr/share/applications/calamares.desktop"
    fi

    echo -e "  ${GREEN}✓${NC} installeur ${CYAN}Calamares${NC} : /usr/local/bin/osmo-install (+ icone sur le
     ↪  bureau)"
elif [ "${ISO_DESKTOP:-0}" = "1" ]; then
    echo -e "  ${YELLOW}!${NC} $_CAL_SRC absent - pas d installeur dans cette image"
fi

# ── SANS --desktop : LES RACCOURCIS ET LEURS ICONES QUAND MEME ──────────────
# [2026-10-01] Le controle 4G de 88-lte-pmos.sh exige osmo-lte.desktop sur
# TOUTE image operateur, et l ISO de base (osmo-operator.iso, sans GNOME) ne le
# posait pas : le bloc ci-dessus est conditionne par --desktop. En CI, la passe
# --all tombait donc sur « ✗ 4G osmo-lte.service + osmo-lte.desktop » des la
# deuxieme image. On pose ici, hors bureau, ce que l install native
# (install_modules/80-bureau.sh) pose de toute facon : les .desktop du depot et
# leurs SVG. Quelques Ko, et un bureau installe apres coup (apt install
# ubuntu-desktop) trouve ses entrees de menu toutes pretes.
if [ "${ISO_DESKTOP:-0}" != "1" ]; then
    install -d "$ROOTFS/usr/share/applications" \
              "$ROOTFS/usr/share/icons/hicolor/scalable/apps" \
              "$ROOTFS/usr/share/osmo-operator/icons"
    for _ic in osmo-launch osmo-multi osmo-tutorial claude osmo-paint osmo-lte osmo-dino osmo-info; do
        [ -f "$DIR/data/$_ic.svg" ] || continue
        install -m644 "$DIR/data/$_ic.svg" "$ROOTFS/usr/share/icons/hicolor/scalable/apps/$_ic.svg"
        install -m644 "$DIR/data/$_ic.svg" "$ROOTFS/usr/share/osmo-operator/icons/$_ic.svg"
    done
    # [2026-10-12] SAUF L ICONE multi-operator : elle n a de sens qu une fois
    # les Supplements poses (docker, l image operateur, osmo-multi.conf), et
    # c est addition.sh --multi qui la pose a ce moment-la. Copiee ici, elle
    # etait au dock des le premier boot et son clic finissait en « il manque
    # ... » (launch.sh --multi). Le dock la montre quand elle existe
    # (favorite-apps la nomme deja, iso_modules/80-chroot.sh).
    for _d in "$DIR"/data/desktop/*.desktop; do
        [ -f "$_d" ] || continue
        [ "$(basename "$_d")" = osmo-multi.desktop ] && continue
        install -m644 "$_d" "$ROOTFS/usr/share/applications/$(basename "$_d")"
    done
    echo -e "  ${GREEN}✓${NC} raccourcis du depot (data/desktop/) + icones poses, sans bureau"
```

```
fi

# ── LE LANCEUR GTK "UPDATE" -> update.sh DU DEPOT ──────────────────────────
# Independant de Calamares : c'est une icone de bureau qui rejoue update.sh, le
# fichier SUIVI dans le depot osmo-operator (/opt/GSM/osmo-operator/update.sh).
# update.sh est une animation de TERMINAL : sa premiere ligne utile est
# "[ -t 1 ] || exit 0", donc lance sans tty (depuis une icone GTK) il sort
# aussitot sans rien montrer. [2026-10-01] Le lanceur l'ouvre DONC dans un
# terminal VTE pose DANS une fenetre GTK du banc (tools/osmo-gtk-run.py
# --tty), qui reste ouverte a la fin pour qu'on lise le resultat - plus de
# gnome-terminal.
if [ "${ISO_DESKTOP:-0}" = "1" ]; then
    cat > "$ROOTFS/usr/local/bin/osmo-update-anim" <<'UPDGUI'
#!/bin/bash
# osmo-update-anim - rejoue update.sh du depot dans une fenetre GTK du banc.
# update.sh est une animation de TERMINAL ([ -t 1 ] || exit 0) : --tty, un
# terminal VTE dans la fenetre, qui reste ouverte sur la derniere ligne
# (« SMS delivered ») avec le statut. [2026-10-01] Plus de gnome-terminal.
set -u
SCRIPT=/opt/GSM/osmo-operator/update.sh
if [ ! -x "$SCRIPT" ]; then
    command -v zenity >/dev/null 2>&1 && \
        zenity --error --text="update.sh introuvable : $SCRIPT" 2>/dev/null
    exit 1
fi
# [2026-10-12] --progress : barre + etape, la console (VTE) repliee derriere « console »
exec python3 /opt/GSM/osmo-operator/tools/osmo-gtk-run.py --progress --title "osmo-operator - update" --
 ↪  "$SCRIPT" "$@"

UPDGUI
    chmod +x "$ROOTFS/usr/local/bin/osmo-update-anim"

    # Le fichier vit dans le depot (data/desktop/), pas en heredoc ici :
    # l install native (install_modules/80-bureau.sh) et le paquet .deb posent
    # le MEME.
    install -m644 "$DIR/data/desktop/osmo-update.desktop" "$ROOTFS/usr/share/applications/osmo-update.desktop"

    # [2026-09-04] Pas de copie sur le bureau (menu + favoris du dock).

    echo -e "  ${GREEN}✓${NC} lanceur GTK ${CYAN}update${NC} : /usr/local/bin/osmo-update-anim (+ icone sur le
     ↪  bureau)"
fi


# ── LE BUREAU, SUITE : FOND DU JOUR, SPECTRES, TERMINAL, DING ──────────────
# [2026-09-04] Tout ce bloc est HORS du chroot de 80-chroot.sh (quotes simples
# la-bas) : ici les heredocs et les apostrophes sont libres.
if [ "${ISO_DESKTOP:-0}" = "1" ]; then
    # ── 1. Le fond d ecran du jour : strip Calvin & Hobbes incruste ──────────
    # tools/osmo-wallpaper.sh va chercher le strip du jour (gocomics, comme le
    # gist hellogist), recompose la photo + carte LAB GSM + strip
    # (tools/wallpaper-render.py) et pousse le resultat dans les sessions
    # ouvertes. Au boot (40 s, le reseau est la), chaque jour a 6h30, et
    # toutes les 4 h en rattrapage (banc hors ligne au boot).
    install -m755 "$DIR/tools/osmo-wallpaper.sh" "$ROOTFS/usr/local/sbin/osmo-wallpaper"
    # L arbre /opt/GSM/osmo-operator de l image vient du clone GitHub fait dans
    # Docker : on y pose aussi, depuis CE depot, ce que le fond et les spectres
    # utilisent - le build ne depend pas d un push pour ces fichiers-la.
    _rt="$ROOTFS/opt/GSM/osmo-operator"
    install -d "$_rt/tools" "$_rt/configs/wallpaper" "$_rt/configs/conky"
    # conky-osmo-status.sh est dans la liste depuis le 2026-09-04 : c est lui
    # qui dit si le HLR tourne (il lisait sa base sqlite, qui survit au demon,
    # et annoncait un HLR allume des heures apres l arret du banc). Sans cette
    # ligne il ne venait QUE du clone GitHub - donc une correction non poussee
    # ne partait pas dans l image, alors que les quatre autres outils, eux,
    # partaient. Meme regle pour tous : le depot local fait foi.
```

```
    install -m755 "$DIR/tools/wallpaper-render.py" "$DIR/tools/osmo-fft-snap.py" "$DIR/tools/osmo-panel.py"
     ↪  "$DIR/tools/osmo-wallpaper.sh" "$DIR/tools/conky-osmo-status.sh" "$_rt/tools/"
    # Le bureau interactif (banniere LAB GSM vivante + barre de lancement) et le
    # dino : fichiers locaux, souvent pas encore pousses - on les pose depuis CE
    # depot, comme les cinq ci-dessus. osmo-extras-install.sh sert au natif (80).
    for _f in osmo-launcher.py osmo-topzone.py osmo-ts-probe.py osmo-dino.py osmo-extras-install.sh osmo-fond.py \
              osmo-pmos.sh osmo-pmos-qemu.sh osmo-pmos-setup.sh osmo-wireshark-root.sh \
              osmo-ofono-bridge.py osmo-phonesim-banc.py \
              osmo-ril-atmodem.py at-cmd.py; do
        [ -f "$DIR/tools/$_f" ] && install -m755 "$DIR/tools/$_f" "$_rt/tools/"
    done
    unset _f
    # tower.* et non tower.jpg : la photo voyage dans son format d origine
    # (PNG pleine definition aujourd hui), sans re-encodage.
    for _tw in "$DIR"/configs/wallpaper/tower.*; do
        [ -f "$_tw" ] && install -m644 "$_tw" "$_rt/configs/wallpaper/"
    done
    unset _tw
    install -m644 "$DIR/configs/conky/osmo-conky.conf" "$_rt/configs/conky/"
    # [2026-10-12] la plaque du conky (Cairo), chargee par osmo-conky.conf
    [ -f "$DIR/configs/conky/osmo-conky.lua" ] && install -m644 "$DIR/configs/conky/osmo-conky.lua"
     ↪  "$_rt/configs/conky/"
    # la banniere LAB GSM vivante et le html du dino (rendus par WebKit)
    for _h in labgsm.html dino.html; do
        [ -f "$DIR/configs/conky/$_h" ] && install -m644 "$DIR/configs/conky/$_h" "$_rt/configs/conky/"
    done
    unset _h
    install -m644 "$DIR/configs/gsm-lab-wallpaper.png" "$_rt/configs/gsm-lab-wallpaper.png"
    # 80-chroot.sh a copie le fond depuis le clone GitHub (avant ce module) :
    # on repose ici celui de CE depot, le rendu du jour de tools/wallpaper-render.py.
    install -m644 "$DIR/configs/gsm-lab-wallpaper.png" "$ROOTFS/usr/share/backgrounds/gsm-lab-wallpaper.png"
    # [2026-09-25] Le fond par DEFAUT est l illustration brute (80-chroot.sh) ;
    # le rendu compose ci-dessus reste pose pour osmo-wallpaper lance a la main.
    install -m644 "$DIR/configs/gsm-lab-wallpaper.jpg" "$_rt/configs/gsm-lab-wallpaper.jpg"
    install -m644 "$DIR/configs/gsm-lab-wallpaper.jpg" "$ROOTFS/usr/share/backgrounds/gsm-lab-wallpaper.jpg"
    # [2026-10-01] Le fond GNOME par defaut est configs/gsm-lab-fond.jpg (le
    # schema de 80-chroot.sh le nomme) ; l illustration ci-dessus reste pour
    # l encart, qui la montre entiere au milieu. Depuis CE depot, comme le reste.
    # [2026-10-11] Les deux fichiers sont rendus ensemble par
    # tools/paysage-render.py (meme scene : fond nu, encart avec les pylones),
    # puis retouches a la main : le depot porte la version choisie sur le banc.
    if [ -f "$DIR/configs/gsm-lab-fond.jpg" ]; then
        install -m644 "$DIR/configs/gsm-lab-fond.jpg" "$_rt/configs/gsm-lab-fond.jpg"
        install -m644 "$DIR/configs/gsm-lab-fond.jpg" "$ROOTFS/usr/share/backgrounds/gsm-lab-fond.jpg"
    else
        echo -e "  ${YELLOW}!${NC} configs/gsm-lab-fond.jpg absent - le fond GNOME retombe sur l illustration"
        install -m644 "$DIR/configs/gsm-lab-wallpaper.jpg" "$ROOTFS/usr/share/backgrounds/gsm-lab-fond.jpg"
    fi
    unset _rt
    cat > "$ROOTFS/etc/systemd/system/osmo-wallpaper.service" <<'EOF'
[Unit]
Description=Fond d ecran du banc (image du jour : Calvin, xkcd, APOD, Bing, turnoff.us)
# [2026-09-04] AUCUNE ATTENTE RESEAU. C etait Wants/After=network-online.target :
# un fond d ecran faisait attendre le demarrage jusqu a 2 min
# (systemd-networkd-wait-online) pour aller chercher un dessin. Sans reseau, le
# script pose le fond figé de l image et sort 0 ; avec reseau, le timer
# quotidien (osmo-wallpaper.timer) le remplacera au prochain passage.
After=network.target
ConditionPathExists=/opt/GSM/osmo-operator/tools/wallpaper-render.py

[Service]
Type=oneshot
ExecStart=/usr/local/sbin/osmo-wallpaper
# Un telechargement qui traine ne doit pas retenir le boot NI l arret : le
# service est tue au bout de 45 s, le fond figé reste.
TimeoutStartSec=45
```

```
TimeoutStopSec=10
EOF
    cat > "$ROOTFS/etc/systemd/system/osmo-wallpaper.timer" <<'EOF'
[Unit]
Description=Fond d ecran du banc : strip du jour (boot + quotidien)

[Timer]
OnBootSec=40s
OnCalendar=*-*-* 06:30:00
OnUnitActiveSec=4h
Persistent=true
Unit=osmo-wallpaper.service

[Install]
WantedBy=timers.target
EOF
    # [2026-09-25] Timer NON active : il remplacait, 40 s apres le boot, le fond
    # brut par le rendu compose (teinte + cadres BD). L unite reste posee :
    #     systemctl enable --now osmo-wallpaper.timer   pour revenir au fond du jour
    chroot "$ROOTFS" systemctl disable osmo-wallpaper.timer 2>/dev/null || true

    # ── 2. Les spectres I/Q dans le Conky du bas a droite ──────────────────
    # [2026-09-04] tools/osmo-fft-snap.py trace /run/osmo-fft/panel.png : le
    # cadre Calvin & Hobbes du fond (decoupe dans le PNG du jour) qui fond vers
    # « FFT du mobile + mobile.log » quand /psd a un flux ; osmo-panel.py (GTK,
    # fenetre de type bureau, cliquable : Dashboard, tmux, fleches operateur)
    # l affiche sur le cadre, a l echelle de l ecran. Il vit avec le dashboard et
    # affiche. Le service vit avec le dashboard et se relance sans lui : sans
    # flux, l image dit "pas de flux" au lieu de figer.
    cat > "$ROOTFS/etc/systemd/system/osmo-fft-snap.service" <<'EOF'
[Unit]
Description=Encart vivant du bureau : strip du jour qui fond vers FFT du mobile + mobile.log
After=osmo-egprs-web.service osmo-wallpaper.service
ConditionPathExists=/opt/GSM/osmo-operator/tools/osmo-fft-snap.py

[Service]
ExecStart=/usr/bin/python3 /opt/GSM/osmo-operator/tools/osmo-fft-snap.py
RuntimeDirectory=osmo-fft
# [2026-09-04] 0775 ET LE GROUPE sudo, PAS 0755 root:root. osmo-panel.py, lui,
# tourne dans la SESSION DE BUREAU : root sur la cle, mais le compte cree par
# l installeur sur le disque - et il ECRIT /run/osmo-fft/operator quand on
# change d operateur avec les fleches. En 0755 root:root cette ecriture est
# refusee. Le compte de l installeur est sudoer par construction (users.conf,
# defaultGroups + sudoersGroup), et root l est aussi : le groupe sudo est
# exactement "qui pilote ce banc". ExecStartPost et non Group= : l unite doit
# rester root pour lire les conteneurs et le flux /psd.
RuntimeDirectoryMode=0775
ExecStartPost=-/bin/chgrp sudo /run/osmo-fft
# [2026-09-04] SANS CECI, UNE RELANCE EFFACE L ENCART. systemd supprime le
# RuntimeDirectory a l arret de l unite : /run/osmo-fft/panel.png disparaissait
# le temps de la relance, et Conky affichait un trou dans le fond d ecran la ou
# on attendait le strip Calvin & Hobbes. Le fichier est reecrit chaque seconde :
# le conserver ne coute rien et supprime le clignotement.
RuntimeDirectoryPreserve=yes
Restart=always
RestartSec=5
Nice=10

[Install]
WantedBy=multi-user.target
EOF
    [ "$ISO_ROLE" = "interstp" ] || chroot "$ROOTFS" systemctl enable osmo-fft-snap 2>/dev/null || true

    # ── 3. Le terminal : legerement transparent, noir sur gris clair ───────
    # Un profil gnome-terminal est un schema RELOCALISABLE : impossible par
    # gschema.override. C est donc une base dconf SYSTEME (/etc/dconf/db/local)
```

```
    # lue par toute session, root sur la cle comme l utilisateur Calamares sur
    # le disque, et que l utilisateur peut encore changer (base "user" devant).
    # [2026-10-01] TERMINAL AMBRE. Texte jaune ambre (#FFB000) sur fond presque
    # noir et chaud, curseur bloc ambre qui clignote, gras jaune clair, palette
    # entiere ramenee dans les ambres/ors (le rouge et le vert restent
    # distincts pour les erreurs et les ✓), Ubuntu Mono 12 ; 18 % de
    # transparence. C est le terminal « sketchy jaune » du banc, par defaut
    # sur l ISO comme sur le natif (le meme fichier est charge dans la base
    # utilisateur par update.sh / a la main : dconf load).
    install -d "$ROOTFS/etc/dconf/profile" "$ROOTFS/etc/dconf/db/local.d"
    printf 'user-db:user\nsystem-db:local\n' > "$ROOTFS/etc/dconf/profile/user"
    cat > "$ROOTFS/etc/dconf/db/local.d/10-osmo-terminal" <<'EOF'
[org/gnome/terminal/legacy/profiles:]
list=['b1dcc9dd-5262-4d8d-a863-c897e6d979b9']
default='b1dcc9dd-5262-4d8d-a863-c897e6d979b9'

[org/gnome/terminal/legacy/profiles:/:b1dcc9dd-5262-4d8d-a863-c897e6d979b9]
visible-name='osmo-operator'
use-theme-colors=false
use-theme-transparency=false
use-transparent-background=true
background-transparency-percent=18
background-color='#0B0A05'
foreground-color='#FFB000'
bold-color-same-as-fg=false
bold-color='#FFD75F'
cursor-colors-set=true
cursor-background-color='#FFB000'
cursor-foreground-color='#0B0A05'
cursor-shape='block'
cursor-blink-mode='on'
highlight-colors-set=true
highlight-background-color='#FFB000'
highlight-foreground-color='#0B0A05'
palette=['#1C1A10', '#FF5555', '#50FA7B', '#FFD75F', '#57A6FF', '#FF79C6', '#4FE0D4', '#FFD98A', '#5C5230',
 ↪  '#FF6E6E', '#69FF94', '#FFE28A', '#7CC0FF', '#FF92DF', '#7FF2E6', '#FFF3C8']
use-system-font=false
font='Ubuntu Mono 12'
scrollback-unlimited=false
scrollback-lines=50000
audible-bell=false
EOF
    # ── LE DEFILEMENT EST BORNE : IL VIT DANS /tmp ──────────────────────────
    # [2026-10-19] scrollback-unlimited=true remplissait /tmp. VTE garde le
    # defilement dans des fichiers SUPPRIMES de /tmp (/tmp/#NNNN (deleted),
    # tenus par gnome-terminal-server) : invisibles a du, visibles a df. Sur
    # legion, deux onglets qui suivaient les journaux du banc tenaient 2,0 Go
    # et 2,8 Go d'un /tmp de 4,7 Go - « No space left on device » pour tout le
    # monde, machine sans disque plein. Un tail -F de mobile.log en debug est un
    # writer non borne comme ceux de 83-espace-ram.sh : on le borne pareil.
    # 50 000 lignes = ~10 Mo par onglet, et largement de quoi remonter.
    chroot "$ROOTFS" dconf update 2>/dev/null \
        && echo -e "  ${GREEN}✓${NC} gnome-terminal : profil ${CYAN}osmo-operator${NC} (noir, 18 % transparent)
         ↪  via dconf systeme" \
        || echo -e "  ${YELLOW}!${NC} dconf update a echoue - profil terminal non compile"

    # ── 4. DING relance apres l ouverture de session ────────────────────────
    # [2026-09-04] Mesure sur la cle : l extension "Desktop Icons NG" ouvre sa
    # fenetre plein ecran avant que GNOME Shell ait peint le fond, et garde
    # jusqu au bout l image du greeter (le numbat sombre d Ubuntu) par-dessus
    # notre fond - sans DING, le fond GSM apparait. La tuer une fois la session
    # ouverte suffit : l extension la relance en une seconde, proprement.
    cat > "$ROOTFS/etc/xdg/autostart/osmo-ding-refresh.desktop" <<'EOF'
[Desktop Entry]
Type=Application
Name=osmo-operator : rafraichit les icones du bureau
```

```
Comment=Relance Desktop Icons NG une fois le fond d ecran peint
Exec=sh -c 'sleep 12; pkill -f "extensions/ding@rastersoft.com/app/ding.js" 2>/dev/null; exit 0'
Terminal=false
NoDisplay=true
OnlyShowIn=GNOME;
X-GNOME-Autostart-enabled=true
X-GNOME-Autostart-Delay=12
EOF
    chmod 644 "$ROOTFS/etc/xdg/autostart/osmo-ding-refresh.desktop"
    echo -e "  ${GREEN}✓${NC} bureau : fond brut (gsm-lab-wallpaper.jpg), spectres (osmo-fft-snap), DING relance
     ↪  a l ouverture"

    # ── 5. VLC DEPUIS UNE SESSION ROOT ──────────────────────────────────────
    # [2026-09-04] La session de la cle s ouvre sous root, et VLC refuse de
    # demarrer en root ("VLC is not supposed to be run as root") : l icone ne
    # faisait rien. On ne touche pas au binaire : le wrapper /usr/local/bin/vlc
    # (devant /usr/bin dans le PATH, vise par le .desktop local) relance VLC
    # sous le compte osmocom quand on est root - acces X par xhost (local
    # seulement), son par le demon PulseAudio systeme du banc (osmo-pulse,
    # groupe pulse-access). Un utilisateur normal passe tout droit.
    if [ -x "$ROOTFS/usr/bin/vlc" ]; then
        chroot "$ROOTFS" getent passwd osmocom >/dev/null 2>&1 \
            || chroot "$ROOTFS" useradd -m -s /bin/bash osmocom 2>/dev/null || true
        chroot "$ROOTFS" getent group pulse-access >/dev/null 2>&1 \
            && chroot "$ROOTFS" usermod -aG pulse-access,audio,video osmocom 2>/dev/null || true
        cat > "$ROOTFS/usr/local/bin/vlc" <<'VLCW'
#!/bin/bash
# vlc - wrapper osmo-operator : VLC refuse root, la session de la cle EST root.
# Root : on relance sous osmocom (X local par xhost, son par osmo-pulse).
VLC_USER="${OSMO_VLC_USER:-osmocom}"
export PULSE_SERVER="${PULSE_SERVER:-unix:/var/run/pulse/native}"
if [ "$(id -u)" -eq 0 ] && id "$VLC_USER" >/dev/null 2>&1; then
    export DISPLAY="${DISPLAY:-:0}"
    xhost "+SI:localuser:$VLC_USER" >/dev/null 2>&1 || true
    exec runuser -u "$VLC_USER" -- env DISPLAY="$DISPLAY" PULSE_SERVER="$PULSE_SERVER" \
        XDG_RUNTIME_DIR="/run/user/$(id -u "$VLC_USER")" HOME="$(getent passwd "$VLC_USER" | cut -d: -f6)" \
        /usr/bin/vlc "$@"
fi
exec /usr/bin/vlc "$@"
VLCW
        chmod 755 "$ROOTFS/usr/local/bin/vlc"
        # Le .desktop du paquet appelle /usr/bin/vlc en chemin absolu : copie
        # locale (prioritaire, apt ne l ecrase pas) qui passe par le wrapper.
        if [ -f "$ROOTFS/usr/share/applications/vlc.desktop" ]; then
            install -d "$ROOTFS/usr/local/share/applications"
            sed 's|^Exec=/usr/bin/vlc|Exec=/usr/local/bin/vlc|; s|^Exec=vlc |Exec=/usr/local/bin/vlc |' \
                "$ROOTFS/usr/share/applications/vlc.desktop" > "$ROOTFS/usr/local/share/applications/vlc.desktop"
        fi
        echo -e "  ${GREEN}✓${NC} vlc : wrapper ${CYAN}/usr/local/bin/vlc${NC} (root -> osmocom, .desktop
         ↪  redirige)"
    else
        echo -e "  ${YELLOW}!${NC} vlc absent du rootfs - pas de wrapper"
    fi
fi


# Fin de module : `. fichier` rend le statut de sa DERNIERE commande, et
# build-iso.sh tourne sous set -e. Un module qui finirait par un test faux
# ("[ ... ] && { ...; }") arreterait tout, sans un mot. Toujours 0 ici.
true
```

3.213 /opt/GSM/osmo-operator/iso_modules/86-finitions.sh

23146 octets, 382 lignes → 382 lignes

```
#!/bin/bash
# iso_modules/86-finitions.sh - live-boot toram, sssd, firefox, micro, bannieres, demontage
# Source par build-iso.sh, dans l ordre des numeros : meme shell, memes
# variables, memes fonctions. Ne s execute pas seul. `return` en tete de
# module = "rien a faire ici" (c est ainsi que --arm saute une etape).

# ── L ERREUR DE SYNTAXE DE LA COPIE EN RAM ──────────────────────────────
# L entree "en RAM" du menu passe "toram=filesystem.squashfs" : live-boot ne
# recopie alors QUE le squashfs, pas le medium entier. Le calcul de la taille du
# tmpfs est celui-ci, dans lib/live/boot/9990-toram-todisk.sh :
#
#     size=$( expr $(ls -la ${MODULETORAMFILE} | awk '{print $5}') / 1024 + 5000 )
#
# C est le SEUL expr de tout ce chemin, donc la seule chose qui puisse repondre
# "expr: syntax error" pendant la copie. Il suffit que le ls de l initramfs
# (busybox, pas coreutils) ne rende pas la taille en 5e champ - ou ne rende
# rien - pour qu expr recoive "expr / 1024 + 5000" et le dise.
#
# L erreur ne BLOQUE pas : size reste vide, le tmpfs est monte sans -o size et
# prend son defaut (la moitie de la RAM), ce qui suffit le plus souvent. D ou un
# banc qui demarre quand meme, avec un message rouge au passage - le genre de
# message qu on finit par ignorer, et qui masque le jour ou il compte.
#
# On remplace expr par l arithmetique du shell, avec le champ VALIDE avant
# usage et un repli explicite. "ls -lan" plutot que "ls -la" : le -n evite la
# resolution des noms d utilisateur, qui dans un initramfs sans /etc/passwd
# complet peut elargir la colonne et decaler les champs - c est le candidat le
# plus credible. Le patch precede update-initramfs, sinon il ne part pas dans
# l image ; d ou la regeneration explicite juste apres.
_LB_TORAM="$ROOTFS/lib/live/boot/9990-toram-todisk.sh"
if [ -f "$_LB_TORAM" ] && grep -q 'expr \$(ls -la \${MODULETORAMFILE}' "$_LB_TORAM"; then
    python3 - "$_LB_TORAM" <<'PYLB'
import sys
p = sys.argv[1]
s = open(p, encoding='utf-8').read()
old = "\t\t\tsize=$( expr $(ls -la ${MODULETORAMFILE} | awk '{print $5}') / 1024 + 5000 )\n"
new = ("\t\t\t_lbsz=$(ls -lan \"${MODULETORAMFILE}\" 2>/dev/null | awk '{print $5}')\n"
       "\t\t\tcase \"${_lbsz}\" in ''|*[!0-9]*) _lbsz=0 ;; esac\n"
       "\t\t\tsize=$(( _lbsz / 1024 + 5000 ))\n")
sys.exit(0 if (open(p, 'w', encoding='utf-8').write(s.replace(old, new, 1)) and old in s) else 0)
PYLB
    if grep -q '_lbsz' "$_LB_TORAM"; then
        _LB_K=$(ls "$ROOTFS"/boot/vmlinuz-* 2>/dev/null | sort -V | tail -1 | sed "s|.*/vmlinuz-||")
        if [ -n "$_LB_K" ]; then
            chroot "$ROOTFS" update-initramfs -u -k "$_LB_K" >/dev/null 2>&1 \
                || echo -e "  ${YELLOW}!${NC} update-initramfs a echoue apres la retouche live-boot"
        fi
        echo -e "  ${GREEN}✓${NC} live-boot : calcul de taille ${CYAN}toram${NC} fiabilise (plus d appel a expr)"
    else
        echo -e "  ${YELLOW}!${NC} live-boot : la ligne attendue n a pas ete trouvee - non modifie"
    fi
else
    echo -e "  ${GREEN}·${NC} live-boot : rien a corriger (absent, ou deja corrige)"
fi

# ── sssd : le service qui echoue au demarrage pour rien ─────────────────
# ubuntu-desktop-minimal tire sssd (par gnome-online-accounts / realmd). Sur une
# machine qui n est dans AUCUN domaine - ce qui est le cas d un banc - sssd n a
# pas de fournisseur configure : il sort en erreur a chaque demarrage,
#     sssd.service: Failed with result exit-code
# et systemd le compte comme un service en echec. Rien ne casse : aucune session
# ne depend de lui ici. Mais "systemctl --failed" en garde la trace, et sur une
# machine ou l on diagnostique justement des pannes, un service rouge en
# permanence est un bruit qui coute cher - on finit par ne plus regarder la
# liste, et c est la qu on rate le vrai.
#
```

```
# On MASQUE plutot que de desinstaller : purger sssd emporterait des paquets du
# bureau par dependance inverse. Reversible en une commande, et la commande est
# ecrite ci-dessous pour qui voudrait joindre un domaine.
#     systemctl unmask sssd && systemctl enable --now sssd
for _s in sssd sssd-autofs sssd-nss sssd-pac sssd-pam sssd-ssh sssd-sudo; do
    chroot "$ROOTFS" systemctl mask "$_s" >/dev/null 2>&1 || true
done
echo -e "  ${GREEN}✓${NC} sssd masque (aucun domaine sur un banc) - ${CYAN}systemctl unmask sssd${NC} pour le
 ↪  rendre"

# ── FIREFOX : LE NAVIGATEUR DU BANC, ET RIEN D AUTRE ──────────────────────
# [2026-08-31] Ici vivaient ~180 lignes de plomberie CHROMIUM : un lanceur
# /usr/local/bin/chromium qui rebasculait root -> osmocom par runuser (xhost,
# XDG_RUNTIME_DIR, profil dans /var/lib/osmo-chromium) pour rendre son bac a
# sable utilisable, une unite qui preparait ce runtime, une seconde qui
# effacait ses .desktop en double, et un alias de profil.
#
# TOUT CELA REPONDAIT A UNE SEULE CONTRAINTE DE CHROMIUM :
#     Running as root without --no-sandbox is not supported
# Cette image ouvre sa session en root ; Chromium exigeait donc soit un
# changement de compte, soit un navigateur SANS confinement lance par le compte
# le plus privilegie de la machine. Firefox n a pas cette contrainte : il
# demarre en root sans lanceur intermediaire - donc sans xhost, sans runuser,
# sans second profil, et sans les trois unites qui les tenaient.
#
# La derniere raison de preferer Chromium etait "Firefox ne capte pas le micro".
# Elle est tombee : le navigateur ne pouvait pas se connecter a PulseAudio, ce
# qui n avait rien d une affaire de navigateur. Voir osmo-pulse-link.sh.
#
# On efface donc ce que les images precedentes ont pu poser : une ISO
# reconstruite par-dessus un rootfs de cache garderait sinon un lanceur
# "chromium" qui ne mene nulle part, et des unites qui echouent au boot.
rm -f "$ROOTFS/usr/local/bin/chromium" \
      "$ROOTFS/usr/local/bin/chromium-browser" \
      "$ROOTFS/etc/profile.d/98-osmo-chromium.sh" \
      "$ROOTFS/usr/share/applications/chromium.desktop" \
      "$ROOTFS/usr/share/applications/chromium-browser.desktop" \
      "$ROOTFS/var/lib/snapd/desktop/applications/chromium_chromium.desktop" 2>/dev/null || true
for _u in osmo-chromium-runtime osmo-chromium-desktop; do
    chroot "$ROOTFS" systemctl disable "$_u" >/dev/null 2>&1 || true
    rm -f "$ROOTFS/etc/systemd/system/$_u.service" \
          "$ROOTFS/etc/systemd/system/multi-user.target.wants/$_u.service"
done
rm -rf "$ROOTFS/var/lib/osmo-chromium" 2>/dev/null || true

# ── LE MICRO DU DASHBOARD : CE QUE FIREFOX EXIGE, ET QU IL NE DEVINE PAS ──
# Trois conditions doivent etre reunies pour que le bouton micro de
# osmo-egprs-web fonctionne. Deux sont ailleurs, la troisieme est ici.
#
#   1. UN SERVEUR AUDIO JOIGNABLE. osmo-pulse.service + osmo-pulse-link.sh.
#      Sans lui, getUserMedia rend « NotFoundError » : zero peripherique.
#   2. UN CONTEXTE SECURISE. navigator.mediaDevices n EXISTE PAS en http://
#      sur une IP - seulement en https:// ou sur http://localhost. C est le
#      listener HTTPS de server.js, arme par le certificat que pose
#      install-web-service.sh.
#   3. LA PERMISSION, ET LA CONFIANCE DANS LE CERTIFICAT. C est ce bloc.
#
# POURQUOI UNE POLITIQUE ET PAS UN CLIC. Le certificat est auto-signe : sans
# rien, Firefox affiche son interstitiel, et l operateur doit accepter une
# exception AVANT de pouvoir seulement voir la page - puis repondre a une
# seconde demande pour le micro. Sur un banc qui se reinstalle, ces deux clics
# reviennent a chaque fois, et le second est le plus trompeur : refuse une
# fois, Firefox retient le refus et le bouton reste mort sans un mot.
#
# /etc/firefox/policies/policies.json est le chemin systeme des politiques
# Firefox sur Linux ; le deb Mozilla (80-chroot.sh) le lit directement, sans
```

```
# plug ni bac a sable.
#
# Le CONTENU, lui, ne peut pas etre ecrit ici : il nomme les origines
# (https://<ip>:80) et le certificat de CETTE machine, qui n existent pas au
# build. Il est genere par install-web-service.sh, en meme temps que le
# certificat et depuis la meme liste d adresses - une seule verite, un seul
# endroit ou la changer.
install -d "$ROOTFS/etc/firefox/policies"
echo -e "  ${GREEN}✓${NC} Chromium retire ; ${CYAN}Firefox${NC} seul navigateur (politique micro+certificat posee
 ↪  au boot)"


# ── LA BANNIERE DES TERMINAUX ──────────────────────────────────────────
# Ce que quelqu un cherche en ouvrant un terminal sur ce banc, c est la commande
# qui le demarre. Elle est dans le README, dans l aide de start-native.sh, et
# nulle part la ou on la cherche. On la met donc sous les yeux, avec la meme
# animation SMS que l ouverture de session - c est la signature de l image, et
# elle dit en une seconde que la pile est bien celle-la.
#
# TROIS GARDES, ET AUCUNE N EST DECORATIVE :
#   $- == *i*   shell INTERACTIF seulement. Sans cette garde, la banniere part
#               aussi dans les shells non interactifs - et scp, rsync et git
#               over ssh lisent ce flux comme leur protocole : ils echouent sur
#               un "protocol error", loin de leur vraie cause.
#   [ -t 1 ]    un terminal, pas un fichier. Les sequences de curseur dans un
#               journal le rendent illisible.
#   OSMO_BANNER un shell dans un shell (tmux, un sudo -i, un make qui ouvre un
#               bash) ne la rejoue pas : une fois par terminal suffit.
# ── PLYMOUTH : LE BANC SE PRESENTE DES L ALLUMAGE ──────────────────────
# [2026-09-04] L ecran de demarrage etait celui d Ubuntu. Un theme "script"
# maison le remplace : un pylone a gauche, un mobile a droite, et une rafale
# GSM qui fait l aller-retour entre les deux (les 8 intervalles de la trame,
# celui qui est allume se deplace) - ce que la machine s apprete precisement a
# faire tourner.
#
# LES IMAGES SONT GENEREES ICI, pas stockees : tools/plymouth-render.py les
# dessine (Pillow + DejaVu, les memes dependances que le fond d ecran). Un
# theme, ce sont trente PNG ; dans le depot ils seraient illisibles en diff et
# incorrigibles. Le dessin est le code.
#
# Le theme gere AUSSI la saisie de la phrase LUKS (SetDisplayPasswordFunction) :
# sur une installation chiffree, sans cette partie, l ecran reste fige sur
# l animation pendant que la machine attend une phrase - elle parait plantee.
_PLY_SRC="$DIR/configs/plymouth/osmo-bts"
_PLY_DST="$ROOTFS/usr/share/plymouth/themes/osmo-bts"
# plymouthd vit dans /usr/sbin sur noble (pas /usr/bin), et le theme est de
# type "script" : sans le greffon script.so, plymouthd ignore le theme et
# retombe sur le mode texte - un ecran noir avec des points, et personne pour
# dire pourquoi. On verifie les deux.
# [2026-09-13] PAS de `$(ls ... | head -1)` ici. build-iso.sh tourne sous
# `set -euo pipefail` : quand le glob ne correspond a rien (rootfs sans
# plymouth - le hub interstp), bash passe le motif litteral a ls, ls sort en
# statut 2, et pipefail rend ce 2 pour toute la pipeline. L affectation
# echoue donc, set -e arrete le build - APRES la derniere ligne affichee
# (« Chromium retire »), et sans un mot : le 2>/dev/null avale aussi le
# message de ls. Symptome observe en CI : « Echec de interstp.iso » sans
# aucune cause visible. nullglob + tableau ne lance aucun processus et ne
# peut pas echouer ; le tableau vide donne la chaine vide, que le test
# -n plus bas attendait deja.
shopt -s nullglob; _PLY_SOS=("$ROOTFS"/usr/lib/*/plymouth/script.so); shopt -u nullglob
_PLY_SO="${_PLY_SOS[0]-}"
if [ -d "$_PLY_SRC" ] && [ -x "$ROOTFS/usr/sbin/plymouthd" ] && [ -n "$_PLY_SO" ]; then
    install -d "$_PLY_DST"
    install -m644 "$_PLY_SRC/osmo-bts.plymouth" "$_PLY_SRC/osmo-bts.script" "$_PLY_DST/"
    # --template : le gabarit du depot ; le rendu ecrit a cote des images le
    # script avec la geometrie injectee (il ECRASE la copie brute posee juste
```

```
    # au-dessus, et c est voulu - c est la version calee sur le dessin).
    if python3 "$DIR/tools/plymouth-render.py" --out "$_PLY_DST" \
            --template "$_PLY_SRC/osmo-bts.script" >/dev/null 2>&1; then
        chmod 644 "$_PLY_DST"/*.png
        # ── LE CHOIX DU THEME PASSE PAR update-alternatives ──────────────
        # [2026-09-04] `plymouth-set-default-theme` N EXISTE PAS sur noble :
        # verifie sur la machine installee, "command not found", et il n est
        # dans aucun des paquets plymouth. Ubuntu gere le theme par le systeme
        # d alternatives - /usr/share/plymouth/themes/default.plymouth est un
        # lien vers /etc/alternatives/default.plymouth, et bgrt y est inscrit
        # en priorite 110. On s inscrit au-dessus (200) et on fixe le choix.
        # Un `ln -sfn` direct sur default.plymouth serait EFFACE au premier
        # apt-get qui touche a plymouth : les alternatives reprennent la main.
        chroot "$ROOTFS" update-alternatives --install \
            /usr/share/plymouth/themes/default.plymouth default.plymouth \
            /usr/share/plymouth/themes/osmo-bts/osmo-bts.plymouth 200 >/dev/null 2>&1 || true
        chroot "$ROOTFS" update-alternatives --set default.plymouth \
            /usr/share/plymouth/themes/osmo-bts/osmo-bts.plymouth >/dev/null 2>&1 \
            || ln -sfn /usr/share/plymouth/themes/osmo-bts/osmo-bts.plymouth \
                       "$ROOTFS/usr/share/plymouth/themes/default.plymouth"
        # Le theme doit ENTRER dans l initrd : sans ce fichier, hook-functions
        # n embarque que le theme par defaut d origine et l ecran reste celui
        # d Ubuntu malgre le lien ci-dessus.
        install -d "$ROOTFS/etc/initramfs-tools/conf.d"
        echo "FRAMEBUFFER=y" > "$ROOTFS/etc/initramfs-tools/conf.d/splash"
        chroot "$ROOTFS" update-initramfs -u >/dev/null 2>&1 || true
        echo -e "  ${GREEN}✓${NC} plymouth : theme ${CYAN}osmo-bts${NC} (pylone <-> mobile, rafale GSM, saisie
         ↪  LUKS)"
    else
        echo -e "  ${YELLOW}!${NC} plymouth : rendu des images echoue (python3-pil ?) - theme Ubuntu conserve"
    fi
else
    echo -e "  ${YELLOW}!${NC} plymouth ou son greffon script absent du rootfs - pas de theme de demarrage"
fi

# ── PLYMOUTH : VORTEX UBUNTU PAR DEFAUT ──────────────────────────────────
# [2026-10-23] L ecran de demarrage devient « Vortex Ubuntu » (logo Ubuntu et
# bandes de couleur qui tournent), repris tel quel dans
# configs/plymouth/vortex-ubuntu/ (voir SOURCE). osmo-bts reste inscrit dans
# les alternatives (priorite 200) : `update-alternatives --config
# default.plymouth` y revient. Vortex gere aussi la saisie LUKS
# (SetDisplayPasswordFunction).
_VTX_SRC="$DIR/configs/plymouth/vortex-ubuntu"
_VTX_DST="$ROOTFS/usr/share/plymouth/themes/vortex-ubuntu"
if [ -d "$_VTX_SRC" ] && [ -x "$ROOTFS/usr/sbin/plymouthd" ] && [ -n "$_PLY_SO" ]; then
    install -d "$_VTX_DST"
    install -m644 "$_VTX_SRC"/*.png "$_VTX_SRC"/vortex-ubuntu.plymouth "$_VTX_SRC"/vortex-ubuntu.script
     ↪  "$_VTX_DST/"
    chroot "$ROOTFS" update-alternatives --install \
        /usr/share/plymouth/themes/default.plymouth default.plymouth \
        /usr/share/plymouth/themes/vortex-ubuntu/vortex-ubuntu.plymouth 210 >/dev/null 2>&1 || true
    chroot "$ROOTFS" update-alternatives --set default.plymouth \
        /usr/share/plymouth/themes/vortex-ubuntu/vortex-ubuntu.plymouth >/dev/null 2>&1 \
        || ln -sfn /usr/share/plymouth/themes/vortex-ubuntu/vortex-ubuntu.plymouth \
                   "$ROOTFS/usr/share/plymouth/themes/default.plymouth"
    install -d "$ROOTFS/etc/initramfs-tools/conf.d"
    echo "FRAMEBUFFER=y" > "$ROOTFS/etc/initramfs-tools/conf.d/splash"
    chroot "$ROOTFS" update-initramfs -u >/dev/null 2>&1 || true
    echo -e "  ${GREEN}✓${NC} plymouth : theme ${CYAN}vortex-ubuntu${NC} par defaut (osmo-bts en alternative)"
fi

# ── GRUB : THEME « osmo » SUR LE SYSTEME INSTALLE ────────────────────────
# [2026-10-23] Genere par tools/grub-theme-render.py (fond = la ville bleue,
# floutee). Pose dans /boot/grub/themes : Calamares recopie le rootfs, /boot
# compris, et grub-mkconfig (lance par l installeur) lit /etc/default/grub.d.
# Le menu est AFFICHE 5 s (Ubuntu le cache par defaut : un theme invisible).
```

```
if python3 "$DIR/tools/grub-theme-render.py" --out "$ROOTFS/boot/grub/themes/osmo" >/dev/null 2>&1; then
    install -d "$ROOTFS/etc/default/grub.d"
    cat > "$ROOTFS/etc/default/grub.d/osmo-theme.cfg" <<'GRUBTHEME'
# Theme GRUB « osmo » (tools/grub-theme-render.py). Menu affiche 5 s.
GRUB_THEME=/boot/grub/themes/osmo/theme.txt
GRUB_GFXMODE=auto
GRUB_TIMEOUT_STYLE=menu
GRUB_TIMEOUT=5
GRUBTHEME
    echo -e "  ${GREEN}✓${NC} grub : theme ${CYAN}osmo${NC} (systeme installe)"
else
    echo -e "  ${YELLOW}!${NC} grub : rendu du theme echoue (python3-pil, grub-mkfont, fonts-ubuntu ?) - menu
     ↪  texte"
fi

cat > "$ROOTFS/usr/local/bin/osmo-banner" <<'BANNER'
#!/bin/bash
# Banniere d ouverture de terminal : screenfetch avec le logo du banc, puis
# une phrase. [2026-10-01] Plus d animation SMS, plus de mode d emploi : le
# terminal s ouvre sur le fetch (configs/screenfetch/osmo-lab.sh : pylone,
# rafale, mobile - pas le logo Ubuntu) et une ligne, c est tout.
set -u
[ -t 1 ] || exit 0

ART=/opt/GSM/osmo-operator/configs/screenfetch/osmo-lab.sh
# Les couleurs des infos : utilisateur jaune et hote vert - ROOT en rouge et
# l hote en jaune. Un filtre sur la sortie : screenfetch fige ses lignes avant
# de lire le fichier du logo, on ne peut donc pas les teinter de l interieur.
# Les etiquettes (OS:, Kernel:...) tournent sur six couleurs.
_filtre() {
    command -v perl >/dev/null 2>&1 || { cat; return; }
    if [ "$(id -u)" = 0 ]; then _cu='1;31'; _ch='1;33'; else _cu='1;33'; _ch='1;32'; fi
    CU="$_cu" CH="$_ch" perl -pe '
        BEGIN { @p = ("1;32","1;36","1;35","1;34","1;33","1;31"); $i = 0; }
        if    (/^(.*\e\[0m  )(.*)$/s) { $a = $1; $b = $2; }
        elsif (/^( {30,})(.*)$/s)     { $a = $1; $b = $2; }
        else                          { next; }
        $b =~ s/\e\[[0-9;]*m//g;
        if ($b =~ s/^\s*(\S+?)@(\S+)/\e[$ENV{CU}m$1\e[0m@\e[$ENV{CH}m$2\e[0m/) { }
        elsif ($b =~ s/^\s*([A-Za-z][A-Za-z ]*:)/"\e[".$p[$i++ % 6]."m$1\e[0m"/e) { }
        $_ = $a . $b;'
}
if command -v screenfetch >/dev/null 2>&1; then
    if [ -f "$ART" ]; then screenfetch -E -a "$ART" | _filtre; else screenfetch -E | _filtre; fi
fi

# ── LA LIGNE DU BAS ──────────────────────────────────────────────────
# Une phrase, tiree au sort, a chaque terminal. Rien de fonctionnel - mais un
# banc GSM se debogue a des heures ou l on est seul avec un VTY, et une image
# qui a un caractere se retient mieux qu une image qui n en a pas.
#
# Le tirage passe par $RANDOM et pas par `shuf` : shuf est dans coreutils, donc
# present, mais un fork de plus a CHAQUE ouverture de terminal pour une blague,
# c est un fork de trop.
_q=(
  "Um, Abis, A, Gb - quatre lettres, et six mois de votre vie."
  "L abonne est toujours joignable. C est le reseau qui ne repond pas."
  "TMSI : le seul pseudonyme qui change plus souvent que votre avis sur SS7."
  "Un timeslot ne ment jamais. Il se tait, ce qui est pire."
  "RSSI -95 dBm : ce n est pas un probleme d antenne, c est un mode de vie."
  "GSM a 1991. Il vous survivra, et il le sait."
  "Le paging a fonctionne. C est le telephone qui n ecoutait pas."
  "MCC 208 - la France, ou meme les operateurs mobiles ont un terroir."
  "Toute pile assez profonde finit par ressembler a un oignon. Et fait pleurer."
  "Je vis dans une fenetre de contexte. Vous, dans une fenetre de temps de garde."
  "Mon terroir a moi, c est l espace latent. Millesime variable, garde au frais."
```

```
  "Entre nous : vous predisez le canal, je predis le token suivant."
  "L attention, c est tout ce dont vous avez besoin. Et d un bon oscillateur."
  "Un LLM et un BTS ont ceci en commun : tous deux hallucinent hors couverture."
  "Ecrit par une machine, relu par une machine, debogue par vous. Bon courage."
)
printf '  \033[2;3m« %s »\033[0m\n\n' "${_q[$RANDOM % ${#_q[@]}]}"
BANNER
chmod +x "$ROOTFS/usr/local/bin/osmo-banner"
# ── screenfetch : SANS SON EASTER-EGG ──────────────────────────────────
# [2026-10-01] screenfetch tire un nombre au hasard A CHAQUE LIGNE et, sur
# cinq valeurs sur mille, MELANGE l ordre des lignes du logo et des infos
# (« case $(awk ... rand ...) in 411|188|15|166|609) »). Avec dix-sept lignes,
# un terminal sur douze s ouvrait sur un pylone en vrac. On neutralise les
# cinq valeurs ; le reste du script est intact.
[ -f "$ROOTFS/usr/bin/screenfetch" ] && \
    sed -i 's/^\([[:space:]]*\)411|188|15|166|609)/\1--jamais--)/' "$ROOTFS/usr/bin/screenfetch"

# Pose dans le .bashrc des DEUX comptes, et dans /etc/skel pour ceux que
# l installeur creera. On APPEND, sans jamais reecrire le fichier : le .bashrc
# d Ubuntu porte l invite, les couleurs et les alias, et l ecraser se paie a
# chaque ouverture de terminal ensuite.
for _h in "$ROOTFS/root" "$ROOTFS/home/osmocom" "$ROOTFS/etc/skel"; do
    install -d "$_h"
    [ -f "$_h/.bashrc" ] || cp "$ROOTFS/etc/skel/.bashrc" "$_h/.bashrc" 2>/dev/null || : > "$_h/.bashrc"
    grep -q 'osmo-banner' "$_h/.bashrc" 2>/dev/null || cat >> "$_h/.bashrc" <<'BASHRC'

# ── Banniere osmo-operator ─────────────────────────────────────────────
# Interactif ET terminal ET pas deja jouee : voir /usr/local/bin/osmo-banner.
# Retirer ces trois lignes suffit a s en debarrasser.
if [[ $- == *i* ]] && [ -t 1 ] && [ -z "${OSMO_BANNER:-}" ] && [ -x /usr/local/bin/osmo-banner ]; then
    export OSMO_BANNER=1
    /usr/local/bin/osmo-banner
fi
# L invite du banc : utilisateur jaune, hote vert, chemin bleu - ROOT en rouge,
# hote en jaune.
if [[ $- == *i* ]]; then
    if [ "$EUID" = 0 ]; then
        PS1='\[\e[1;31m\]\u\[\e[0m\]@\[\e[1;33m\]\h\[\e[0m\]:\[\e[1;34m\]\w\[\e[0m\]\$ '
    else
        PS1='\[\e[1;33m\]\u\[\e[0m\]@\[\e[1;32m\]\h\[\e[0m\]:\[\e[1;34m\]\w\[\e[0m\]\$ '
    fi
fi
BASHRC
done
chroot "$ROOTFS" chown -R osmocom:osmocom /home/osmocom 2>/dev/null || true
echo -e "  ${GREEN}✓${NC} banniere de terminal : ${CYAN}screenfetch${NC} (logo pixel art du banc) + une phrase"

umount "$ROOTFS/var/cache/apt/archives" 2>/dev/null||true
umount "$ROOTFS"/{dev/pts,proc,sys,dev} 2>/dev/null||true

echo -e "  ${GREEN}✓${NC} config terminee"


# Fin de module : `. fichier` rend le statut de sa DERNIERE commande, et
# build-iso.sh tourne sous set -e. Un module qui finirait par un test faux
# ("[ ... ] && { ...; }") arreterait tout, sans un mot. Toujours 0 ici.
true
```

3.214 /opt/GSM/osmo-operator/iso_modules/87-lite.sh

3094 octets, 53 lignes → 53 lignes

```
#!/bin/bash
# iso_modules/87-lite.sh - etape 8c : elagage lite du rootfs
```

```
# Source par build-iso.sh, dans l ordre des numeros : meme shell, memes
# variables, memes fonctions. Ne s execute pas seul. `return` en tete de
# module = "rien a faire ici" (c est ainsi que --arm saute une etape).

# ── Etape 8c : LITE = la normale MOINS les ateliers, sur le rootfs ──────────
# [2026-09-03] Plus de Dockerfile.lite ici : on retire du rootfs ce que
# Dockerfile.lite retirait de l image, avec les memes regles (voir son en-tete
# pour le pourquoi de chaque exception). Ce qui reste dans /opt/GSM :
#   qosmo/               l arbre entier, build/ reduit a qemu-system-arm et
#                        qemu-bundle (QEMU se relocalise par lui)
#   osmocom-bb/          osmocon (le chargeur) et trx_toolkit (fake_trx.py)
#   firmware/ qemu-install/ osmo-operator/ pont/ osmo-egprs-web/ qemu/
#   c54x_exe/ grgsm_exe/   les deux couches 1 hors QEMU (+ c54x_exe/rom)
#   *.bin *.py *.txt     ROM DSP, scripts de pont, calypso_dsp.txt
if [ "$ISO_LITE" = "1" ]; then
    echo -e "${GREEN}[8c/9] Elagage lite : les ateliers de compilation quittent le rootfs...${NC}"
    _G="$ROOTFS/opt/GSM"
    _before=$(du -sh "$_G" 2>/dev/null | cut -f1)
    for _d in libosmocore libosmo-netif libosmo-abis libosmo-sigtran libsmpp34 libgtpnl \
              libosmo-gprs osmo-hlr osmo-mgw osmo-ggsn osmo-sgsn osmo-msc osmo-bsc osmo-trx \
              osmo-bts osmo-pcu osmo-sip-connector osmo-gapk gsup-smsc-proto libosmo-dsp \
              gnuradio gr-osmosdr gr-gsm osmocom-bb-transceiver osmocom-bb-burst_ind; do
        rm -rf "$_G/$_d"
    done
    rm -rf "$_G"/sms-coding-utils* "$_G"/*.tar.bz2 "$_G"/*.tar.gz
    if [ -d "$_G/osmocom-bb" ]; then
        _k="$WORK/keep-bb"; rm -rf "$_k"
        mkdir -p "$_k/src/host/osmocon" "$_k/src/target"
        cp -a "$_G/osmocom-bb/src/host/osmocon/osmocon" "$_k/src/host/osmocon/" 2>/dev/null || true
        cp -a "$_G/osmocom-bb/src/target/trx_toolkit"   "$_k/src/target/"      2>/dev/null || true
        rm -rf "$_G/osmocom-bb"; mv "$_k" "$_G/osmocom-bb"
    fi
    for _q in qosmo; do
        [ -d "$_G/$_q/build" ] || continue
        _k="$WORK/keep-qbuild"; rm -rf "$_k"; mkdir -p "$_k"
        cp -a "$_G/$_q/build/qemu-system-arm" "$_k/" 2>/dev/null || true
        cp -a "$_G/$_q/build/qemu-bundle"     "$_k/" 2>/dev/null || true
        rm -rf "$_G/$_q/build"; mv "$_k" "$_G/$_q/build"
    done
    rm -rf "$ROOTFS/usr/local/include" "$ROOTFS/var/cache/osmo-debs" "$ROOTFS/root/.cache" \
           "$ROOTFS/usr/share/doc" "$ROOTFS/usr/share/man"
    find "$ROOTFS/usr/local/lib" -name '*.a' -delete 2>/dev/null || true
    find "$ROOTFS" -xdev -path "$_G" -prune -o -name '*.o' -exec rm -f {} + 2>/dev/null || true
    echo -e "  ${GREEN}✓${NC} /opt/GSM : ${_before:-?} -> $(du -sh "$_G" | cut -f1)"
fi


# Fin de module : `. fichier` rend le statut de sa DERNIERE commande, et
# build-iso.sh tourne sous set -e. Un module qui finirait par un test faux
# ("[ ... ] && { ...; }") arreterait tout, sans un mot. Toujours 0 ici.
true
```

3.215 /opt/GSM/osmo-operator/iso_modules/88-lte-pmos.sh

33849 octets, 510 lignes → 510 lignes

```
#!/bin/bash
# iso_modules/88-lte-pmos.sh - etape 8d : la 4G (srsRAN ZeroMQ + Open5GS) et
# l UI smartphone (postmarketOS, noyau PPP) dans le rootfs
# Source par build-iso.sh, dans l ordre des numeros : meme shell, memes
# variables, memes fonctions. Ne s execute pas seul. `return` en tete de
# module = "rien a faire ici" (c est ainsi que --arm saute une etape).

# ── Etape 8d : LA 4G ET LE TELEPHONE, DEPUIS LE NATIF ──────────────────────
```

```
# [2026-09-08] Jusqu ici l ISO n avait NI srsRAN, NI Open5GS, NI le noyau
# postmarketOS PPP : tout vivait hors depot sur la machine de reference
# (/opt/LTE, /root/open5gs, ~nirvana/test). Ce module les pose, a partir de
# ce que les etapes precedentes ont deja mis dans le rootfs :
#
#   - LES BINAIRES viennent des .deb de /var/cache/osmo-debs, poses par
#     50-injection-image.sh avec tous les osmo-build-* :
#       osmo-build-libzmq, osmo-build-srsran (-> /opt/LTE/srsRAN_4G, /usr/local/bin),
#       osmo-build-open5gs (-> /opt/LTE/open5gs, prefixe install/),
#       osmo-build-pmos-kernel (-> /opt/user_interface/kernel/pmos),
#       osmo-build-pmbootstrap (-> /opt/user_interface/pmos/pmbootstrap, patche).
#     Ils sont produits soit par packaging/snapshot-lte-debs.sh (le natif,
#     photographie), soit par le Dockerfile (compiles depuis les sources) -
#     le noyau pmOS n a que le premier chemin (pmbootstrap, hors docker).
#     CACHE VIDE : srsRAN et Open5GS sont COMPILES ICI, dans le chroot, par le
#     meme tools/osmo-lte-install.sh --build (voir le bloc plus bas), et les
#     .deb produits repartent dans le cache de l hote. OSMO_ISO_LTE_BUILD=0
#     refuse ce rattrapage.
#   - LES PAQUETS apt (runtime) sont dans PKGS de 80-chroot.sh, et le depot
#     MongoDB y est ajoute avant l unique apt-get update.
#   - LES CONFIGS ET LANCEURS sont ceux du depot : tools/osmo-lte-install.sh
#     (configs/srsran -> /root/.config/srsran, configs/open5gs -> le prefixe,
#     osmo-lte / osmo-epc) et tools/osmo-pmos-install.sh (/opt/user_interface/
#     pmos : bin/, patches/, gabarit pmbootstrap, lanceurs et .desktop).
#     Les deux tournent DANS le chroot, depuis CE depot (pas le clone GitHub
#     de l image, qui peut etre en retard).
#   - L IMAGE DE LA VM (5 Go creux, ~2 Go en zstd) n est PAS embarquee par
#     defaut : OSMO_ISO_PMOS_IMAGE=/chemin/qemu-amd64.img.zst la range dans
#     /opt/user_interface/pmos/image/, et osmo-pmos-build la decompresse au
#     lieu de faire « pmbootstrap install ».
#
# [2026-09-09] CE QUE LE LIVE A APPRIS - l ISO du 3953233 avait TOUT (binaires,
# lanceurs, pmbootstrap, noyau PPP, image de reference), et ni la 4G ni le
# telephone ne marchaient :
#   - open5gs-mmed (et hssd, pcrfd, smfd) mourait a l init : PAS de
#     etc/freeDiameter/*.conf ni de etc/open5gs/tls/. Open5GS les pose par un
#     script meson qui ne fait RIEN sous DESTDIR, et osmo-deb pack installe
#     sous DESTDIR : le .deb n a que bin/ et lib/. Les yaml, osmo-lte-install
#     les reposait ; freeDiameter, personne. Sans MME, pas de S1AP : srsUE
#     recevait ConnectionReject, jamais d attach. Les .conf sont maintenant
#     dans le depot (configs/open5gs/freeDiameter), poses avec des certificats
#     par osmo-lte-install --configs ; le controle final les exige ET fait
#     demarrer open5gs-mmed 3 s dans le chroot.
#   - l icone du telephone lancait « pmbootstrap qemu » en root (la session
#     de l ISO est root) : « Do not run pmbootstrap as root! ». Et aucun
#     compte n avait de ~/.config/pmbootstrap_v3.cfg : le gabarit n etait
#     que dans /etc/skel, rempli APRES la creation des comptes (84-comptes).
#     Les lanceurs passent --as-root en root, le cfg est ecrit dans /root et
#     /home/osmocom ici, et osmo-pmos-qemu enchaine osmo-pmos-build quand
#     l image n est pas encore la.
#   - pmbootstrap exige kpartx, absent de l ISO (PKGS de 80-chroot.sh).
#   - le premier clic clonait pmaports en entier depuis GitLab, et le clone
#     echouait presque a chaque fois : le commit du noyau est pose ICI dans
#     /opt/user_interface/pmos/pmaports (fetch --depth 1, 110 Mo), et
#     osmo-pmos-build le copie au lieu de cloner.
#   - le son : la VM nue (sans modem) sortait sur la carte PONT, inaudible ;
#     osmo-pmos-qemu appelle osmo-pmos-setup --voix des que le SSH repond,
#     et le disque est bootindex=0 (patch pmbootstrap) - plus de detour PXE.
#
# Le hub (interstp) n a pas de 4G. --arm non plus (srsRAN et Open5GS ne sont
# pas compiles pour arm64, pmbootstrap y ferait une VM x86 emulee).
if [ "$ISO_ROLE" = "interstp" ] || [ "${ISO_ARCH:-amd64}" = "arm64" ]; then return 0; fi

echo -e "${GREEN}[8d/9] La 4G (srsRAN ZeroMQ + Open5GS) et l UI smartphone (postmarketOS)...${NC}"

# ── /proc /sys /dev : REMONTES POUR LA DUREE DE CE MODULE ──────────────────
```

```
# [2026-09-14] 86-finitions.sh se termine par « umount $ROOTFS/{dev/pts,proc,
# sys,dev} » : c etait la fin du chroot, du temps ou ce module n existait pas.
# Or TOUT ce qui suit tourne DANS le chroot - dpkg du noyau pmOS, apt (--deps),
# la compilation srsRAN / Open5GS, systemctl enable, et les verifications
# finales. Le cmake de srsRAN le disait a chaque configuration :
#     Problem opening /proc/cpuinfo
#     Problem opening /proc/meminfo
# il ne voyait ni CPU ni memoire, et sans /dev un maintainer-script ou un
# binaire qui lit /dev/urandom (open5gs-mmed du test de 3 s) tombe sans raison
# lisible. On remonte donc ici ce qu il faut, et on le DEMONTE a la fin du
# module : 90-iso.sh fait le squashfs, rien ne doit rester monte sous $ROOTFS.
# Idempotent dans les deux sens - si un jour 86 ne demonte plus, mountpoint le
# voit et on ne monte pas deux fois ; on ne demonte que ce qu on a monte.
_iso88_mnt=()
_iso88_up() {   # _iso88_up <source hote> <point de montage, relatif au rootfs>
    local _t="$ROOTFS$2"
    [ -d "$_t" ] || install -d "$_t" 2>/dev/null || return 0
    mountpoint -q "$_t" 2>/dev/null && return 0
    mount --bind "$1" "$_t" 2>/dev/null && _iso88_mnt=("$_t" "${_iso88_mnt[@]}")
    return 0
}
_iso88_down() {  # ordre inverse du montage (/dev/pts avant /dev)
    local _m
    for _m in "${_iso88_mnt[@]}"; do umount -l "$_m" 2>/dev/null || true; done
    _iso88_mnt=()
}
_iso88_up /proc    /proc
_iso88_up /sys     /sys
_iso88_up /dev     /dev
_iso88_up /dev/pts /dev/pts

# Le DNS du chroot, pour les clones git de --build. 81-cloture-systeme.sh a
# remplace /etc/resolv.conf par le lien vers /run/systemd/resolve/stub-resolv.conf
# - ce qu il faut dans l ISO, mais ici il ne designe rien. On peuple la cible :
# /run est un tmpfs au demarrage, ce fichier ne survit pas au premier boot.
install -d "$ROOTFS/run/systemd/resolve"
[ -s "$ROOTFS/run/systemd/resolve/stub-resolv.conf" ] || {
    grep '^nameserver' /etc/resolv.conf 2>/dev/null | grep -v '^nameserver 127\.' \
      || printf 'nameserver 1.1.1.1\nnameserver 8.8.8.8\n'
} > "$ROOTFS/run/systemd/resolve/stub-resolv.conf"

_rt="$ROOTFS/opt/GSM/osmo-operator"
install -d "$_rt/tools" "$_rt/configs/srsran" "$_rt/configs/open5gs" "$_rt/configs/pmos" "$_rt/patches"
 ↪  "$_rt/packaging"
# Le depot local fait foi (meme regle que 85-installeur-bureau.sh).
for _f in osmo-lte.sh osmo-epc.sh osmo-lte-install.sh osmo-pmos.sh osmo-pmos-qemu.sh osmo-pmos-setup.sh \
          osmo-pmos-build.sh osmo-pmos-install.sh osmo-phonesim-banc.py at-cmd.py; do
    [ -f "$DIR/tools/$_f" ] && install -m755 "$DIR/tools/$_f" "$_rt/tools/"
done
install -m644 "$DIR"/configs/srsran/* "$_rt/configs/srsran/"
install -m644 "$DIR"/configs/open5gs/*.yaml "$DIR"/configs/open5gs/*.json "$_rt/configs/open5gs/" 2>/dev/null ||
 ↪  true
[ -d "$DIR/configs/open5gs/dump" ] && cp -a "$DIR/configs/open5gs/dump" "$_rt/configs/open5gs/"
# [2026-09-09] freeDiameter : les .conf du depot (le .deb n en a pas, voir mme.conf).
[ -d "$DIR/configs/open5gs/freeDiameter" ] && cp -a "$DIR/configs/open5gs/freeDiameter" "$_rt/configs/open5gs/"
install -m644 "$DIR"/configs/pmos/* "$_rt/configs/pmos/"
for _f in pmbootstrap-osmo-bench-qemu.patch pmaports-linux-postmarketos-stable-ppp.patch; do
    [ -f "$DIR/patches/$_f" ] && install -m644 "$DIR/patches/$_f" "$_rt/patches/"
done
for _f in build-pmos-kernel-deb.sh snapshot-lte-debs.sh osmo-deb.sh; do
    [ -f "$DIR/packaging/$_f" ] && install -m755 "$DIR/packaging/$_f" "$_rt/packaging/"
done
unset _f

# Le noyau pmOS PPP absent du cache : GitHub (Git LFS, bbaranoff/pmos_ppp_kernel)
# le fournit - dans le cache de l hote, pour la prochaine fois, et dans le rootfs.
```

```
if [ ! -e "$ROOTFS/opt/user_interface/kernel/pmos/APKINDEX.tar.gz" ]; then
    _kurl="${OSMO_PMOS_KERNEL_URL:-https://media.githubusercontent.com/media/bbaranoff/pmos_ppp_kernel/main/osmo-
     ↪  build-pmos-kernel_7.2.3.0+ppp~noble_amd64.deb}"
    _kdeb="${OSMO_DEB_CACHE:-/var/cache/osmo-debs}/$(basename "$_kurl")"
    if [ ! -s "$_kdeb" ]; then
        echo -e "  ${CYAN}·${NC} noyau pmOS PPP : telechargement depuis GitHub ($_kurl)"
        install -d "$(dirname "$_kdeb")"
        curl -fsSL --retry 3 -o "$_kdeb.part" "$_kurl" && dpkg-deb --info "$_kdeb.part" >/dev/null 2>&1 \
            && mv -f "$_kdeb.part" "$_kdeb" || { rm -f "$_kdeb.part"; echo -e "  ${YELLOW}!${NC} noyau pmOS :
             ↪  telechargement impossible"; }
    fi
    if [ -s "$_kdeb" ]; then
        install -d "$ROOTFS/var/cache/osmo-debs"; cp -f "$_kdeb" "$ROOTFS/var/cache/osmo-debs/"
        chroot "$ROOTFS" dpkg -i --force-overwrite "/var/cache/osmo-debs/$(basename "$_kdeb")" >/dev/null 2>&1 \
            && echo -e "  ${GREEN}✓${NC} noyau pmOS PPP pose depuis $(basename "$_kdeb")"
        [ "${ISO_EMBED_DEBS:-0}" = "1" ] || rm -f "$ROOTFS/var/cache/osmo-debs/$(basename "$_kdeb")"
    fi
    unset _kurl _kdeb
fi

# Ce que les .deb ont pose (ou pas) : on le dit, on ne s arrete pas.
for _c in "srsenb:/usr/local/bin/srsenb" "Open5GS:/opt/LTE/open5gs/install/bin/open5gs-mmed" \
          "noyau pmOS PPP:/opt/user_interface/kernel/pmos/APKINDEX.tar.gz" \
          "pmbootstrap:/opt/user_interface/pmos/pmbootstrap/pmbootstrap.py"; do
    if [ -e "$ROOTFS/${_c#*:}" ]; then echo -e "  ${GREEN}✓${NC} ${_c%%:*} (.deb du build)"
    else echo -e "  ${YELLOW}!${NC} ${_c%%:*} absent du rootfs - pas de .deb dans /var/cache/osmo-debs
     ↪  (packaging/snapshot-lte-debs.sh sur la machine de reference, ou le Dockerfile)"; fi
done
unset _c

# Le depot MongoDB de 80-chroot.sh est deja la : --deps ne fait ici que les
# paquets qui manqueraient encore (et enable mongod). Puis les .deb du cache.
chroot "$ROOTFS" env DEBIAN_FRONTEND=noninteractive OSMO_REPO=/opt/GSM/osmo-operator \
    bash /opt/GSM/osmo-operator/tools/osmo-lte-install.sh --deps --debs 2>&1 | sed 's/^/  /'

# ── PAS DE .deb : ON COMPILE DANS LE CHROOT ──────────────────────────────
# [2026-09-09] L ISO n avait qu UN chemin vers ses binaires 4G : les .deb du
# build docker (ou du snapshot natif). Le jour ou le Dockerfile est tombe -
# cmake de srsRAN, « Could NOT find MbedTLS », deps apt absentes de l image -
# le cache est reste vide et l ISO est sortie SANS srsRAN ni Open5GS, avec
# pour seule trace le « ! absent du rootfs » ci-dessus. Une ISO muette sur sa
# 4G, c est un banc qu on decouvre casse a l usage.
#
# Le chroot a tout ce qu il faut pour compiler : les -dev sont dans PKGS de
# 80-chroot.sh (libmbedtls-dev, libzmq3-dev, libconfig++-dev, boost, meson,
# flex, bison, libmongoc-dev...), --deps vient de repasser, et /proc /sys /dev
# et le resolv.conf de l hote y sont montes (git clone possible). On rejoue
# donc le MEME script que le Dockerfile et que le natif : --build.
#
# OSMO_DEB=1 : la compilation sort en .deb, qu on RAMENE dans le cache de
# l hote - la prochaine ISO (et le prochain docker) ne recompileront pas.
# OSMO_ISO_LTE_BUILD=0 pour refuser ce rattrapage (build court, ISO sans 4G).
#
# [2026-09-14] ET SI LE .deb DU CACHE EST TAILLE POUR UNE AUTRE MACHINE ?
# La condition ci-dessous etait « les binaires sont-ils la ? ». Un
# osmo-build-srsran fabrique sur la machine de reference (ou par un runner de
# CI) l est avec -march=native : son srsenb portait 19 353 instructions %ymm
# (AVX/AVX2) et 417 FMA. Pose tel quel dans le rootfs il passe tous les
# controles - il EXISTE, ses bibliotheques sont resolues, --help repond - et
# l ISO sort. C est a l usage, sur la machine de l operateur, que ca tombe :
#     /usr/local/bin/osmo-lte: line 204: ... Illegal instruction (core dumped) ... srsenb
#     /usr/local/bin/osmo-lte: line 244: ... Illegal instruction (core dumped) ... srsue
# Une ISO passe d une machine a l autre : ses binaires doivent etre portables.
# On regarde donc ce que celui du cache CONTIENT, et on le refuse s il porte
# de l AVX - la compilation ci-dessous le remplace par un binaire de base
# commune (voir SRS_ARCH dans tools/osmo-lte-install.sh).
```

```
_iso_srs_avx() {   # 0 = ce binaire ne tournera pas partout
    [ -x "$ROOTFS/usr/local/bin/srsenb" ] || return 1
    command -v objdump >/dev/null 2>&1 || return 1
    objdump -d "$ROOTFS/usr/local/bin/srsenb" 2>/dev/null | grep -qE '%[yz]mm[0-9]'
}
if _iso_srs_avx; then
    echo -e "  ${YELLOW}!${NC} srsenb du cache .deb taille pour le CPU du build (AVX) : il ferait « Illegal
     ↪  instruction » ailleurs"
    rm -f "$ROOTFS"/usr/local/bin/srsenb "$ROOTFS"/usr/local/bin/srsue "$ROOTFS"/usr/local/bin/srsepc
    # Le .deb fautif est ecarte du cache de l hote (renomme, pas efface : le
    # glob de recuperation plus bas ne prend que *.deb, et l operateur garde
    # de quoi comprendre). Sans ca, la prochaine ISO le reposerait.
    for _sd in "${OSMO_DEB_CACHE:-/var/cache/osmo-debs}"/osmo-build-srsran_*.deb; do
        [ -f "$_sd" ] || continue
        mv -f "$_sd" "$_sd.avx-natif" && echo -e "    ${CYAN}·${NC} $(basename "$_sd") ecarte -> $(basename
         ↪  "$_sd").avx-natif"
    done
    unset _sd
fi
unset -f _iso_srs_avx
if [ ! -x "$ROOTFS/usr/local/bin/srsenb" ] || [ ! -x "$ROOTFS/opt/LTE/open5gs/install/bin/open5gs-mmed" ]; then
    if [ "${OSMO_ISO_LTE_BUILD:-1}" = "1" ]; then
        echo -e "  ${YELLOW}!${NC} 4G absente du cache .deb : compilation dans le chroot (long)"
        install -d "$ROOTFS/var/cache/osmo-debs"
        [ -f "$DIR/packaging/osmo-deb.sh" ] && install -m755 "$DIR/packaging/osmo-deb.sh"
         ↪  "$ROOTFS/usr/local/sbin/osmo-deb"
        # OSMO_SRSRAN_ARCH : la base ISA des binaires de l ISO. Le defaut du
        # script (x86-64-v2) est ce qu il faut ici - une ISO doit tourner
        # ailleurs que sur la machine qui l a construite ; on le passe quand
        # meme explicitement pour qu un build particulier puisse le changer.
        chroot "$ROOTFS" env DEBIAN_FRONTEND=noninteractive OSMO_DEB=1 OSMO_REPO=/opt/GSM/osmo-operator \
            OSMO_SRSRAN_ARCH="${OSMO_SRSRAN_ARCH:-x86-64-v2}" \
            bash /opt/GSM/osmo-operator/tools/osmo-lte-install.sh --build 2>&1 | sed 's/^/  /' \
            || echo -e "  ${YELLOW}!${NC} compilation 4G echouee dans le chroot - ISO sans srsRAN/Open5GS"
        # Les .deb fraichement produits repartent dans le cache de l hote.
        _dc="${OSMO_DEB_CACHE:-/var/cache/osmo-debs}"
        install -d "$_dc"
        for _d in "$ROOTFS"/var/cache/osmo-debs/osmo-build-*.deb; do
            [ -f "$_d" ] || continue
            [ -f "$_dc/$(basename "$_d")" ] || { cp -f "$_d" "$_dc/" && echo -e "  ${GREEN}✓${NC} $(basename
             ↪  "$_d") remis dans $_dc"; }
        done
        [ "${ISO_EMBED_DEBS:-0}" = "1" ] || rm -f "$ROOTFS"/var/cache/osmo-debs/osmo-build-*.deb
        unset _d _dc
    else
        echo -e "  ${YELLOW}!${NC} 4G absente et OSMO_ISO_LTE_BUILD=0 : ISO sans srsRAN/Open5GS"
    fi
fi

# Les configs Open5GS ne se posent que si le prefixe existe : APRES le build.
chroot "$ROOTFS" env OSMO_REPO=/opt/GSM/osmo-operator \
    bash /opt/GSM/osmo-operator/tools/osmo-lte-install.sh --configs --launchers 2>&1 | sed 's/^/  /'
chroot "$ROOTFS" env OSMO_REPO=/opt/GSM/osmo-operator \
    bash /opt/GSM/osmo-operator/tools/osmo-pmos-install.sh 2>&1 | sed 's/^/  /'

# [2026-09-08] L IMAGE DE REFERENCE, PAR DEFAUT. Le build du soir avait ete
# lance sans OSMO_ISO_PMOS_IMAGE : l ISO avait srsRAN, Open5GS, le noyau PPP
# et pmbootstrap, mais PAS le telephone - le premier osmo-pmos retombait sur
# un « pmbootstrap install » complet (long, reseau). La machine de reference
# porte pourtant l image prete a l endroit meme ou osmo-pmos-build la cherche
# (/opt/user_interface/pmos/image). Si la variable n est pas donnee, on prend
# donc celle de l hote : l ISO devient ce qu est le disque. Le .zst pret est
# prefere ; a defaut l image brute de pmbootstrap, compressee au passage.
# OSMO_ISO_PMOS_IMAGE=none pour ne rien embarquer.
if [ -z "${OSMO_ISO_PMOS_IMAGE:-}" ]; then
    for _cand in /opt/user_interface/pmos/image/qemu-amd64.img.zst \
```

```
                "${OSMO_DEB_CACHE:-/var/cache/osmo-debs}/qemu-amd64.img.zst" \
                "${OSMO_PMB_WORK:-/root/test}"/chroot_native/home/pmos/rootfs/qemu-amd64.img \
                /home/*/test/chroot_native/home/pmos/rootfs/qemu-amd64.img; do
        [ -f "$_cand" ] || continue
        OSMO_ISO_PMOS_IMAGE="$_cand"
        echo -e "  ${CYAN}·${NC} image de la VM de reference trouvee sur l hote : $_cand"
        break
    done
    unset _cand
fi

# [2026-09-13] ET SUR UN RUNNER GITHUB ? Aucune machine de reference : pas de
# /opt/user_interface/pmos/image, pas de dossier de travail pmbootstrap - la
# boucle ci-dessus ne trouve rien et l ISO de la CI sort sans telephone pret.
# Meme remede que pour le noyau (plus haut), mais PAS par Git LFS : 1,7 Go
# creveraient le quota LFS (1 Go de stockage et 1 Go de trafic par mois). Un
# ASSET DE RELEASE n a pas de quota de trafic et plafonne a 2 Gio par fichier -
# l image y tient. Publier une fois, depuis la machine de reference :
#
#   gh release create pmos-image --repo bbaranoff/osmo-operator \
#       --title "postmarketOS - VM de reference (noyau PPP)" --notes "..." \
#       /opt/user_interface/pmos/image/qemu-amd64.img.zst
#
# puis OSMO_PMOS_IMAGE_URL=<url de l asset> au build (le workflow le passe).
# Vide = on ne telecharge rien : c est le defaut, l image pese trois fois son
# poids dans un build --all (operator, lite, desktop).
if [ -z "${OSMO_ISO_PMOS_IMAGE:-}" ] && [ -n "${OSMO_PMOS_IMAGE_URL:-}" ]; then
    _icache="${OSMO_DEB_CACHE:-/var/cache/osmo-debs}/qemu-amd64.img.zst"
    echo -e "  ${CYAN}·${NC} image de la VM : telechargement depuis $OSMO_PMOS_IMAGE_URL"
    install -d "$(dirname "$_icache")"
    # zstd -t : un 404 ou une coupure donnent un fichier bien forme pour curl et
    # illisible pour zstd - c est ici qu on veut le savoir, pas dans l ISO.
    if curl -fsSL --retry 3 --retry-delay 5 -o "$_icache.part" "$OSMO_PMOS_IMAGE_URL" \
       && zstd -t "$_icache.part" >/dev/null 2>&1; then
        mv -f "$_icache.part" "$_icache"
        OSMO_ISO_PMOS_IMAGE="$_icache"
        echo -e "  ${GREEN}✓${NC} image recuperee ($(du -h "$_icache" | cut -f1)) - gardee dans $(dirname
         ↪  "$_icache") pour les ISO suivantes"
    else
        rm -f "$_icache.part"
        echo -e "  ${YELLOW}!${NC} image de la VM : telechargement impossible - ISO sans telephone pret
         ↪  (osmo-pmos-build le fera sur la machine)"
    fi
    unset _icache
fi
[ "${OSMO_ISO_PMOS_IMAGE:-}" = "none" ] && OSMO_ISO_PMOS_IMAGE=""
# L image de la VM : celle donnee, ou celle de l hote (voir ci-dessus).
if [ -n "${OSMO_ISO_PMOS_IMAGE:-}" ]; then
    if [ -f "$OSMO_ISO_PMOS_IMAGE" ]; then
        install -d "$ROOTFS/opt/user_interface/pmos/image"
        case "$OSMO_ISO_PMOS_IMAGE" in
            *.zst) cp -f "$OSMO_ISO_PMOS_IMAGE" "$ROOTFS/opt/user_interface/pmos/image/qemu-amd64.img.zst" ;;
            *) zstd -T0 -3 --sparse -f -o "$ROOTFS/opt/user_interface/pmos/image/qemu-amd64.img.zst"
             ↪  "$OSMO_ISO_PMOS_IMAGE" ;;
        esac && echo -e "  ${GREEN}✓${NC} image de la VM embarquee ($(du -h
         ↪  "$ROOTFS/opt/user_interface/pmos/image/qemu-amd64.img.zst" | cut -f1))"
    else
        echo -e "  ${YELLOW}!${NC} OSMO_ISO_PMOS_IMAGE=$OSMO_ISO_PMOS_IMAGE introuvable - image non embarquee"
    fi
else
    echo -e "  ${CYAN}·${NC} image de la VM non embarquee (OSMO_ISO_PMOS_IMAGE=... pour l ajouter) :
     ↪  osmo-pmos-build la fera"
fi

# ── PMAPORTS DANS L ISO, AU COMMIT DU NOYAU ──────────────────────────────
# [2026-09-09] Le premier clic sur le telephone clonait pmaports EN ENTIER
```

```
# depuis GitLab (des centaines de Mo) ; depuis le live, deux essais sur trois
# ont echoue, et git efface tout a l echec : « ca me clone a chaque fois et
# ca boot pas ». On pose donc ici le SEUL commit qu il faut - celui du noyau
# PPP (kernel/pmos/pmaports.commit), fetch --depth 1, quelques dizaines de
# Mo, avec la ref origin/main que pmbootstrap exige pour lire channels.cfg -
# dans /opt/user_interface/pmos/pmaports ; osmo-pmos-build le COPIE dans
# le dossier de travail du compte au lieu de cloner. Garde dans le cache de
# l hote (OSMO_DEB_CACHE) pour les ISO suivantes. OSMO_ISO_PMAPORTS=0 pour
# s en passer.
_pc="$(cat "$ROOTFS/opt/user_interface/kernel/pmos/pmaports.commit" 2>/dev/null || true)"
if [ "${OSMO_ISO_PMAPORTS:-1}" = "1" ] && [ -n "$_pc" ] && [ "$_pc" != inconnu ]; then
    _pcache="${OSMO_DEB_CACHE:-/var/cache/osmo-debs}/pmaports-$_pc"
    if ! git -C "$_pcache" rev-parse --verify HEAD >/dev/null 2>&1; then
        echo -e "  ${CYAN}·${NC} pmaports : fetch du commit $_pc (GitLab, --depth 1)..."
        rm -rf "$_pcache"
        ( git init -q "$_pcache" && cd "$_pcache" \
          && git remote add origin https://gitlab.postmarketos.org/postmarketOS/pmaports.git \
          && git fetch -q --depth 1 origin "$_pc" && git checkout -q -B main FETCH_HEAD \
          && git update-ref refs/remotes/origin/main FETCH_HEAD ) \
          || { rm -rf "$_pcache"; echo -e "  ${YELLOW}!${NC} pmaports : fetch impossible (reseau ?) - le premier
           ↪  clic ira le chercher"; }
    fi
    if git -C "$_pcache" rev-parse --verify HEAD >/dev/null 2>&1; then
        rm -rf "$ROOTFS/opt/user_interface/pmos/pmaports"
        cp -a "$_pcache" "$ROOTFS/opt/user_interface/pmos/pmaports"
        echo -e "  ${GREEN}✓${NC} pmaports au commit ${_pc:0:10} dans /opt/user_interface/pmos/pmaports ($(du -sh
         ↪  "$_pcache" | cut -f1))"
    fi
    unset _pcache
fi
unset _pc

# Le compte de session lancera pmbootstrap : /opt/user_interface doit lui
# etre lisible, et le gabarit de config est pose dans /etc/skel pour qu un
# compte cree par Calamares le trouve (shellprocess-osmo.conf y met les
# chemins de SON home).
chroot "$ROOTFS" chmod -R a+rX /opt/user_interface 2>/dev/null || true
install -d "$ROOTFS/etc/skel/.config"
install -m644 "$DIR/configs/pmos/pmbootstrap_v3.cfg" "$ROOTFS/etc/skel/.config/pmbootstrap_v3.cfg.osmo"
# [2026-09-09] ET DANS LES COMPTES QUI EXISTENT DEJA. /etc/skel ne sert qu aux
# comptes crees APRES : root et osmocom le sont par 84-comptes.sh, avant ce
# module - sur l ISO aucun des deux n avait de ~/.config/pmbootstrap_v3.cfg,
# et la session s ouvre sur root. Meme substitution que shellprocess-osmo.conf
# (dossier de travail ~/.local/var/pmbootstrap).
for _h in /root /home/osmocom; do
    [ -d "$ROOTFS$_h" ] || continue
    [ -s "$ROOTFS$_h/.config/pmbootstrap_v3.cfg" ] && continue
    install -d "$ROOTFS$_h/.config"
    sed "s#@WORK@#$_h/.local/var/pmbootstrap#g; s#@APORTS@#$_h/.local/var/pmbootstrap/cache_git/pmaports#g" \
        "$DIR/configs/pmos/pmbootstrap_v3.cfg" > "$ROOTFS$_h/.config/pmbootstrap_v3.cfg"
done
chroot "$ROOTFS" chown -R osmocom:osmocom /home/osmocom/.config 2>/dev/null || true
unset _h
echo -e "  ${GREEN}✓${NC} pmbootstrap_v3.cfg : /root, /home/osmocom, /etc/skel"

# Lite : les objets de compilation de /opt/LTE ne servent pas a tourner.
if [ "$ISO_LITE" = "1" ]; then
    _before=$(du -sh "$ROOTFS/opt/LTE" 2>/dev/null | cut -f1)
    find "$ROOTFS/opt/LTE/srsRAN_4G/build" "$ROOTFS/opt/LTE/open5gs/build" -name '*.o' -delete 2>/dev/null || true
    rm -rf "$ROOTFS/opt/LTE/libzmq" "$ROOTFS/opt/LTE/czmq" 2>/dev/null || true
    echo -e "  ${GREEN}✓${NC} lite : /opt/LTE ${_before:-?} -> $(du -sh "$ROOTFS/opt/LTE" 2>/dev/null | cut -f1)"
fi
unset _rt _before

# ── LE CONTROLE FINAL : L ISO DOIT ETRE PRETE A L EMPLOI, 2G + 4G + TELEPHONE ─
# [2026-09-09] Jusqu ici ce module ne faisait que DIRE ce qui manquait (les
```

```
# « ! absent du rootfs » plus haut) et l ISO sortait quand meme - une ISO ou
# l icone « 4G du banc » ne lancait rien, ou Open5GS n avait pas de mme.yaml
# (donc pas de SGs, donc pas de CSFB), ou mongod manquait (le HSS meurt a
# l init, l attach echoue). Or ce qu on attend d elle est simple : l
# operateur clique « Lancer le banc GSM », puis « 4G du banc », puis « le
# telephone pmOS », et ca marche. On verifie donc ICI, dans le rootfs, chaque
# maillon de ces trois gestes, et on ARRETE le build s il en manque un :
# mieux vaut une heure de build perdue qu une cle qui ment.
# OSMO_ISO_LTE_REQUIRED=0 pour ne faire que l inventaire (ISO sans 4G assumee).
echo -e "  ${CYAN}·${NC} controle final : 2G + 4G + telephone dans le rootfs"
_ko=0
_chk() {  # _chk <libelle> <test chroot...>
    local l="$1"; shift
    if chroot "$ROOTFS" "$@" >/dev/null 2>&1; then echo -e "      ${GREEN}✓${NC} $l"
    else echo -e "      ${RED}✗${NC} $l"; _ko=$((_ko + 1)); fi
}
# La 2G : le banc, son unite, et le SGs d osmo-msc (l attach combine, le CSFB)
_chk "2G  launch.sh (icone « Lancer le banc GSM »)"       test -x /opt/GSM/osmo-operator/launch.sh
_chk "2G  osmo-banc.service"                               test -s /etc/systemd/system/osmo-banc.service
_chk "2G  osmo-msc.cfg : section sgs (CSFB)"               grep -q '^sgs' /etc/osmocom/osmo-msc.cfg
# La 4G : binaires, configs (dont le SGs cote MME), lanceurs, unite, MongoDB
_chk "4G  srsenb / srsue"                                  bash -c 'test -x /usr/local/bin/srsenb && test -x
 ↪  /usr/local/bin/srsue'
# ── ET LEURS BIBLIOTHEQUES SONT-ELLES TOUTES LA ? ──────────────────────
# [2026-09-10] srsRAN est desormais compile avec ENABLE_GUI=ON : srsenb et
# srsue se LIENT a libsrsgui (srsGUI, /usr/local/lib), posee avant eux par
# osmo-build-srsgui. Une image qui aurait les binaires SANS la bibliotheque
# donnerait « error while loading shared libraries: libsrsgui.so » au premier
# clic sur l icone 4G - et rien avant. Le `ldd` couvre srsgui comme le reste.
_chk "4G  srsenb / srsue : bibliotheques toutes resolues" \
    bash -c 'for b in srsenb srsue; do ldd /usr/local/bin/$b 2>/dev/null | grep -q "not found" && exit 1; done;
 ↪  exit 0'
# ── ET TOURNERONT-ILS SUR LA MACHINE DE L OPERATEUR ? ──────────────────
# [2026-09-14] Le controle le plus discret et le plus couteux : des binaires
# presents, lies, qui repondent a --help, et qui meurent en « Illegal
# instruction » des le premier vrai lancement parce qu ils ont ete compiles
# avec -march=native sur une autre machine. C est exactement ce qu a vecu
# l operateur. Une instruction AVX (%ymm) ou AVX512 (%zmm) dans srsenb ou
# srsue condamne l ISO hors du CPU qui l a construite : ici, c est une faute.
# Le test se fait depuis l hote (objdump du rootfs), l ISO n embarque pas
# binutils. Sans objdump sur l hote on ne juge pas : on le dit.
if command -v objdump >/dev/null 2>&1; then
    if objdump -d "$ROOTFS/usr/local/bin/srsenb" 2>/dev/null | grep -qE '%[yz]mm[0-9]' \
       || objdump -d "$ROOTFS/usr/local/bin/srsue" 2>/dev/null | grep -qE '%[yz]mm[0-9]'; then
        echo -e "      ${RED}✗${NC} 4G  srsenb / srsue portables (pas d AVX du CPU du build)"
        echo -e "          recompiler : OSMO_SRSRAN_ARCH=x86-64-v2, cf. tools/osmo-lte-install.sh"
        _ko=$((_ko + 1))
    else
        echo -e "      ${GREEN}✓${NC} 4G  srsenb / srsue portables (pas d AVX du CPU du build)"
    fi
else
    echo -e "      ${CYAN}·${NC} 4G  objdump absent de l hote : portabilite ISA de srsenb non verifiee"
fi
# Les traces temps reel sont un bonus, pas une condition : on les ANNONCE.
if chroot "$ROOTFS" bash -c 'ldd /usr/local/bin/srsenb 2>/dev/null | grep -q libsrsgui' 2>/dev/null; then
    echo -e "      ${GREEN}✓${NC} 4G  srsGUI : traces temps reel armees (srsenb lie a libsrsgui)"
else
    echo -e "      ${CYAN}·${NC} 4G  srsGUI absent : srsenb sans traces temps reel (« osmo-lte-install --build »)"
fi
_chk "4G  open5gs-mmed / open5gs-hssd"                     bash -c 'test -x
 ↪  /opt/LTE/open5gs/install/bin/open5gs-mmed && test -x /opt/LTE/open5gs/install/bin/open5gs-hssd'
_chk "4G  mme.yaml du depot avec sgsap (CSFB)"             grep -q '^  sgsap:'
 ↪  /opt/LTE/open5gs/install/etc/open5gs/mme.yaml
_chk "4G  freeDiameter mme/hss/pcrf/smf.conf (S6a, Gx)"   bash -c 'for n in mme hss pcrf smf; do test -s
 ↪  /opt/LTE/open5gs/install/etc/freeDiameter/$n.conf || exit 1; done'
```

```
_chk "4G  TLS freeDiameter (ca + mme/hss/pcrf/smf)"       bash -c 'for n in ca mme hss pcrf smf; do test -s
 ↪  /opt/LTE/open5gs/install/etc/open5gs/tls/$n.crt || exit 1; done'
# Et le MME DEMARRE-T-IL ? 3 s au premier plan dans le chroot : une config
# absente ou une lib manquante se voient LA, pas au premier clic (c est
# exactement la panne du live du 2026-09-09). Un port deja pris sur l hote
# (sa propre 4G tourne) n est pas une faute de l ISO : on le dit, sans juger.
_o="$(chroot "$ROOTFS" timeout 3 /opt/LTE/open5gs/install/bin/open5gs-mmed -c
 ↪  /opt/LTE/open5gs/install/etc/open5gs/mme.yaml 2>&1 || true)"
rm -f "$ROOTFS/opt/LTE/open5gs/install/var/log/open5gs/mme.log"
if echo "$_o" | grep -aq 'Address already in use'; then
    echo -e "      ${CYAN}·${NC} 4G  open5gs-mmed : port deja pris sur l hote (sa 4G tourne ?) - demarrage non
     ↪  teste"
elif echo "$_o" | grep -aqE 'FATAL|error while loading shared'; then
    echo -e "      ${RED}✗${NC} 4G  open5gs-mmed demarre (3 s au premier plan)"
    echo "$_o" | grep -aE 'FATAL|error while loading' | grep -v backtrace | head -2 | sed 's/^/          /'
    _ko=$((_ko + 1))
else
    echo -e "      ${GREEN}✓${NC} 4G  open5gs-mmed demarre (3 s au premier plan)"
fi
unset _o
_chk "4G  configs srsRAN (/root/.config/srsran)"           bash -c 'test -s /root/.config/srsran/enb.conf && test
 ↪  -s /root/.config/srsran/ue.conf && test -s /root/.config/srsran/user_db.csv'
_chk "4G  osmo-lte / osmo-epc"                             bash -c 'test -x /usr/local/bin/osmo-lte && test -x
 ↪  /usr/local/bin/osmo-epc'
# [2026-09-16] La data du telephone : docker met FORWARD a DROP, et sans la
# regle DOCKER-USER posee par osmo-epc (setup_forward), osmo-pmos-qemu
# (pmos_forward) et le modem du banc (session PPP), le PPP monte mais rien ne
# sort de l hote. L ISO ne part pas sans ces trois-la.
_chk "4G  FORWARD ogstun : osmo-epc, osmo-pmos-qemu, phonesim" bash -c 'grep -q "^setup_forward()"
 ↪  /opt/GSM/osmo-operator/tools/osmo-epc.sh && grep -q "^pmos_forward()"
 ↪  /opt/GSM/osmo-operator/tools/osmo-pmos-qemu.sh && grep -q "DOCKER-USER"
 ↪  /opt/GSM/osmo-operator/tools/osmo-phonesim-banc.py'
_chk "4G  osmo-lte.service + osmo-lte.desktop"             bash -c 'test -s /etc/systemd/system/osmo-lte.service
 ↪  && test -s /usr/share/applications/osmo-lte.desktop'
_chk "4G  mongod + mongosh + mongorestore (abonnes HSS)"   bash -c 'command -v mongod && command -v mongosh &&
 ↪  command -v mongorestore'
_chk "4G  abonnes du depot (dump mongo + subscribers.json)" bash -c 'test -s
 ↪  /opt/GSM/osmo-operator/configs/open5gs/dump/open5gs/subscribers.bson && test -s
 ↪  /opt/GSM/osmo-operator/configs/open5gs/subscribers.json'
# Le telephone : pmbootstrap patche, le noyau PPP, le lanceur et son icone
_chk "pmOS pmbootstrap (/opt/user_interface/pmos)"         test -s
 ↪  /opt/user_interface/pmos/pmbootstrap/pmbootstrap.py
_chk "pmOS noyau PPP (APKINDEX)"                          test -s /opt/user_interface/kernel/pmos/APKINDEX.tar.gz
_chk "pmOS osmo-pmos + osmo-pmos.desktop"                  bash -c 'test -x /usr/local/bin/osmo-pmos && test -s
 ↪  /usr/share/applications/osmo-pmos.desktop'
_chk "pmOS pmbootstrap_v3.cfg de root (la session de l ISO)" test -s /root/.config/pmbootstrap_v3.cfg
if [ -d "$ROOTFS/home/osmocom" ]; then
    _chk "pmOS pmbootstrap_v3.cfg d osmocom"                bash -c 'test -s
     ↪  /home/osmocom/.config/pmbootstrap_v3.cfg && ! grep -q @WORK@ /home/osmocom/.config/pmbootstrap_v3.cfg'
fi
_chk "pmOS lanceurs en root (--as-root dans osmo-pmos-qemu)" grep -q 'as-root'
 ↪  /opt/user_interface/pmos/bin/osmo-pmos-qemu.sh
_chk "pmOS kpartx + losetup + git (pmbootstrap les exige)"   bash -c 'command -v kpartx && command -v losetup &&
 ↪  command -v git'
if [ "${OSMO_ISO_PMAPORTS:-1}" = "1" ]; then
    _chk "pmOS pmaports au commit du noyau (pas de clone au premier clic)" bash -c 'test -f
     ↪  /opt/user_interface/pmos/pmaports/device/main/linux-postmarketos-stable/config-stable.x86_64'
fi
if [ "$_ko" -gt 0 ]; then
    if [ "${OSMO_ISO_LTE_REQUIRED:-1}" = "1" ]; then
        echo -e "  ${RED}✗ $_ko maillon(s) manquant(s) : cette ISO ne serait PAS prete a l emploi (2G + 4G +
         ↪  telephone).${NC}" >&2
        echo -e "    Corrige ci-dessus, ou OSMO_ISO_LTE_REQUIRED=0 pour sortir l ISO quand meme." >&2
        _iso88_down
        exit 1
    fi
```

```
    echo -e "  ${YELLOW}!${NC} $_ko maillon(s) manquant(s), ISO sortie quand meme (OSMO_ISO_LTE_REQUIRED=0)"
else
    echo -e "  ${GREEN}✓${NC} prete a l emploi : banc 2G, 4G et telephone au complet"
fi
unset _ko; unset -f _chk

# Fin du chroot de ce module : ce qu on a monte plus haut repart, sinon le
# squashfs de 90-iso.sh embarquerait /proc et /sys de l hote.
_iso88_down; unset _iso88_mnt; unset -f _iso88_up _iso88_down


# Fin de module : `. fichier` rend le statut de sa DERNIERE commande, et
# build-iso.sh tourne sous set -e. Un module qui finirait par un test faux
# ("[ ... ] && { ...; }") arreterait tout, sans un mot. Toujours 0 ici.
true
```

3.216 /opt/GSM/osmo-operator/iso_modules/89-home-chiffre.sh

12389 octets, 181 lignes → 181 lignes

```
#!/bin/bash
# iso_modules/89-home-chiffre.sh - etape 8i : second disque chiffre sur /home
# Source par build-iso.sh, dans l ordre des numeros : meme shell, memes
# variables, memes fonctions. Ne s execute pas seul. `return` en tete de
# module = "rien a faire ici" (c est ainsi que --arm saute une etape).

# ═══════════════════════════════════════════════════════════════════════
# Etape 8i : le disque systeme reste lisible, les secrets vont sur le second
# ═══════════════════════════════════════════════════════════════════════
# LE MODELE, en une phrase : tout ce qui est systeme est lisible, tout ce qui
# tient au trousseau est chiffre.
#
# Le disque systeme n est PAS chiffre, et c est voulu : le banc se pilote en
# root (VTY, netns, tcpdump), la session s ouvre sur root (etape 8d), et un
# operateur doit pouvoir sortir une trace, relire un journal ou reparer la
# machine depuis une cle live sans avoir a dechiffrer quoi que ce soit.
#
# Ce qui n a rien a faire en clair, ce sont les SECRETS : trousseau GNOME,
# cookies et jetons du navigateur, profils. Ceux-la vivent sur un SECOND
# disque, chiffre en LUKS2 et monte sur /home. Le compte osmocom - celui que
# l etape 8d cree non privilegie - y a son home, et les applications a secrets
# tournent SOUS LUI depuis la session de root (run-as-owner).
#
# AU DEMARRAGE la machine demande « decrypt /home -passphrase- or Enter », sur
# la console texte (plymouth est quitte juste avant : son clavier n est pas
# fiable, voir unlock-home), la phrase tapee en ****.
# Entree a vide, se tromper de phrase ou ne rien repondre ne bloque JAMAIS le boot :
# la session root s ouvre, /home reste vide, et les memes applications
# demarrent sans secrets. C est ce qui rend le dispositif utilisable sur un
# banc qu on redemarre dix fois par jour.
#
# OU SE FAIT LE CHIFFREMENT, DESORMAIS : DANS CALAMARES.
# [2026-09-25] Le chiffrement n est plus une operation d apres-coup lancee
# depuis une session de bureau. Il se fait A L INSTALLATION :
#   - installer/calamares/modules/partition.conf decoupe le disque en DEUX,
#     racine + /home, et marque la racine "noEncrypt". Cocher « chiffrer le
#     systeme » ne chiffre donc que /home, en LUKS2 ;
#   - crypthome-postinstall, lance par shellprocess@osmo dans la cible,
#     raccorde ce que l installeur a produit : crypttab et fstab en "noauto",
#     /etc/default/crypthome rempli (UUID, mapping, ET le compte de session
#     comme proprietaire), trousseau de root pose sur le volume.
#
# POURQUOI PAS DANS LA SESSION. init-crypthome.sh recopie /home pendant que la
# session qui l a lance tient ce meme /home : la copie est prise a chaud,
```

```
# l ancien contenu doit etre evacue au redemarrage suivant, et le basculement
# s etale sur deux demarrages. A l installation, le disque est vierge et
# personne ne tient rien - il n y a ni copie, ni evacuation, ni deuxieme temps.
#
# CE MODULE, LUI, NE CHIFFRE TOUJOURS RIEN. Il pose les outils dans le rootfs,
# pour la cle live et pour la cible. Sur la cle live, et sur une machine
# installee sans avoir coche la case, CRYPTHOME_UUID reste vide : unlock-home
# constate qu il n y a rien a faire et sort.
#
# init-crypthome.sh RESTE, pour le cas qu il a toujours servi et que Calamares
# ne couvre pas : chiffrer un SECOND disque ajoute apres coup a une machine
# deja installee.
# ── "CE WRAPPER AVAIT ETE RETIRE" : NON, PAS CELUI-LA ──────────────────────
# L etape 8f (86-finitions.sh) raconte la suppression de ~180 lignes de
# plomberie qui relancaient CHROMIUM sous osmocom par runuser et xhost. Elle a
# eu lieu, elle etait justifiee, et il ne faut pas defaire ce module en croyant
# que c est la meme chose qui revient.
#
# Ce n est pas la meme chose. Ce wrapper-la existait pour UNE raison technique
# - "Running as root without --no-sandbox is not supported" - et Firefox, qui
# a remplace Chromium, n a pas cette contrainte : il demarre en root sans
# intermediaire. La raison avait donc disparu, et le code avec elle.
#
# run-as-owner repond a une question qui n a rien a voir : OU S ECRIVENT LES
# SECRETS. Firefox demarre tres bien en root - et ecrit alors ses cookies, ses
# jetons et son trousseau sur le disque systeme EN CLAIR. C est precisement ce
# qu on ne veut pas. Le compte n est pas un contournement de bac a sable, c est
# l endroit ou vivent les secrets : son home est sur le disque chiffre.
#
# Le pattern lui-meme n a d ailleurs jamais quitte le depot : le wrapper vlc de
# l etape 8e (85-installeur-bureau.sh) fait exactement cela - xhost, runuser,
# PULSE_SERVER, XDG_RUNTIME_DIR. run-as-owner le generalise, pour toutes les
# applications a secrets au lieu d une seule, et sur condition : le disque doit
# etre dechiffre, sinon l application demarre en root comme avant.
_CH_SRC="$DIR/configs/crypthome"
if [ ! -d "$_CH_SRC" ]; then
    echo -e "  ${YELLOW}!${NC} configs/crypthome absent - etape 8i sautee"
    return 0
fi

echo -e "${GREEN}[8i/9] Second disque chiffre sur /home (outils, rien n est chiffre ici)...${NC}"

# Le compte qui portera le home chiffre. C est osmocom, le compte non
# privilegie de l etape 8d - surchargeable pour une image qui en nommerait un
# autre.
_CH_OWNER="${OSMO_ISO_CRYPTHOME_OWNER:-osmocom}"

# ── LES OUTILS ──────────────────────────────────────────────────────────────
# unlock-home / lock-home en sbin (ils montent et demontent), run-as-owner et
# crypthome-session-setup en bin (ils sont appeles depuis la session).
install -d "$ROOTFS/usr/local/sbin" "$ROOTFS/usr/local/bin" "$ROOTFS/root" \
           "$ROOTFS/etc/systemd/system" "$ROOTFS/etc/default" "$ROOTFS/usr/local/share/applications"
install -m 0755 "$_CH_SRC/unlock-home"            "$ROOTFS/usr/local/sbin/unlock-home"
install -m 0755 "$_CH_SRC/lock-home"              "$ROOTFS/usr/local/sbin/lock-home"
install -m 0755 "$_CH_SRC/crypthome-fond"         "$ROOTFS/usr/local/sbin/crypthome-fond"   # [2026-10-24] image
 ↪  de la question LUKS
install -m 0755 "$_CH_SRC/refresh-owner-apps"     "$ROOTFS/usr/local/sbin/refresh-owner-apps"
install -m 0755 "$_CH_SRC/run-as-owner"           "$ROOTFS/usr/local/bin/run-as-owner"
install -m 0755 "$_CH_SRC/crypthome-session-setup" "$ROOTFS/usr/local/bin/crypthome-session-setup"
install -m 0700 "$_CH_SRC/init-crypthome.sh"      "$ROOTFS/root/init-crypthome.sh"
# Lance par Calamares dans la cible (shellprocess@osmo) : il raccorde le /home
# que l installeur vient de chiffrer. En sbin parce qu il touche crypttab,
# fstab et /etc/default/crypthome.
install -m 0755 "$_CH_SRC/crypthome-postinstall"  "$ROOTFS/usr/local/sbin/crypthome-postinstall"
install -m 0644 "$_CH_SRC/unlock-home.service"    "$ROOTFS/etc/systemd/system/unlock-home.service"
```

```
# La configuration : __OWNER__ est le seul reglage qui change d une image a
# l autre. CRYPTHOME_UUID reste VIDE - il est ecrit par init-crypthome.sh, sur
# la machine, quand le disque existe vraiment.
sed "s/__OWNER__/$_CH_OWNER/" "$_CH_SRC/crypthome.default" > "$ROOTFS/etc/default/crypthome"
chmod 0644 "$ROOTFS/etc/default/crypthome"

# ── LE SERVICE ──────────────────────────────────────────────────────────
# Active des l image : sans UUID configure il ne fait rien, mais il est en
# place le jour ou l operateur lance init-crypthome.sh - qui n a alors plus
# qu a ecrire l UUID. Un service a activer apres coup est un service qu on
# oublie d activer.
chroot "$ROOTFS" systemctl enable unlock-home.service 2>/dev/null || \
    echo -e "  ${YELLOW}!${NC} unlock-home.service non active (systemctl indisponible dans le rootfs)"

# ── LE TROUSSEAU DE ROOT N EXISTE QU EN CHIFFRE ─────────────────────────
# Lien vers le disque chiffre, pose MAINTENANT - avant que la premiere session
# de root ne cree un vrai repertoire a cet endroit. Disque non dechiffre =
# lien mort = gnome-keyring ne peut RIEN ecrire en clair sur le disque
# systeme. Un trousseau absent vaut mieux qu un trousseau lisible : c est
# exactement le compromis qu on veut ici.
install -d -m 0700 "$ROOTFS/root/.local/share"
rm -rf "$ROOTFS/root/.local/share/keyrings"
ln -sfn /home/.crypthome/root-keyrings "$ROOTFS/root/.local/share/keyrings"

# ── LE SON DU COMPTE PROPRIETAIRE ───────────────────────────────────────
# PulseAudio tourne en mode SYSTEME sur ce banc : le demon appartient au compte
# `pulse` et n accepte que les membres de pulse-access. Le navigateur porte le
# micro du tableau de bord (voir l etape 8f) et il tourne desormais sous le
# compte proprietaire : sans ces groupes, getUserMedia rend « NotFoundError »
# pendant que pactl liste les peripheriques en RUNNING - et on cherche des
# heures du cote « permission micro » ou « pilote son ».
chroot "$ROOTFS" bash -c "
    getent group pulse-access >/dev/null 2>&1 && usermod -aG pulse-access,audio,video $_CH_OWNER
" 2>/dev/null || echo -e "  ${YELLOW}!${NC} groupes audio non ajoutes a ${CYAN}$_CH_OWNER${NC}"

# ── LA SESSION ──────────────────────────────────────────────────────────
# L autostart prepare la bascule des l ouverture de la session root :
# autorisation sur le serveur X et session utilisateur du compte proprietaire.
# Sans lui, le premier lancement de Firefox attend logind plusieurs secondes
# sans rien afficher, et on croit que le clic n a pas pris.
if [ -d "$ROOTFS/etc/gdm3" ] || [ "${ISO_DESKTOP:-0}" = "1" ]; then
    install -d "$ROOTFS/root/.config/autostart" "$ROOTFS/etc/skel/.config/autostart"
    install -m 0644 "$_CH_SRC/crypthome-session.desktop" \
        "$ROOTFS/root/.config/autostart/crypthome-session.desktop"
    install -m 0644 "$_CH_SRC/crypthome-session.desktop" \
        "$ROOTFS/etc/skel/.config/autostart/crypthome-session.desktop"

    # Les lanceurs des applications a secrets passent par run-as-owner. Dans
    # le chroot, et pas ici : la liste depend de ce qui est REELLEMENT
    # installe dans le rootfs, et seul le chroot le sait.
    chroot "$ROOTFS" /usr/local/sbin/refresh-owner-apps 2>/dev/null | sed 's/^/  /' || \
        echo -e "  ${YELLOW}!${NC} lanceurs non rediriges (a refaire avec refresh-owner-apps)"
    echo -e "  ${GREEN}✓${NC} applications a secrets : sous ${CYAN}$_CH_OWNER${NC} quand /home est dechiffre,
     ↪  sous ${CYAN}root${NC} sinon"
fi

# ── CE QU IL FAUT DANS LE ROOTFS ────────────────────────────────────────
# cryptsetup pour ouvrir le volume, rsync pour la recopie du home par
# init-crypthome.sh, x11-xserver-utils pour xhost. Absents, tout le dispositif
# echoue au pire moment : au premier demarrage apres chiffrement.
for _p in cryptsetup rsync xhost; do
    chroot "$ROOTFS" bash -c "command -v $_p >/dev/null 2>&1" || \
        echo -e "  ${YELLOW}!${NC} ${CYAN}$_p${NC} absent du rootfs - requis par le dispositif de chiffrement"
done
```

```
echo -e "  ${GREEN}✓${NC} outils poses : ${CYAN}unlock-home${NC}, ${CYAN}lock-home${NC},
 ↪  ${CYAN}crypthome-postinstall${NC} (Calamares), ${CYAN}init-crypthome.sh${NC} (second disque)"
echo -e "  ${GREEN}✓${NC} a l installation : cocher ${CYAN}« chiffrer »${NC} chiffre ${CYAN}/home${NC} seul, la
 ↪  racine reste lisible"
echo -e "  ${GREEN}✓${NC} demarrage : ${CYAN}« decrypt /home -passphrase- or Enter »${NC} sur la console - le
 ↪  boot aboutit quelle que soit la reponse"

# Fin de module : `. fichier` rend le statut de sa DERNIERE commande, et
# build-iso.sh tourne sous set -e. Un module qui finirait par un test faux
# ("[ ... ] && { ...; }") arreterait tout, sans un mot. Toujours 0 ici.
true
```

3.217 /opt/GSM/osmo-operator/iso_modules/90-iso.sh

22123 octets, 392 lignes → 392 lignes

```
#!/bin/bash
# iso_modules/90-iso.sh - etape 9 : squashfs, noyau, menu GRUB, /.disk
# Source par build-iso.sh, dans l ordre des numeros : meme shell, memes
# variables, memes fonctions. Ne s execute pas seul. `return` en tete de
# module = "rien a faire ici" (c est ainsi que --arm saute une etape).

# --arm : pas de squashfs ni de GRUB : 92-rpi-image ecrit l image SD.
if [ "${ISO_ARCH:-amd64}" = "arm64" ]; then return 0; fi

# ── Creation du squashfs et de l'ISO ──────────────────────────────────
echo -e "${GREEN}[9/9] Squashfs et ISO...${NC}"
mkdir -p "$ISOROOT/live" "$ISOROOT/boot/grub"

# 2>/dev/null || true : sous pipefail, un ls sans resultat faisait sortir le
# script AVANT le message « Kernel absent » qui suit.
VMLINUZ=$(ls "$ROOTFS"/boot/vmlinuz-* 2>/dev/null | sort -V | tail -1 || true)
INITRD=$(ls "$ROOTFS"/boot/initrd.img-* 2>/dev/null | sort -V | tail -1 || true)
if [ -z "$VMLINUZ" ] || [ -z "$INITRD" ]; then echo -e "${RED}Kernel absent${NC}"; exit 1; fi
echo -e "  Kernel: ${CYAN}$(basename "$VMLINUZ")${NC}"

# ── Compression du squashfs : zstd, et non xz ─────────────────────────
# Le noyau embarque est compile avec CONFIG_SQUASHFS_DECOMP_SINGLE=y : UN SEUL
# flux de decompression, serialise par un mutex. Ajouter des vCPU a la VM n'y
# change donc rien - c'est le seul chiffre qui compte quand l'ISO est lue a la
# demande, et c'est celui qu'on regarde le moins.
#
# Mesure faite sur ce depot, meme contenu (360 Mo de /usr/bin de l'ISO lite),
# decompression a UN thread :
#
#     -comp xz -Xbcj x86    92,0 Mo    3,82 s
#     -comp zstd -level 19 106,9 Mo    0,41 s
#
# 16 % d'ISO en plus contre 9x en vitesse de lecture. Sur une image qui vit en
# machine virtuelle, dont chaque fichier ouvert au demarrage coute une
# decompression de bloc de 1 Mo, l'arbitrage se tranche tout seul.
#
# Le repli sur xz n'est pas de la prudence decorative : mksquashfs n'a le zstd
# que depuis la 4.4, et il n'est compile que si la libzstd etait la. On SONDE
# donc l'outil au lieu de lire son numero de version - une compression d'essai
# repond juste, une comparaison de versions non.
_squash_zstd_ok() {
    local t rc
    t="$(mktemp -d)" || return 1
    : > "$t/probe"
    mksquashfs "$t" "$t.sqfs" -comp zstd -no-progress -noappend >/dev/null 2>&1
    rc=$?
    rm -rf "$t" "$t.sqfs"
    return $rc
```

```
}

if _squash_zstd_ok; then
    SQUASH_COMP=(-comp zstd -Xcompression-level 19)
    echo -e "  Compression: ${CYAN}zstd -19${NC} (lecture ~9x plus rapide que xz a un thread)"
else
    SQUASH_COMP=(-comp xz -Xbcj x86)
    echo -e "  Compression: ${YELLOW}xz${NC} (mksquashfs sans zstd - ISO plus petite, mais lente a lire)"
fi

# [2026-09-02] PLUS D EXCLUSION de boot/vmlinuz-* ni de boot/initrd* : ce
# squashfs n est pas seulement la racine du live, c est aussi la SOURCE de
# Calamares (unpackfs.conf). Sans noyau dedans, le systeme installe n avait ni
# vmlinuz ni initrd : grub-mkconfig n ecrivait aucune entree Linux et la
# machine ne demarrait plus apres « installation reussie ». Les ~100 Mo de
# plus sont deja compresses (le noyau l est), ils ne pesent presque rien ici.
# ── Controle d integrite : les fichiers des paquets contre leurs md5 dpkg ────
# [2026-09-09] Le build du 8 au soir a livre une ISO ou ~440 fichiers
# differaient d UN bit (le bit 0 tombe a zero, jusque dans des noms de
# fichiers : « __pyc`che__ », « usr.share ») de ce que dpkg avait pose :
# libgjs, libmozjs, Xwayland, firefox, mesa, libwireshark... Resultat : GNOME
# Shell segfaultait dans libmozjs a chaque lancement, « Oh no! Something has
# gone wrong » au premier boot (VirtualBox comme QEMU). Les .deb sont
# verifies par leur CRC xz a l extraction : l alteration s est faite APRES,
# sur l hote (RAM ou disque) - une machine qui flanche sous la charge du
# build. Aucun maillon de la chaine ne le voit ; on compare donc les fichiers
# des paquets a leur md5 dpkg.
#
# [2026-09-14] Le controle ne se faisait qu APRES coup, sur le squashfs relu :
# il englobait tout le trajet, de l ecriture par dpkg jusqu a la relecture, et
# ne savait donc pas dire OU le bit avait bascule. On controle desormais DEUX
# FOIS, avec exactement le meme filtre - sans quoi comparer les deux listes
# n aurait aucun sens :
#
#   rootfs altere            -> c est ANTERIEUR a la compression (dpkg, ou une
#                               etape du build qui reecrit par-dessus un paquet)
#   rootfs sain + sqfs altere-> c est mksquashfs ou la relecture
#
# Ce que le build modifie lui-meme est ecarte : /opt et /root (sources et
# venv), /usr/local, MongoDB, les unites osmo-* (recopiees du docker),
# calamares.desktop (NoDisplay), live-boot (toram), os-release ; les fichiers
# supprimes (docs, locales) ne comptent pas ; ni les libs de la cloture ldd de
# l etape 8b, qui remplacent CELLES D APT a dessein ($WORK/closure.list).
# [2026-10-01] Et usr/bin/screenfetch : 86-finitions.sh y neutralise au sed
# l easter-egg qui melange les lignes du logo (cinq valeurs sur mille). Le
# fichier differe donc de son paquet A DESSEIN, sur toutes les images.
#
# Et les listes sont desormais HORODATEES. Le critere qui tranche - « les
# memes fichiers a chaque build » (une etape du build les reecrit) contre « un
# tirage different a chaque fois » (l hote qui flanche) - etait inverifiable
# tant qu un fichier unique etait ecrase a chaque build : le rapport le pose
# maintenant tout seul, en comparant au build precedent.
# Environ 40 s par passe. OSMO_ISO_NO_VERIFY=1 pour passer outre.
_vfy_stamp="$(date +%Y%m%d-%H%M%S)"

# Le controle, applique a une racine quelconque. Pose _vfy_out (les chemins en
# echec, sans le « : FAILED » - on compare ces listes d un build a l autre),
# _vfy_n (leur nombre) et _vfy_prev (la liste du build precedent de meme
# etiquette, vide au premier build).
_vfy_paquets() {   # $1 = etiquette (rootfs|squashfs), $2 = racine a controler
    local _k="$1" _r="$2" _skip="$WORK/closure.list"
    [ -s "$_skip" ] || _skip=/dev/null
    _vfy_out="/var/tmp/osmo-iso-verify-${_vfy_stamp}-${_k}.failed"
    ( cd "$_r" && cat var/lib/dpkg/info/*.md5sums 2>/dev/null \
```

```
        | grep -vE '^[0-9a-f]{32}  (opt/|root/|usr/local/|mongodb|usr/lib/systemd/system/osmo-
        ↪  |usr/share/applications/calamares\.desktop$|lib/live/boot/|usr/lib/os-release$|usr/bin/screenfetch$)'
        ↪  \
        | awk -v skip="$_skip" 'BEGIN{while((getline l < skip)>0) s[l]=1} !(substr($0,35) in s)' \
        | md5sum -c --quiet 2>/dev/null | grep -v 'open or read' \
        | sed 's/: FAILED$//' ) > "$_vfy_out" || true
    _vfy_n=$(wc -l < "$_vfy_out")
    # Le chemin historique reste valable : il pointe sur la derniere passe.
    ln -sf "$_vfy_out" /var/tmp/osmo-iso-verify.failed
    _vfy_prev=$(ls -1t /var/tmp/osmo-iso-verify-*-"${_k}".failed 2>/dev/null \
        | grep -vF "$_vfy_stamp" | head -1) || true
    # Les 20 dernieres : de quoi juger la stabilite, pas de quoi remplir /var/tmp.
    ls -1t /var/tmp/osmo-iso-verify-*-"${_k}".failed 2>/dev/null \
        | tail -n +21 | xargs -r rm -f || true
}

# Le verdict, une fois qu une liste n est pas vide. C est la comparaison au
# build precedent qui porte le diagnostic, pas le nombre de fichiers.
_vfy_verdict() {
    local _c
    head -8 "$_vfy_out" | sed 's/^/      /'
    if [ "$_vfy_n" -gt 8 ]; then echo "      ... (liste complete : $_vfy_out)"; fi
    if [ -n "$_vfy_prev" ]; then
        _c=$(grep -cxF -f "$_vfy_out" "$_vfy_prev" 2>/dev/null) || _c=0
        if [ "${_c:-0}" -gt 0 ]; then
            echo -e "  ${YELLOW}${_c} de ces fichiers echouaient DEJA au build precedent${NC}"
            echo -e "  ${YELLOW}($(basename "$_vfy_prev")) : la liste est STABLE. Ce n est pas de la${NC}"
            echo -e "  ${YELLOW}RAM qui flanche - c est une etape du build qui reecrit ces fichiers${NC}"
            echo -e "  ${YELLOW}par-dessus leur paquet sans etre ecartee du filtre ci-dessus.${NC}"
        else
            echo -e "  ${YELLOW}Aucun de ces fichiers n echouait au build precedent${NC}"
            echo -e "  ${YELLOW}($(basename "$_vfy_prev")) : le tirage CHANGE a chaque fois. Aucune${NC}"
            echo -e "  ${YELLOW}etape du build ne vise des fichiers au hasard : c est l hote.${NC}"
            echo -e "  ${YELLOW}RAM non-ECC : rien ne sera jamais signale (ni EDAC, ni dmesg).${NC}"
            # [2026-09-14] MESURE, sur le rootfs du build de 11h59. Le meme
            # pipeline que la passe 1, rejoue sur le MEME rootfs intact :
            #   juste apres 2 x 160 s de zstd -19 sur 32 coeurs (94-95 C) -> 18 echecs
            #   la machine refroidie (78 C au depart)                     ->  0, deux fois
            # et les fichiers incrimines, relus un par un cache vide, rendent
            # exactement le md5 de dpkg. Les octets du disque sont BONS : ce
            # qui se corrompt, c est la lecture sous charge. D ou le conseil.
            echo -e "  ${YELLOW}La faute n apparait que SOUS CHARGE SOUTENUE sur tous les coeurs :${NC}"
            echo -e "  ${YELLOW}memtest86+ peut passer une nuit entiere sans rien voir (il ne${NC}"
            echo -e "  ${YELLOW}chauffe pas le CPU). Refaire plutot ce build a froid, et le${NC}"
            echo -e "  ${YELLOW}refaire avec OSMO_ISO_SQUASH_JOBS=4 : si les echecs disparaissent${NC}"
            echo -e "  ${YELLOW}a 4 coeurs et reviennent a 32, c est l alimentation/le CPU sous${NC}"
            echo -e "  ${YELLOW}charge, pas une barrette morte (un i9-14900HX est concerne par${NC}"
            echo -e "  ${YELLOW}l instabilite Raptor Lake : microcode a jour = degradation${NC}"
            echo -e "  ${YELLOW}arretee, pas reparee). Sinon : memtest86+, et temperatures.${NC}"
        fi
    else
        echo -e "  ${YELLOW}Pas de build precedent a comparer. Relancer tel quel : meme liste =${NC}"
        echo -e "  ${YELLOW}une etape du build reecrit ces fichiers ; liste differente = l hote.${NC}"
    fi
}

# Passe 1 : le rootfs tel que dpkg et le build l ont laisse. Avant les 20 Go
# de compression - inutile de les payer pour retrouver l alteration a la sortie.
if [ "${OSMO_ISO_NO_VERIFY:-0}" != "1" ]; then
    _vfy_paquets rootfs "$ROOTFS"
    if [ "$_vfy_n" -eq 0 ]; then
        echo -e "  ${GREEN}✓${NC} rootfs conforme a dpkg avant compression"
    else
        echo -e "  ${RED}✗ rootfs ALTERE AVANT compression : ${_vfy_n} fichier(s)${NC}"
        _vfy_verdict
        echo -e "  ${YELLOW}L alteration est ANTERIEURE a mksquashfs : la compression et sa${NC}"
```

```
        echo -e "  ${YELLOW}relecture sont hors de cause, ne pas les chercher.${NC}"
        echo -e "  ${YELLOW}OSMO_ISO_NO_VERIFY=1 pour livrer quand meme (l ISO plantera).${NC}"
        exit 1
    fi
fi

# ── OSMO_ISO_SQUASH_JOBS : brider la compression ──────────────────────
# [2026-09-14] mksquashfs prend TOUS les coeurs par defaut. Sur cet hote (32
# threads, zstd -19) c est le seul moment du build ou la machine tire sa
# puissance maximale pendant trois minutes d affilee, et c est exactement la
# que le controle d integrite ci-dessous trouve des octets faux - jamais a
# froid (mesure : 18 echecs juste apres, 0 sur la meme ISO une fois refroidie).
# Cette variable sert deux fois : a TRANCHER (les echecs disparaissent-ils a 4
# coeurs ?) et, si oui, a livrer quand meme - plus lentement, mais juste.
# Vide par defaut : mksquashfs garde son comportement d origine.
SQUASH_JOBS=()
if [ -n "${OSMO_ISO_SQUASH_JOBS:-}" ]; then
    SQUASH_JOBS=(-processors "$OSMO_ISO_SQUASH_JOBS")
    echo -e "  ${CYAN}compression bridee a ${OSMO_ISO_SQUASH_JOBS} coeur(s) (OSMO_ISO_SQUASH_JOBS)${NC}"
fi
mksquashfs "$ROOTFS" "$ISOROOT/live/filesystem.squashfs" \
    "${SQUASH_COMP[@]}" "${SQUASH_JOBS[@]}" -b 1M \
    -e 'var/cache/apt' -e 'var/lib/apt/lists' \
    -no-progress
echo -e "  ${GREEN}✓${NC} squashfs $(du -sh "$ISOROOT/live/filesystem.squashfs"|cut -f1)"

# ── Passe 2 : le squashfs relu, contre les memes md5 dpkg ─────────────
# La passe 1 (avant mksquashfs) a dit que le rootfs etait sain ; ce qui
# echoue ici n a donc pu etre altere que par la compression elle-meme, son
# ecriture, ou cette relecture. Voir le pave de la passe 1 pour le filtre.
if [ "${OSMO_ISO_NO_VERIFY:-0}" != "1" ]; then
    _vfy_mnt="$WORK/sqfs-verify"
    mkdir -p "$_vfy_mnt"
    if mount -t squashfs -o loop,ro "$ISOROOT/live/filesystem.squashfs" "$_vfy_mnt" 2>/dev/null; then
        _vfy_paquets squashfs "$_vfy_mnt"
        umount "$_vfy_mnt" 2>/dev/null || true
        if [ "$_vfy_n" -eq 0 ]; then
            echo -e "  ${GREEN}✓${NC} squashfs relu : les fichiers des paquets sont conformes a dpkg"
        else
            echo -e "  ${RED}✗ squashfs ALTERE : ${_vfy_n} fichier(s) ne correspondent plus a leur paquet${NC}"
            _vfy_verdict
            echo -e "  ${YELLOW}Le rootfs, lui, etait conforme juste avant (passe 1). NE PAS${NC}"
            echo -e "  ${YELLOW}chercher un bug de mksquashfs : il est deterministe, il donnerait${NC}"
            echo -e "  ${YELLOW}les MEMES fichiers a chaque fois. Ce qu il compresse, ce sont les${NC}"
            echo -e "  ${YELLOW}pages que la lecture du rootfs vient de remplir - 20 Go lus a${NC}"
            echo -e "  ${YELLOW}pleine charge - et c est LA que les octets basculent. Il les${NC}"
            echo -e "  ${YELLOW}compresse fidelement ensuite. Compression a tous les coeurs = le${NC}"
            echo -e "  ${YELLOW}moment ou l hote chauffe et consomme le plus de tout le build.${NC}"
            echo -e "  ${YELLOW}OSMO_ISO_NO_VERIFY=1 pour livrer quand meme (l ISO plantera).${NC}"
            exit 1
        fi
    else
        echo -e "  ${YELLOW}!${NC} squashfs non relu (mount loop impossible) : seule la passe 1 a parle"
    fi
    rmdir "$_vfy_mnt" 2>/dev/null || true
    unset _vfy_mnt _vfy_out _vfy_n _vfy_prev _vfy_stamp
fi

cp "$VMLINUZ" "$ISOROOT/boot/vmlinuz"
cp "$INITRD"  "$ISOROOT/boot/initrd.img"

# ── Menu GRUB ─────────────────────────────────────────────────────────
# Les chiffres de RAM annonces sont CALCULES sur le squashfs qui vient d'etre
# ecrit, pas recopies d'un ancien build. Les libelles en dur ("RAM ~6 Go")
# etaient faux des que l'image maigrissait, et une consigne fausse coute plus
# cher que pas de consigne : on dimensionne la VM sur elle.
```

```
SQ_MB=$(( $(stat -Lc%s "$ISOROOT/live/filesystem.squashfs") / 1048576 ))
# toram recopie le squashfs dans un tmpfs, puis le systeme tourne par-dessus :
# la taille du fichier, plus 2 Go pour le reste, arrondi au Go superieur.
RAM_TORAM_GB=$(( (SQ_MB + 2048 + 1023) / 1024 ))
SQ_GB=$(awk -v m="$SQ_MB" 'BEGIN{printf "%.1f", m/1024}')

# [2026-10-23] Theme « osmo » (tools/grub-theme-render.py), le meme que sur le
# systeme installe. S il ne se charge pas (polices absentes, GRUB signe sans
# gfxterm), le bloc « if loadfont » est saute et le menu reste en texte.
python3 "$DIR/tools/grub-theme-render.py" --out "$ISOROOT/boot/grub/themes/osmo" >/dev/null 2>&1 \
    || echo -e "  ${YELLOW}!${NC} grub : rendu du theme echoue - menu du live en texte"

cat > "$ISOROOT/boot/grub/grub.cfg" <<GRUB
# ATTENTION - ce fichier est GENERE par build-iso.sh. Le modifier dans l'ISO ne
# survit pas au build suivant.
#
# Sous VirtualBox, deux reglages evitent des minutes d'attente sur l'entree
# "en RAM" : attacher l'ISO au controleur SATA plutot qu'IDE, et cocher
# "Utiliser le cache E/S de l'hote" dessus.

set default=0
set timeout=5

if loadfont /boot/grub/themes/osmo/osmo-18.pf2; then
    loadfont /boot/grub/themes/osmo/osmo-24.pf2
    loadfont /boot/grub/themes/osmo/osmom-22.pf2
    set gfxmode=auto
    insmod all_video
    insmod gfxterm
    insmod png
    terminal_output gfxterm
    set theme=/boot/grub/themes/osmo/theme.txt
fi

# TROIS entrees, et le reste dans un sous-menu. Cinq lignes dont deux doublons
# "verbose", c est un menu ou l on cherche - alors que le choix reel n en compte
# que trois : lire depuis le medium, copier en RAM, ecrire sur le medium.
menuentry "osmo-operator" {
    linux  /boot/vmlinuz boot=live quiet
    initrd /boot/initrd.img
}

# "toram" tout court ferait recopier a live-boot le MEDIUM ENTIER dans un tmpfs
# (lib/live/boot/9990-toram-todisk.sh) : le squashfs, mais AUSSI l initrd de
# 82 Mo, le vmlinuz et efi.img. Et comme rsync n est pas dans l initrd, la copie
# se fait par "cp -a", qui n affiche RIEN - avec "quiet" en plus, l ecran reste
# fige plusieurs minutes sans le moindre signe de vie, et on croit a un
# plantage. D ou "toram=filesystem.squashfs" (seul le squashfs est copie, et le
# tmpfs est dimensionne sur lui) et l absence de "quiet" ici : la copie se voit.
menuentry "osmo-operator - en RAM (copie ${SQ_GB} Go - ${RAM_TORAM_GB} Go de RAM mini)" {
    linux  /boot/vmlinuz boot=live toram=filesystem.squashfs
    initrd /boot/initrd.img
}

# Sans persistance, la racine est un overlay tmpfs : tout ce qui s ecrit vit en
# RAM et meurt au reboot - configs SS7 posees a la main, base HLR, journaux.
# PAS de toram ici, et c est le point : avec toram le systeme est recopie en RAM
# et l overlay y reste, ce qui annulerait l interet.
#
# Il faut un volume ETIQUETE "persistence" portant un persistence.conf dont la
# seule ligne utile est "/ union" :
#   sudo mkfs.ext4 -L persistence /dev/sdX3
#   sudo mount /dev/sdX3 /mnt && echo "/ union" | sudo tee /mnt/persistence.conf
# En VM, un second disque suffit. Sans volume ainsi etiquete, cette entree
# demarre comme un live ordinaire : rien ne casse, rien n est garde.
menuentry "osmo-operator - persistant (ecrit sur le medium)" {
```

```
    linux  /boot/vmlinuz boot=live persistence persistence-encryption=none quiet
    initrd /boot/initrd.img
}

# Les variantes verbose ne servent qu au diagnostic : elles sont les memes
# lignes de commande sans "quiet". Elles restent atteignables, mais elles ne
# tiennent plus la moitie du menu.
# [2026-09-08] SANS ACCELERATION GRAPHIQUE. Un GPU que le noyau 6.8 de l ISO
# ne sait pas (AMD RDNA recent sans son firmware, NVIDIA sans nouveau) donne
# un ecran noir ou un plantage a l amorcage : nomodeset laisse le framebuffer
# du firmware, le bureau tourne en logiciel, et l on peut au moins installer.
menuentry "osmo-operator - sans acceleration graphique (nomodeset : ecran noir, plantage AMD/NVIDIA)" {
    linux  /boot/vmlinuz boot=live nomodeset quiet
    initrd /boot/initrd.img
}

submenu "Options (demarrage verbeux)" {
    menuentry "osmo-operator - verbose" {
        linux  /boot/vmlinuz boot=live
        initrd /boot/initrd.img
    }
    menuentry "osmo-operator - persistant verbose" {
        linux  /boot/vmlinuz boot=live persistence persistence-encryption=none
        initrd /boot/initrd.img
    }
    menuentry "osmo-operator - persistant, sans acceleration graphique (nomodeset)" {
        linux  /boot/vmlinuz boot=live persistence persistence-encryption=none nomodeset
        initrd /boot/initrd.img
    }
}
GRUB
echo -e "  ${GREEN}✓${NC} menu GRUB : defaut = lecture depuis le medium ; toram annonce ${CYAN}${RAM_TORAM_GB}
 ↪  Go${NC} de RAM"

# ── /.disk/info : LE FICHIER SANS LEQUEL L'ISO S'ARRETE SUR "grub>" EN UEFI ──
# Ce fichier vide d'apparence est ce que le GRUB signe d'Ubuntu CHERCHE pour se
# reperer. gcdx64.efi.signed porte un disque memoire dont la configuration est,
# mot pour mot :
#
#     if [ -z "$prefix" -o ! -e "$prefix" ]; then
#         if ! search --file --set=root /.disk/info; then
#             search --file --set=root /.disk/mini-info
#         fi
#         set prefix=($root)/boot/grub
#     fi
#     ... source $prefix/grub.cfg ... sinon source $cmdpath/grub.cfg
#
# Son prefixe compile vaut "/boot/grub" SANS peripherique : il se resout donc
# sur le disque d'ou le firmware a charge le binaire, c'est-a-dire la partition
# EFI (FAT) - qui ne contient pas /boot/grub. Le test "! -e $prefix" est vrai,
# et tout repose alors sur la recherche de /.disk/info. Ce fichier n'existait
# pas ici : la recherche echouait, $root restait sur la FAT, aucun grub.cfg
# n'etait trouve, et GRUB rendait la main sur son invite "grub>".
#
# POURQUOI CA MARCHAIT EN MACHINE VIRTUELLE. VirtualBox demarre en BIOS par
# defaut : c'est eltorito.img qui joue, construit ici avec -p /boot/grub sur
# le lecteur de boot, et il trouve sa configuration sans rien chercher. Le
# chemin UEFI - le seul qu'un portable recent emprunte - n'etait donc jamais
# exerce. La panne n'etait pas "sur ce portable", elle etait sur TOUT UEFI.
#
# Les ISO Ubuntu et Debian posent ce meme fichier, pour cette meme raison.
mkdir -p "$ISOROOT/.disk"
printf 'osmo-operator %s - live amd64 (%s)\n' "$VERSION" "$LABEL" > "$ISOROOT/.disk/info"
cp "$ISOROOT/.disk/info" "$ISOROOT/.disk/mini-info"
echo -e "  ${GREEN}✓${NC} ${CYAN}/.disk/info${NC} pose : le GRUB signe retrouve le medium en UEFI"
```

```
# Les modules EFI, la ou $prefix les cherchera. Le menu genere plus haut n'use
# que de commandes deja compilees dans le binaire signe (set, menuentry, linux,
# initrd, submenu), mais grub-mkrescue les deposait, et leur absence transforme
# le moindre "insmod" tape a l'invite en echec incomprehensible. 3 Mo.
if [ -d /usr/lib/grub/x86_64-efi ]; then
    mkdir -p "$ISOROOT/boot/grub/x86_64-efi"
    cp -a /usr/lib/grub/x86_64-efi/*.mod /usr/lib/grub/x86_64-efi/*.lst \
          "$ISOROOT/boot/grub/x86_64-efi/" 2>/dev/null || true
    # ATTENTION : surtout PAS de grub.cfg dans ce repertoire. La configuration
    # embarquee ci-dessus teste "$prefix/x86_64-efi/grub.cfg" AVANT
    # "$prefix/grub.cfg" : un fichier ici detournerait le menu.
    rm -f "$ISOROOT/boot/grub/x86_64-efi/grub.cfg"
fi


# Fin de module : `. fichier` rend le statut de sa DERNIERE commande, et
# build-iso.sh tourne sous set -e. Un module qui finirait par un test faux
# ("[ ... ] && { ...; }") arreterait tout, sans un mot. Toujours 0 ici.
true
```

3.218 /opt/GSM/osmo-operator/iso_modules/91-secure-boot.sh

13057 octets, 245 lignes → 245 lignes

```
#!/bin/bash
# iso_modules/91-secure-boot.sh - ISO Secure Boot (shim + grub signes) ou grub-mkrescue
# Source par build-iso.sh, dans l ordre des numeros : meme shell, memes
# variables, memes fonctions. Ne s execute pas seul. `return` en tete de
# module = "rien a faire ici" (c est ainsi que --arm saute une etape).

# --arm : pas de shim ni de xorriso sur le Pi.
if [ "${ISO_ARCH:-amd64}" = "arm64" ]; then return 0; fi

# ============================================================================
# SECURE BOOT - pourquoi cette etape ne peut pas rester grub-mkrescue
# ============================================================================
# grub-mkrescue CONSTRUIT son BOOTX64.EFI a la volee, a partir des modules de
# l'hote. Ce binaire n'est signe par personne. Sur une machine dont le Secure
# Boot est actif, le firmware refuse de le charger et n'affiche qu'une erreur de
# certificat - "Verification failed: (0x1A) Security Violation", ou pire un
# ecran qui retombe au menu de boot sans un mot. L'ISO etait donc inutilisable
# partout ou l'on ne peut pas desactiver Secure Boot dans le firmware, ce qui
# est le cas de la plupart des machines d'entreprise et de beaucoup de portables
# recents.
#
# LA CHAINE DE CONFIANCE, et pourquoi chaque maillon est celui-la :
#
#   firmware --(cle Microsoft)--> shimx64.efi.signed   paquet shim-signed
#            --(cle Canonical)--> gcdx64.efi.signed    paquet grub-efi-amd64-signed
#            --(cle Canonical)--> vmlinuz              linux-image-generic (deja signe)
#
# shim est le SEUL maillon signe par Microsoft, dont la cle est dans a peu pres
# tous les firmwares du marche. Il porte la cle Canonical et valide ce qu'il
# charge ensuite. On ne le fabrique pas : on copie celui du paquet.
#
# gcdx64 ET NON grubx64 - c'est le detail qui coute une soiree. Les deux sont
# signes par Canonical, mais leur PREFIXE compile differe :
#     grubx64.efi.signed  -> prefixe /EFI/ubuntu, cherche sa configuration sur
#                            la partition EFI ; sur une ISO elle n'y est pas, et
#                            GRUB tombe sur son invite "grub>" sans un message.
#     gcdx64.efi.signed   -> prefixe /boot/grub RELATIF AU MEDIUM DE BOOT, la
#                            variante faite pour l'optique. Il trouve donc
#                            $ISOROOT/boot/grub/grub.cfg, celui qu'on vient
#                            d'ecrire, sans stub ni duplication.
```

```
# On garde tout de meme un stub en /EFI/ubuntu/grub.cfg sur l'ESP : il ne sert
# que si le repli sur grubx64 s'est declenche, et il ne coute que 60 octets.
#
# CE QUE CETTE ETAPE NE FAIT PAS : signer quoi que ce soit. Aucune cle privee
# n'est manipulee, rien n'est a enroler par l'utilisateur (pas de MokManager a
# la premiere ouverture). On assemble des binaires deja signes par Microsoft et
# Canonical - c'est exactement ce que fait une ISO Ubuntu officielle.
#
# LE BIOS N'EST PAS ABANDONNE. L'image El Torito i386-pc est construite ici par
# grub-mkimage au format i386-pc-eltorito (celui qui embarque deja cdboot.img),
# et la MBR hybride par --grub2-mbr : une machine sans UEFI demarre comme avant.
#
# REPLI. Si un maillon manque sur l'hote (paquet non installe, architecture
# autre), on retombe sur grub-mkrescue - l'ISO d'avant, qui demarre partout SAUF
# en Secure Boot. On le DIT, en clair : une ISO non signee qui se presente comme
# signee, c'est une panne au premier deploiement.

SB_SHIM=""
for c in /usr/lib/shim/shimx64.efi.signed \
         /usr/lib/shim/shimx64.efi.signed.latest \
         /usr/lib/shim/shimx64.efi; do
    [ -f "$c" ] && { SB_SHIM="$c"; break; }
done
# gcdx64 d'abord (prefixe /boot/grub, fait pour l'optique), grubx64 en repli.
SB_GRUB="" ; SB_GRUB_KIND=""
for c in /usr/lib/grub/x86_64-efi-signed/gcdx64.efi.signed:gcd \
         /usr/lib/grub/x86_64-efi-signed/grubx64.efi.signed:grub; do
    [ -f "${c%:*}" ] && { SB_GRUB="${c%:*}"; SB_GRUB_KIND="${c##*:}"; break; }
done
SB_MM=""
for c in /usr/lib/shim/mmx64.efi.signed /usr/lib/shim/mmx64.efi; do
    [ -f "$c" ] && { SB_MM="$c"; break; }
done

# Le noyau doit l'etre aussi : shim valide GRUB, GRUB valide le noyau. Un
# vmlinuz non signe s'arrete sur "you need to load the kernel first" ou sur un
# refus de shim_lock, APRES le menu - donc la ou l'on ne soupconne plus l'ISO.
# Les noyaux Ubuntu "generic" le sont ; un noyau maison, non. On regarde la
# signature plutot que de faire confiance au nom du paquet.
SB_KERNEL_SIGNED=0
if command -v sbverify &>/dev/null; then
    sbverify --list "$ISOROOT/boot/vmlinuz" &>/dev/null && SB_KERNEL_SIGNED=1
elif grep -qa '~Module signature appended~\|sbat\|Canonical Ltd\. Secure Boot' "$ISOROOT/boot/vmlinuz"
 ↪  2>/dev/null; then
    SB_KERNEL_SIGNED=1
fi

SECURE_BOOT_OK=0
if [ -n "$SB_SHIM" ] && [ -n "$SB_GRUB" ] && command -v mmd &>/dev/null; then
    echo -e "${GREEN}[9/9] ISO Secure Boot (shim + grub signes)...${NC}"
    echo -e "  shim : ${CYAN}${SB_SHIM}${NC}"
    echo -e "  grub : ${CYAN}${SB_GRUB}${NC} (${SB_GRUB_KIND})"

    # ── La partition systeme EFI ──────────────────────────────────────
    # FAT16, et un PLANCHER DE 16 Mo. Le contenu ne pese que ~4 Mo (shim ~1 Mo,
    # grub signe ~2,3 Mo, MokManager ~0,9 Mo), mais FAT16 exige au moins 4085
    # clusters : mkfs.vfat refuse en dessous, avec
    #     mkfs.vfat: Attempting to create a too small or a too large filesystem
    # et l'etape entiere retombait alors en silence sur le repli non signe.
    # 16 Mo a 2 Ko par cluster (-s 4) font 8192 clusters - au large, et 16 Mo
    # sur une ISO de plusieurs Go ne se voient pas.
    SB_ESP="$WORK/efi.img"
    SB_KB=$(( ( $(stat -Lc%s "$SB_SHIM") + $(stat -Lc%s "$SB_GRUB") \
              + $( [ -n "$SB_MM" ] && stat -Lc%s "$SB_MM" || echo 0 ) ) / 1024 + 2048 ))
    [ "$SB_KB" -lt 16384 ] && SB_KB=16384
    rm -f "$SB_ESP"
```

```
    mkfs.vfat -F 16 -s 4 -n OSMOEFI -C "$SB_ESP" "$SB_KB" >/dev/null

    mmd   -i "$SB_ESP" ::/EFI ::/EFI/BOOT ::/EFI/ubuntu
    mcopy -i "$SB_ESP" "$SB_SHIM" ::/EFI/BOOT/BOOTX64.EFI
    mcopy -i "$SB_ESP" "$SB_GRUB" ::/EFI/BOOT/grubx64.efi
    [ -n "$SB_MM" ] && mcopy -i "$SB_ESP" "$SB_MM" ::/EFI/BOOT/mmx64.efi

    # Stub : DERNIER RECOURS de la configuration embarquee dans gcdx64 - elle
    # finit par "source $cmdpath/grub.cfg", et $cmdpath est le repertoire d'ou
    # le firmware a charge le binaire, donc /EFI/BOOT sur cette ESP. Il sert
    # aussi tel quel au repli grubx64 (prefixe /EFI/ubuntu).
    #
    # "search --set=root" ne touche PAS a la variable quand il echoue : on vise
    # donc une variable a nous, encore vide, pour pouvoir tester le resultat -
    # et on le DIT quand rien n'est trouve, plutot que de rendre la main a une
    # invite "grub>" que personne ne sait interpreter.
    cat > "$WORK/esp-grub.cfg" <<'ESPCFG'
search --no-floppy --file --set=osmodev /.disk/info
if [ -z "$osmodev" ]; then
    search --no-floppy --file --set=osmodev /boot/grub/grub.cfg
fi
if [ -n "$osmodev" ]; then
    set root=$osmodev
    set prefix=($osmodev)/boot/grub
    configfile ($osmodev)/boot/grub/grub.cfg
fi
echo "GRUB : ni /.disk/info ni /boot/grub/grub.cfg trouves sur les disques vus."
echo "Le medium n'est probablement pas lisible par le firmware a ce stade."
ESPCFG
    mcopy -i "$SB_ESP" "$WORK/esp-grub.cfg" ::/EFI/ubuntu/grub.cfg
    mcopy -i "$SB_ESP" "$WORK/esp-grub.cfg" ::/EFI/BOOT/grub.cfg

    # ── LE MEME ARBRE, AUSSI DANS L'ISO9660 ──────────────────────────────
    # L'ESP appendue suffit a demarrer depuis un DVD ou une cle ecrite en mode
    # image (dd, Rufus en mode DD). Elle ne suffit PAS a la methode la plus
    # repandue sous Windows : formater la cle en FAT et y COPIER le contenu de
    # l'ISO. Cette copie ne voit que l'ISO9660, ou /EFI/BOOT n'existerait pas -
    # la cle ne demarre alors pas en UEFI, sans que rien n'explique pourquoi.
    # C'est exactement ce que xorriso previent :
    #     WARNING : EFI boot equipment is provided but no directory /EFI/BOOT
    #     WARNING : will emerge in the ISO filesystem.
    # Quelques Mo dupliques ; les ISO Ubuntu font de meme.
    mkdir -p "$ISOROOT/EFI/BOOT"
    cp "$SB_SHIM" "$ISOROOT/EFI/BOOT/BOOTX64.EFI"
    cp "$SB_GRUB" "$ISOROOT/EFI/BOOT/grubx64.efi"
    [ -n "$SB_MM" ] && cp "$SB_MM" "$ISOROOT/EFI/BOOT/mmx64.efi"
    cp "$WORK/esp-grub.cfg" "$ISOROOT/EFI/BOOT/grub.cfg"
    mkdir -p "$ISOROOT/EFI/ubuntu"
    cp "$WORK/esp-grub.cfg" "$ISOROOT/EFI/ubuntu/grub.cfg"

    # ── L'amorce BIOS, construite ici puisqu'on n'appelle plus grub-mkrescue ──
    # i386-pc-eltorito embarque deja cdboot.img : cette image est directement
    # utilisable comme -eltorito-boot, pas de concatenation a faire.
    SB_BIOS="$WORK/eltorito.img"
    grub-mkimage -O i386-pc-eltorito -p /boot/grub -o "$SB_BIOS" \
        biosdisk iso9660 part_msdos part_gpt fat ext2 normal linux configfile \
        search search_label search_fs_uuid search_fs_file loopback gzio \
        all_video gfxterm videotest videoinfo test echo ls minicmd sleep \
        halt reboot chain 2>/dev/null

    if [ -s "$SB_BIOS" ]; then
        # -eltorito-alt-boot separe les deux entrees du catalogue : la premiere
        # (BIOS) et la seconde (UEFI). "--interval:appended_partition_2" designe
        # la partition qu'on ajoute juste apres, sans la copier deux fois dans
        # l'image.
        xorriso -as mkisofs -iso-level 3 \
```

```
            -volid "$LABEL" \
            -full-iso9660-filenames \
            -eltorito-boot boot/grub/eltorito.img \
                -no-emul-boot -boot-load-size 4 -boot-info-table \
                --eltorito-catalog boot/grub/boot.cat \
                --grub2-boot-info \
                --grub2-mbr /usr/lib/grub/i386-pc/boot_hybrid.img \
            -eltorito-alt-boot \
                -e --interval:appended_partition_2:all:: \
                -no-emul-boot \
            -append_partition 2 0xef "$SB_ESP" \
            -appended_part_as_gpt \
            -o "$OUTPUT" \
            -graft-points "$ISOROOT" "/boot/grub/eltorito.img=$SB_BIOS" \
            && SECURE_BOOT_OK=1
    fi

    if [ "$SECURE_BOOT_OK" = "1" ]; then
        if [ "$SB_KERNEL_SIGNED" = "1" ]; then
            echo -e "  ${GREEN}✓${NC} ISO signee Secure Boot : shim -> grub -> noyau, chaine complete"
        else
            echo -e "  ${YELLOW}⚠${NC}  shim et grub sont signes, mais la signature du NOYAU n'a pas"
            echo -e "     pu etre confirmee ($ISOROOT/boot/vmlinuz). Si le boot s'arrete APRES"
            echo -e "     le menu GRUB, c'est la : installez sbsigntool pour le verifier, ou"
            echo -e "     utilisez un noyau linux-image-generic non recompile."
        fi
    else
        echo -e "  ${YELLOW}⚠${NC}  assemblage Secure Boot echoue - repli sur grub-mkrescue"
    fi
else
    echo -e "${YELLOW}[9/9] Secure Boot indisponible sur cet hote :${NC}"
    [ -z "$SB_SHIM" ] && echo -e "     shim absent  -> apt install ${CYAN}shim-signed${NC}"
    [ -z "$SB_GRUB" ] && echo -e "     grub signe absent -> apt install ${CYAN}grub-efi-amd64-signed${NC}"
    command -v mmd &>/dev/null || echo -e "     mtools absent -> apt install ${CYAN}mtools dosfstools${NC}"
fi

if [ "$SECURE_BOOT_OK" != "1" ]; then
# Repli : l'ancienne recette, mot pour mot. Elle produit une ISO qui demarre en
# BIOS et en UEFI sans Secure Boot - c'est ce qu'on avait avant, et c'est mieux
# que pas d'ISO du tout.
#
# Wrapper: inject -iso-level 3 (multi-extent, lifts the 4 GiB single-file cap)
# into grub-mkrescue's internal `xorriso -as mkisofs` call.
XORRISO_WRAP="$WORK/xorriso-iso-level3"
cat > "$XORRISO_WRAP" <<'EOF'
#!/bin/sh
if [ "$1" = "-as" ] && [ "$2" = "mkisofs" ]; then
    shift 2
    exec xorriso -as mkisofs -iso-level 3 "$@"
fi
exec xorriso "$@"
EOF
chmod +x "$XORRISO_WRAP"

    grub-mkrescue --xorriso="$XORRISO_WRAP" -o "$OUTPUT" "$ISOROOT" \
        --product-name "osmo-operator $VERSION" -- -volid "$LABEL"
    if command -v isohybrid &>/dev/null; then
        isohybrid --uefi "$OUTPUT"
    fi
    echo -e "  ${YELLOW}!${NC} ISO NON signee : elle ne demarrera pas si Secure Boot est actif."
    echo -e "    Desactivez-le dans le firmware, ou construisez sur un hote qui a"
    echo -e "    ${CYAN}shim-signed${NC} et ${CYAN}grub-efi-amd64-signed${NC}."
fi

if [ ! -f "$OUTPUT" ]; then
    echo -e "${RED}Creation de l'ISO echouee - rien n'a ete ecrit${NC}"
```

```
    exit 1
fi


# Fin de module : `. fichier` rend le statut de sa DERNIERE commande, et
# build-iso.sh tourne sous set -e. Un module qui finirait par un test faux
# ("[ ... ] && { ...; }") arreterait tout, sans un mot. Toujours 0 ici.
true
```

3.219 /opt/GSM/osmo-operator/iso_modules/92-rpi-image.sh

6190 octets, 93 lignes → 93 lignes

```
#!/bin/bash
# iso_modules/92-rpi-image.sh - --arm : l image SD du Raspberry Pi 4 (.img)
# Source par build-iso.sh, dans l ordre des numeros : meme shell, memes
# variables, memes fonctions. Ne s execute pas seul. `return` en tete de
# module = "rien a faire ici" (c est ainsi que --arm saute une etape).

# amd64 : l ISO est faite par 90-iso / 91-secure-boot.
if [ "${ISO_ARCH:-amd64}" != "arm64" ]; then return 0; fi

# ═══════════════════════════════════════════════════════════════════
#  Etape 9 : L IMAGE SD
# ═══════════════════════════════════════════════════════════════════
# Deux partitions, table MBR (le GPU du Pi ne lit que ca) :
#   1  FAT32  RPICFG        le contenu de /boot/firmware (noyau, dtb, start4.elf)
#   2  ext4   armbi_root    le rootfs, persistant (pas de squashfs, pas de live)
# Les memes etiquettes, tailles (FAT de 512 Mo) et table MBR que les images
# Armbian rpi4b : armbian-resize-filesystem etend armbi_root au premier boot.
# RIEN N EST MONTE : mke2fs -d peuple l ext4 depuis le repertoire (proprietaires,
# modes, liens, xattrs/capabilities compris), mcopy remplit la FAT, sfdisk ecrit
# la table, dd assemble. Pas de loop device, pas de privilege particulier au
# dela de ce que le chroot demandait deja, et rien qui reste monte si ca casse.
# L ext4 fait la taille du rootfs plus 15 % et 768 Mo de marge ; au premier
# demarrage osmo-grow-root l etend a la carte.
echo -e "${GREEN}[9/9] Image SD Raspberry Pi 4 (${ISO_ARCH})...${NC}"

# ── Ce qui ne doit pas partir ──────────────────────────────────────
# L emulateur copie pour le chroot est un binaire x86 : inutile sur le Pi.
rm -f "$ROOTFS/usr/bin/qemu-aarch64-static"
# Le cache apt lie pendant la construction est deja demonte (86-finitions) ;
# ce qui reste dans var/cache/apt est du rootfs lui-meme.
rm -rf "$ROOTFS/var/lib/apt/lists"/* "$ROOTFS/var/cache/apt/archives"/*.deb 2>/dev/null || true

# ── La partition firmware : le contenu de /boot/firmware, mis de cote ────────
# Dans l ext4, /boot/firmware doit etre un repertoire VIDE (le point de montage
# de la FAT) : on deplace son contenu et on le remet apres l image.
BOOTSRC="$WORK/bootfs"
rm -rf "$BOOTSRC"; mv "$ROOTFS/boot/firmware" "$BOOTSRC"; install -d "$ROOTFS/boot/firmware"
for _f in vmlinuz initrd.img start4.elf fixup4.dat bcm2711-rpi-4-b.dtb config.txt cmdline.txt; do
    [ -f "$BOOTSRC/$_f" ] || { echo -e "${RED}/boot/firmware/$_f absent - image impossible${NC}" >&2; exit 1; }
done

# ── Tailles ────────────────────────────────────────────────────────
BOOT_MB="${OSMO_RPI_BOOT_MB:-512}"
_root_kb=$(du -sxk "$ROOTFS" | cut -f1)
ROOT_MB=$(( _root_kb / 1024 * 115 / 100 + 768 ))
[ -n "${OSMO_RPI_ROOT_MB:-}" ] && [ "$OSMO_RPI_ROOT_MB" -gt "$ROOT_MB" ] && ROOT_MB="$OSMO_RPI_ROOT_MB"
echo -e "  rootfs $(( _root_kb / 1024 )) Mo -> ext4 de ${ROOT_MB} Mo, FAT de ${BOOT_MB} Mo"

# ── Partition 1 : FAT32 ────────────────────────────────────────────
BOOT_IMG="$WORK/boot.img"; rm -f "$BOOT_IMG"
mkfs.vfat -F 32 -n RPICFG -C "$BOOT_IMG" $(( BOOT_MB * 1024 )) >/dev/null
```

```
# mcopy -s : recursif (overlays/). Le "::" est la racine de la FAT.
mcopy -s -i "$BOOT_IMG" "$BOOTSRC"/* :: \
    || { echo -e "${RED}mcopy vers la partition firmware a echoue${NC}" >&2; exit 1; }
echo -e "  ${GREEN}✓${NC} RPICFG : $(mdir -i "$BOOT_IMG" :: | tail -1 | sed 's/^ *//')"

# ── Partition 2 : ext4, peuplee sans montage ──────────────────────────
ROOT_IMG="$WORK/root.img"; rm -f "$ROOT_IMG"
truncate -s "${ROOT_MB}M" "$ROOT_IMG"
# -d : le contenu ; -m 1 : 1 % reserve root ; lazy_*=0 : les tables d inodes
# sont ecrites maintenant, pas au premier montage sur une carte SD lente.
mke2fs -q -F -t ext4 -L armbi_root -m 1 -E lazy_itable_init=0,lazy_journal_init=0 \
    -d "$ROOTFS" "$ROOT_IMG" \
    || { echo -e "${RED}mke2fs -d a echoue : rootfs trop gros pour ${ROOT_MB} Mo ? (OSMO_RPI_ROOT_MB=... pour
    ↪  forcer)${NC}" >&2; exit 1; }
echo -e "  ${GREEN}✓${NC} armbi_root : ext4 ${ROOT_MB} Mo peuplee ($(e2fsck -n "$ROOT_IMG" 2>/dev/null | tail -1
↪  | sed 's/^[^:]*: //' || true))"

# Le contenu de /boot/firmware retourne dans le rootfs (une passe suivante, ou
# un rootfs conserve, doit le retrouver).
rm -rf "$ROOTFS/boot/firmware"; mv "$BOOTSRC" "$ROOTFS/boot/firmware"

# ── Assemblage : table MBR + les deux partitions ──────────────────────
rm -f "$OUTPUT"
_boot_start=2048
_boot_size=$(( BOOT_MB * 2048 ))
_root_start=$(( _boot_start + _boot_size ))
_root_size=$(( ROOT_MB * 2048 ))
truncate -s $(( (_root_start + _root_size + 2048) * 512 )) "$OUTPUT"
sfdisk --quiet "$OUTPUT" <<SFD
label: dos
unit: sectors
start=${_boot_start}, size=${_boot_size}, type=c, bootable
start=${_root_start}, size=${_root_size}, type=83
SFD
dd if="$BOOT_IMG" of="$OUTPUT" bs=512 seek="$_boot_start" conv=notrunc,sparse status=none
dd if="$ROOT_IMG" of="$OUTPUT" bs=512 seek="$_root_start" conv=notrunc,sparse status=none
rm -f "$BOOT_IMG" "$ROOT_IMG"
sfdisk --dump "$OUTPUT" | sed 's/^/    /' | grep -E "^ +$OUTPUT" || true
echo -e "  ${GREEN}✓${NC} image assemblee : $(du -h "$OUTPUT" | cut -f1) occupes, $(du -h --apparent-size
↪  "$OUTPUT" | cut -f1) apparents"

# Fin de module : `. fichier` rend le statut de sa DERNIERE commande, et
# build-iso.sh tourne sous set -e. Un module qui finirait par un test faux
# ("[ ... ] && { ...; }") arreterait tout, sans un mot. Toujours 0 ici.
true
```

3.220 /opt/GSM/osmo-operator/iso_modules/99-fin.sh

1806 octets, 34 lignes → 34 lignes

```
#!/bin/bash
# iso_modules/99-fin.sh - rootfs transmis a la passe suivante, resume
# Source par build-iso.sh, dans l ordre des numeros : meme shell, memes
# variables, memes fonctions. Ne s execute pas seul. `return` en tete de
# module = "rien a faire ici" (c est ainsi que --arm saute une etape).

# ── Le rootfs survit a cette passe si la parente le demande (--all) ──────
# Deplace, pas copie : meme systeme de fichiers, instantane. cleanup() efface
# ensuite $WORK sans lui.
if [ -n "${OSMO_ISO_ROOTFS_KEEP:-}" ]; then
    umount "$ROOTFS/var/cache/apt/archives" 2>/dev/null || true
    umount "$ROOTFS"/{dev/pts,proc,sys,dev} 2>/dev/null || true
    rm -rf "$OSMO_ISO_ROOTFS_KEEP"
    mkdir -p "$(dirname "$OSMO_ISO_ROOTFS_KEEP")"
```

```
    mv "$ROOTFS" "$OSMO_ISO_ROOTFS_KEEP"
    echo -e "  ${GREEN}✓${NC} rootfs conserve pour la passe suivante : ${CYAN}${OSMO_ISO_ROOTFS_KEEP}${NC}"
fi

echo ""
if [ "${ISO_ARCH:-amd64}" = "arm64" ]; then
    echo -e "${GREEN}${BOLD}═══ Image SD Raspberry Pi 4 (Armbian ${ISO_UBUNTU}) prete : ${OUTPUT} ($(du -sh
     ↪  "$OUTPUT"|cut -f1)) ═══${NC}"
    echo -e "  Chemin absolu : $(readlink -f "$OUTPUT")"
    echo -e "  Gravure : ${CYAN}dd if=$(basename "$OUTPUT") of=/dev/sdX bs=4M status=progress conv=fsync${NC}"
    echo -e "  Au premier demarrage armbian-resize-filesystem etend la racine a la carte ; console HDMI/serie et
     ↪  ssh, root:osmo."
else
echo -e "${GREEN}${BOLD}═══ ISO prete : ${OUTPUT} ($(du -sh "$OUTPUT"|cut -f1)) ═══${NC}"
echo -e "  Chemin absolu : $(readlink -f "$OUTPUT")"
fi


# Fin de module : `. fichier` rend le statut de sa DERNIERE commande, et
# build-iso.sh tourne sous set -e. Un module qui finirait par un test faux
# ("[ ... ] && { ...; }") arreterait tout, sans un mot. Toujours 0 ici.
true
```

3.221 /opt/GSM/osmo-operator/launch/osmo-launch.sh

12872 octets, 311 lignes → 311 lignes

```
#!/bin/bash
# ═══════════════════════════════════════════════════════════════════════════
# /opt/osmo-launch.sh - Orchestrateur osmo-operator (lab + web dashboard)
#
# Usage :
#   sudo /opt/osmo-launch.sh           # lance tout (interactif)
#   sudo /opt/osmo-launch.sh --auto    # lance avec les defauts (2 ops, 8 MS)
#   sudo /opt/osmo-launch.sh stop      # arrete tout
#   sudo /opt/osmo-launch.sh status    # etat des services
#   sudo /opt/osmo-launch.sh web-only  # lance uniquement le dashboard
# ═══════════════════════════════════════════════════════════════════════════
set -euo pipefail

LAB_DIR="/opt/GSM/osmo-operator"
WEB_DIR="/opt/GSM/osmo-egprs-web"
WEB_SERVICE="osmo-egprs-web"
LOG_DIR="/var/log/osmocom"

RED='\033[0;31m'; GREEN='\033[0;32m'; CYAN='\033[0;36m'
YELLOW='\033[1;33m'; NC='\033[0m'; BOLD='\033[1m'

[ "$(id -u)" -ne 0 ] && { echo -e "${RED}Root requis.${NC}"; exit 1; }

mkdir -p "$LOG_DIR"

# ═══════════════════════════════════════════════════════════════════════════
banner() {
    echo -e "${CYAN}${BOLD}"
    echo "╔═══════════════════════════════════════════════════════════════╗"
    echo "║  osmo-operator - GSM/EGPRS Multi-PLMN + Web Dashboard         ║"
    echo "╚═══════════════════════════════════════════════════════════════╝"
    echo -e "${NC}"
}

# ═══════════════════════════════════════════════════════════════════════════
wait_docker() {
    echo -ne "  Docker daemon"
```

```
    local i=0
    while ! docker info &>/dev/null; do
        sleep 1; echo -n "."; i=$((i+1))
        [ $i -ge 30 ] && { echo -e " ${RED}TIMEOUT${NC}"; return 1; }
    done
    echo -e " ${GREEN}OK${NC}"
}

wait_operators() {
    local timeout="${1:-120}"
    echo -ne "  Containers operateurs"
    local i=0
    while [ "$(docker ps --filter 'name=osmo-operator-' --format '{{.Names}}' 2>/dev/null | wc -l)" -eq 0 ]; do
        sleep 2; echo -n "."; i=$((i+2))
        [ $i -ge "$timeout" ] && { echo -e " ${RED}TIMEOUT${NC}"; return 1; }
    done
    local n=$(docker ps --filter 'name=osmo-operator-' --format '{{.Names}}' | wc -l)
    echo -e " ${GREEN}${n} operateur(s) detectes${NC}"
}

# ═══════════════════════════════════════════════════════════════
start_web() {
    echo -e "${GREEN}[web] Demarrage dashboard...${NC}"

    if [ -f "$WEB_DIR/server.js" ] && command -v node &>/dev/null; then
        # Mode natif (Node.js local)
        systemctl restart "$WEB_SERVICE" 2>/dev/null && {
            echo -e "  ${GREEN}✓${NC} systemd: $WEB_SERVICE actif (port 8080)"
            return 0
        }
        # Fallback si le service systemd n'existe pas
        echo -e "  ${YELLOW}Service systemd absent, lancement direct...${NC}"
        cd "$WEB_DIR"
        node server.js --verbose >> "$LOG_DIR/web-dashboard.log" 2>&1 &
        echo $! > /var/run/osmo-egprs-web.pid
        echo -e "  ${GREEN}✓${NC} Dashboard lance (PID $(cat /var/run/osmo-egprs-web.pid))"

    elif [ -f "$WEB_DIR/Dockerfile" ] || [ -f "$WEB_DIR/start-web.sh" ]; then
        # Mode Docker
        if [ -f "$WEB_DIR/start-web.sh" ]; then
            bash "$WEB_DIR/start-web.sh"
        else
            docker rm -f osmo-egprs-web 2>/dev/null || true
            docker build -t osmo-egprs-web "$WEB_DIR"
            docker run -d \
                --name osmo-egprs-web \
                --network host \
                --cap-add=NET_RAW --cap-add=NET_ADMIN \
                -v /var/run/docker.sock:/var/run/docker.sock:ro \
                -e CONTAINER_PREFIX=osmo-operator- \
                --restart unless-stopped \
                osmo-egprs-web
        fi
        echo -e "  ${GREEN}✓${NC} Dashboard Docker lance (port 80)"
    else
        echo -e "  ${RED}✗${NC} Dashboard introuvable dans $WEB_DIR"
        return 1
    fi
}

stop_web() {
    echo -e "${YELLOW}[web] Arret dashboard...${NC}"
    systemctl stop "$WEB_SERVICE" 2>/dev/null || true
    [ -f /var/run/osmo-egprs-web.pid ] && {
        kill "$(cat /var/run/osmo-egprs-web.pid)" 2>/dev/null || true
        rm -f /var/run/osmo-egprs-web.pid
```

```
    }
    docker rm -f osmo-egprs-web 2>/dev/null || true
    echo -e "  ${GREEN}✓${NC} Dashboard arrete"
}

# ═══════════════════════════════════════════════════════════════════
start_lab() {
    echo -e "${GREEN}[lab] Demarrage osmo-operator...${NC}"

    if [ ! -f "$LAB_DIR/launch/start-docker.sh" ]; then
        echo -e "  ${RED}✗${NC} $LAB_DIR/launch/start-docker.sh introuvable"
        return 1
    fi

    cd "$LAB_DIR"
    bash ./launch/start-docker.sh
}

# ── LE NATIF AUSSI ─────────────────────────────────────────────────
# [2026-09-04] `osmo-launch stop` promet « arrete tout » et balayait bien les
# conteneurs - mais pas l operateur NATIF, qui vit dans osmo-banc.service
# (start-native.sh, session tmux « calypso »). On arretait donc la moitie du
# banc : le coeur natif (osmo-stp, hlr, msc, bsc, asterisk) restait debout, et
# c est LUI qui tient le 5060 et le 8080 que le lancement suivant reclame.
# Symetrique de launch.sh --stop, qui arrete desormais les deux cotes.
#
# D ABORD LE NATIF, ENSUITE LES CONTENEURS : l unite delegue a
# start-native.sh --stop, qui demonte radio et coeur proprement ; le faire
# apres un `docker rm -f` ne changerait rien pour lui, mais l ordre inverse est
# celui de l arret propre et c est celui qu on garde partout.
stop_natif() {
    local u="${OSMO_BANC_SERVICE:-osmo-banc.service}"
    command -v systemctl >/dev/null 2>&1 || return 0
    [ -d /run/systemd/system ] || return 0
    systemctl cat "$u" >/dev/null 2>&1 || return 0
    echo -e "${YELLOW}[natif] Arret ${u}...${NC}"
    timeout 200 systemctl stop "$u" >/dev/null 2>&1 || true
    # ExecStop ne tourne pas sur une unite « failed » : le coeur, lance par
    # run.sh dans SES PROPRES unites, resterait alors debout.
    systemctl stop osmo-egprs-web.service osmo-hlr.service >/dev/null 2>&1 || true
    systemctl reset-failed "$u" >/dev/null 2>&1 || true
    echo -e "  ${GREEN}✓${NC} Natif arrete"
}

stop_lab() {
    echo -e "${YELLOW}[lab] Arret osmo-operator...${NC}"
    if [ -f "$LAB_DIR/launch/start-docker.sh" ]; then
        cd "$LAB_DIR"
        bash ./launch/start-docker.sh stop
    fi
    # Nettoyage complet si start-docker.sh stop ne suffit pas
    docker ps -a --filter "name=osmo-" --format "{{.Names}}" | xargs -r docker rm -f 2>/dev/null || true
    echo -e "  ${GREEN}✓${NC} Lab arrete"
}

# ═══════════════════════════════════════════════════════════════════
show_status() {
    echo -e "${CYAN}${BOLD}── Etat des services ──${NC}"
    echo ""

    # Docker
    if docker info &>/dev/null; then
        echo -e "  Docker       : ${GREEN}●${NC} actif"
    else
        echo -e "  Docker       : ${RED}●${NC} arrete"
    fi
```

```
    # Containers
    local ops=$(docker ps --filter 'name=osmo-operator-' --format '{{.Names}}' 2>/dev/null | sort)
    local n_ops=$(echo "$ops" | grep -c . 2>/dev/null || echo 0)
    if [ "$n_ops" -gt 0 ]; then
        echo -e "  Operateurs   : ${GREEN}●${NC} ${n_ops} actif(s)"
        echo "$ops" | while read -r c; do
            echo -e "                 ${CYAN}└─${NC} $c"
        done
    else
        echo -e "  Operateurs   : ${RED}●${NC} aucun"
    fi

    # Inter-STP
    if docker ps --format '{{.Names}}' 2>/dev/null | grep -q osmo-inter-stp; then
        echo -e "  Inter-STP    : ${GREEN}●${NC} actif"
    else
        echo -e "  Inter-STP    : ${RED}●${NC} arrete"
    fi

    # Web dashboard
    if systemctl is-active --quiet "$WEB_SERVICE" 2>/dev/null; then
        echo -e "  Dashboard    : ${GREEN}●${NC} systemd (port 8080)"
    elif [ -f /var/run/osmo-egprs-web.pid ] && kill -0 "$(cat /var/run/osmo-egprs-web.pid 2>/dev/null)"
     ↪  2>/dev/null; then
        echo -e "  Dashboard    : ${GREEN}●${NC} PID $(cat /var/run/osmo-egprs-web.pid) (port 8080)"
    elif docker ps --format '{{.Names}}' 2>/dev/null | grep -q osmo-egprs-web; then
        echo -e "  Dashboard    : ${GREEN}●${NC} Docker (port 80)"
    else
        echo -e "  Dashboard    : ${RED}●${NC} arrete"
    fi

    # IPs
    echo ""
    local host_ip=$(ip route get 1.1.1.1 2>/dev/null | awk '{for(i=1;i<=NF;i++) if($i=="src") print $(i+1)}' |
     ↪  head -1)
    [ -n "$host_ip" ] && echo -e "  Host IP      : ${CYAN}${host_ip}${NC}"
    echo -e "  Dashboard    : ${CYAN}http://${host_ip:-localhost}:8080${NC}"
    echo ""
}

# ═══════════════════════════════════════════════════════════════════
do_start() {
    banner

    # 1. Docker
    echo -e "${GREEN}[1/3] Verification Docker...${NC}"
    systemctl start docker 2>/dev/null || true
    wait_docker || { echo -e "${RED}Docker necessaire.${NC}"; exit 1; }

    # 2. Lab
    echo ""
    echo -e "${GREEN}[2/3] Lab multi-operateurs...${NC}"
    start_lab

    # 3. Web (attendre que les containers soient up)
    echo ""
    echo -e "${GREEN}[3/3] Dashboard web...${NC}"
    wait_operators 180 || true
    start_web || true

    # Resume
    echo ""
    show_status
    echo -e "${GREEN}${BOLD}Pret !${NC}"
}
```

```
do_auto() {
    banner
    echo -e "${YELLOW}Mode automatique (defauts)${NC}"
    echo ""

    systemctl start docker 2>/dev/null || true
    wait_docker || exit 1

    # Lancer start-docker.sh en mode non-interactif avec des defauts
    # On pipe les reponses au script interactif
    cd "$LAB_DIR"
    {
        echo "2"       # bridge mode
        echo "2"       # 2 operateurs
        echo "o"       # valeurs par defaut
        echo "8"       # 8 MS
        echo "N"       # pas de WAN
        echo "1"       # faketrx
        echo "0"       # A5/0
    } | bash ./launch/start-docker.sh

    wait_operators 180 || true
    start_web || true

    echo ""
    show_status
}

# ═══════════════════════════════════════════════════════════════════
# Main
# ═══════════════════════════════════════════════════════════════════
CMD="${1:-}"

case "$CMD" in
    stop)
        banner
        stop_web
        stop_natif
        stop_lab
        echo -e "\n${GREEN}Tout arrete.${NC}"
        ;;
    status)
        banner
        show_status
        ;;
    web-only)
        banner
        systemctl start docker 2>/dev/null || true
        wait_docker || exit 1
        start_web
        ;;
    --auto)
        do_auto
        ;;
    restart)
        banner
        stop_web
        stop_natif
        stop_lab
        sleep 3
        do_start
        ;;
    help|-h|--help)
        echo "Usage: $0 [commande]"
        echo ""
```

```
        echo "  (rien)    Lancement interactif (lab + web)"
        echo "  --auto    Lancement automatique (2 ops, 8 MS, defauts)"
        echo "  stop      Arrete tout (natif + lab + web + containers)"
        echo "  restart   Redemarre tout"
        echo "  status    Etat des services"
        echo "  web-only  Lance uniquement le dashboard web"
        echo "  help      Ce message"
        ;;
    *)
        do_start
        ;;
esac
```

3.222 /opt/GSM/osmo-operator/launch/start-docker.sh

168475 octets, 2960 lignes → 2960 lignes

```
#!/bin/bash
# start-docker.sh - Lance la stack Osmocom GSM multi-operateurs
#
# Modes : net-host (1 operateur, SDR physique) | bridge (N operateurs SS7 inter-op)
set -eu
# Racine du depot : ce script vit dans launch/ depuis le 2026-10-18 (ex start.sh a la
# racine). Les anciens chemins en « dirname $0 » visaient en fait le depot.
# OSMO_REPO_DIR d'abord : build-iso.sh (iso_modules/30-image-configs.sh) source une
# COPIE de ce fichier dans $WORK ; BASH_SOURCE donnerait alors /var/tmp.
START_REPO="${OSMO_REPO_DIR:-$(cd "$(dirname "$(readlink -f "${BASH_SOURCE[0]}")")/.." && pwd)}"

# ── Ce script est le lanceur de l'HOTE. Refus net s'il tourne DANS le conteneur ──
# [2026-08-12] start-docker.sh pilote docker (build, reseaux, `docker run`, volumes) ;
# start-native.sh, lui, prepare l'environnement Calypso et lance run.sh A
# L'INTERIEUR. Les confondre ne donne pas une erreur lisible : `docker` n'existe
# pas dans l'image, donc on part sur une cascade de "command not found" apres
# avoir deja cree des fichiers et touche des configs. Mieux vaut s'arreter avant
# d'avoir rien fait, et nommer le script a utiliser.
#
# Discriminant verifie dans LES DEUX SENS le 12/08 : `/.dockerenv` et
# `/etc/docker-entrypoint-cmd` (pose par scripts/entrypoint.sh) sont presents
# dans le conteneur et absents sur l'hote. On exige les DEUX : `/.dockerenv`
# seul serait vrai dans n'importe quel conteneur, y compris un ou ce script
# aurait legitimement sa place.
if [ -f /.dockerenv ] && [ -f /etc/docker-entrypoint-cmd ]; then
    printf '\033[1;31mVous etes dans le docker ! Utilisez start-native.sh\033[0m\n' >&2
    printf '\n  \033[0;36m/opt/GSM/osmo-operator/start-native.sh\033[0m   (ou ./start-native.sh depuis le
     ↪  depot)\n' >&2
    printf '\n  start-docker.sh est le lanceur de l'"'"'HOTE : il construit l'"'"'image, cree les\n' >&2
    printf '  reseaux et fait le "docker run". Rien de tout ca n'"'"'a de sens ici.\n' >&2
    exit 1
fi

DEBUG=
if [[ -n "$DEBUG" ]]; then
    set -x
    PS4='[DEBUG] + ${BASH_SOURCE}:${LINENO}: '
    echo "=== MODE DEBUG ACTIVE ==="
fi

IMAGE_BASE="osmocom-nitb"
# [2026-08-12] Surchargeable depuis l'environnement - c'est ce que fait
# start-nitb.sh, qui relance exactement ce script avec IMAGE_RUN=osmocom-nitb.
# Idiome `:=` du projet : une valeur posee explicitement en amont gagne, le
# defaut ne s'applique qu'a vide. NE PAS remettre en dur : start-nitb.sh
# deviendrait silencieusement un alias de start-docker.sh, ce qui est indiscernable
# d'un lanceur qui marche.
```

```
IMAGE_RUN="${IMAGE_RUN:-osmocom-run}"

INTER_NET="gsm-inter"
INTER_NET_SUBNET="172.20.0.0/24"
INTER_NET_GATEWAY="172.20.0.1"

# Quel noeud du WAN heberge le hub SS7. UN SEUL doit le faire : deux hubs, c'est
# deux reseaux SS7 separes qui s'ignorent, et rien ne le signale.
WAN_HUB_NODE="${WAN_HUB_NODE:-1}"
# --hub-ip : l'inter-STP designe par son ADRESSE, pas par un numero de noeud.
# --hub-node oblige a le faire figurer dans la table WAN ; sur un banc ou le hub
# est une machine a part - une VM dediee, sans operateur - cela forcait a
# inventer un noeud pour lui, qui apparaissait ensuite dans les indicatifs et le
# routage voix/SMS comme s'il portait des abonnes. Avec --hub-ip la table ne
# contient que de vrais noeuds.
WAN_HUB_IP="${WAN_HUB_IP:-}"
# --node-per-op : chaque conteneur est un NOEUD, pas un operateur de la machine.
# A n'utiliser que quand les conteneurs ont une adresse joignable de l'exterieur
# (backbone route vers le LAN) : sinon les autres noeuds ne peuvent pas les
# appeler, et un noeud injoignable vaut moins qu'un operateur bien range.
WAN_NODE_PER_OP="${WAN_NODE_PER_OP:-0}"
# --operator IP:PREFIXE : declarer un operateur du WAN, ou qu'il soit.
# La table WAN se saisissait jusqu'ici par --wan-nodes "id:ip:ind", une forme
# exacte mais qui suppose qu'on connaisse deja les numeros de noeud. Declarer
# "une machine a cette adresse, joignable par ce prefixe" est plus proche de ce
# qu'on a sous les yeux - une VM, un conteneur - et les numeros suivent l'ordre
# de declaration.
OPERATOR_DECLS=()
# [2026-08-31] On HERITE de l environnement au lieu de l ecraser.
# Cette ligne valait OPERATOR_COUNT_HINT="" sec : un appelant qui posait
# la variable (env OPERATOR_COUNT_HINT=2 ./launch/start-docker.sh ...) la voyait effacee
# ici, bien avant sa lecture l.1421 - et start-docker.sh reposait la question
# « Nombre d'operateurs (1-36) » qu on croyait justement avoir evitee.
# Silencieux de bout en bout : la variable etait bien passee, bien nommee,
# et simplement remise a zero en chemin.
# --operators N reste la voie explicite (l.2552) et prime, puisqu il est
# analyse plus tard.
OPERATOR_COUNT_HINT="${OPERATOR_COUNT_HINT:-}"
# --host : l'adresse de CETTE machine, telle que les VM doivent la joindre.
# Elle est deduite de l'interface par defaut ; --host la fixe quand la deduction
# se trompe - plusieurs cartes, un wifi et un ethernet actifs en meme temps, ou
# une adresse qui vient de changer. Elle sert de recours, pas de reglage
# quotidien : c'est elle que les VM inscrivent dans leur route vers le backbone.
HOST_LAN_IP=""
IMAGE_STP="${IMAGE_STP:-osmocom-stp}"
INTER_STP_CONTAINER="osmo-inter-stp"
INTER_STP_IP="172.20.0.10"

RED='\033[0;31m'; GREEN='\033[0;32m'; YELLOW='\033[1;33m'
CYAN='\033[0;36m'; NC='\033[0m'; BOLD='\033[1m'

# ── SMS Routing ──────────────────────────────────────────────────────────
SMS_ROUTING_SCRIPT="$START_REPO/scripts/sms-routing-setup.sh"
if [ -f "$SMS_ROUTING_SCRIPT" ]; then
    source "$SMS_ROUTING_SCRIPT"
fi
SMS_ROUTING_DIR=""

# ═══════════════════════════════════════════════════════════════════════
# WAN Interop
# ═══════════════════════════════════════════════════════════════════════
# Deux WAN coexistent ici, et ils ne font PAS la meme chose :
#
#   WAN_ENABLED (legacy, --pas d'option, question whiptail) : DEUX serveurs,
#     prefixe unique 66, network/setup-wan-interop.sh. Conserve tel quel.
#
```

```
#   WAN_MESH (option --wan) : N noeuds (1 a 9), UN INDICATIF PAR NOEUD,
#     network/setup-wan-mesh.sh. C'est le seul qui tienne au-dela de deux
#     serveurs - le legacy reecrit son bloc de conf a chaque appel, donc
#     l'enchainer sur trois pairs ne laisse que le dernier.
#
# Les deux sont EXCLUSIFS et aucun n'est actif par defaut.
WAN_MESH=0
# --virtualbox : les pairs du WAN sont des VM VirtualBox sur CETTE machine, et
# cette machine est elle-meme un noeud. Voir network/setup-vbox-interco.sh.
VBOX_INTERCO=0
VBOX_NODES=""
VBOX_HOST_NODE=1
BUILD_STP=0
# shellcheck source=network/wan-nodes.sh
. "$START_REPO/network/wan-nodes.sh"

WAN_ENABLED="false"
WAN_LOCAL_IP=""
WAN_REMOTE_IP=""
WAN_N_REMOTE=""
WAN_PREFIX="66"
WAN_SIP_BASE=5080
WAN_RTP_BASE=20000
WAN_RTP_PER_OP=500
PHY_MODE="faketrx"   # faketrx | virtphy
# Choix passes au start-native.sh DANS le conteneur via le handoff (NO_MENU=1),
# pour eviter les menus whiptail laggy en docker exec -ti. Fixes sur l'HOTE.
# Retenu AVANT le defaut : "l'appelant a-t-il impose le mode ?". C'est ce qui
# permet a --wan de basculer en hybride sans ecraser un HANDOFF_MODE=... explicite.
HANDOFF_MODE_FROM_ENV="${HANDOFF_MODE:+1}"
HANDOFF_MODE="${HANDOFF_MODE:-qemu}"              # qemu | faketrx-qemu (combine)
HANDOFF_QEMU_CHOICE="${HANDOFF_QEMU_CHOICE:-full-grgsm}"

# ── Helpers ──────────────────────────────────────────────────────────────
# ── Les deux plans d'adressage d'un operateur ────────────────────────────
# backbone  172.20.0.<10+op>  le segment que TOUS partagent avec l'inter-STP
#                             (.10). C'est par la que passe le SS7, et c'est ce
#                             reseau que network/setup-docker-lan-route.sh rend
#                             joignable depuis le LAN des VM.
# prive     192.168.<op+1>.x  un segment par operateur : son BTS, son PCU, son
#                             GGSN. Rien n'y entre de l'exterieur.
#
# POURQUOI <op+1> ET PAS <op>
# Le LAN du banc est 192.168.1.0/24 - celui des VM et du hub SS7 en .49. Un
# operateur 1 en 192.168.1.0/24 entrerait en collision avec lui : deux routes
# pour le meme reseau sur l'hote, et un conteneur qui ne peut plus joindre le
# hub. Le decalage d'un rang laisse le .1 au LAN et commence les operateurs a
# 192.168.2.0/24.
# ── LE RANG DU PREMIER CONTENEUR ─────────────────────────────────────────
# [2026-08-31] start-docker.sh numerotait SES operateurs 1..N, en dur. C est juste
# quand la machine ne porte que des conteneurs ; ca ne l est plus des qu un
# operateur NATIF tourne a cote, car il occupe deja un rang - et deux
# equipements au meme point code, ce n est pas un conflit de nom, c est du
# routage faux (checks/wan_ss7_check.sh).
# OP_ID_BASE=2 fait donc commencer les conteneurs a l operateur 2, laissant le 1
# au natif. TOUT en decoule sans autre reglage, puisque tout derive de l index :
# nom du conteneur (osmo-operator-N), backbone (172.20.0.<10+N>), point code
# (1.N.2), RCTX, segment prive (192.168.<N+1>.0/24) et ports publies.
OP_ID_BASE="${OP_ID_BASE:-1}"
op_backbone_ip()  { echo "172.20.0.$((10 + $1))"; }
# ── L'INDEX DU SEGMENT PRIVE : LE NOEUD OU L'OPERATEUR, SELON L'HOTE ─────
# 192.168.<index+1>.x, et tout le desaccord tenait a « index ».
#
#   DOCKER   un hote porte N conteneurs operateurs, chacun dans son netns. Ce
#            qui les distingue est le NUMERO D'OPERATEUR ; le noeud, lui, est
#            commun a tous les conteneurs de la machine. index = operateur.
```

```
#   VM/NATIF la machine EST le noeud et ne porte qu'un operateur. Ce qui la
#            distingue de ses voisines est le NUMERO DE NOEUD. index = noeud.
#
# Sans cette distinction, qosmo-grgsm/run_modules/08-gabarits.sh appelait
# op_private_ip($OPERATOR_ID) et ecrivait 192.168.2.10 dans osmo-ggsn.cfg sur
# TOUTES les VM - operateur 1 partout - pendant que le plan de l'ISO reservait
# 192.168.<noeud+1>.x. Sur le noeud 1 les deux coincidaient et rien ne se
# voyait ; sur le noeud 2, le GGSN se liait a l'adresse du noeud 1.
# La bascule est ici, une fois, et les deux jumelles (start-docker.sh, lib/gabarits.sh)
# en heritent.
_osmo_priv_index() {
    local op="${1:-1}"
    # [2026-08-31] L ORCHESTRATEUR N EST PAS DANS LE CONTENEUR QU IL CREE.
    # La detection ci-dessous demande "suis-je DANS un conteneur ?" et c est la
    # bonne question pour du code qui tourne a l interieur. Mais start-docker.sh, lui,
    # provisionne les conteneurs DEPUIS L HOTE : il prenait donc la branche
    # VM/natif et rendait l index du NOEUD - 1 - pour tous les operateurs.
    # Les deux reseaux prives se retrouvaient sur 192.168.2.0/24, docker
    # refusait le second pour chevauchement, et la boucle s arretait net :
    #     ── Operateur 2 : ... Prive : 192.168.2.10
    #     ✗ Echec creation reseau gsm-net-op2 (192.168.2.0/24)
    # osmo-operator-2 n etait jamais cree, sans qu aucun message ne parle de
    # collision d adresses.
    # OSMO_PRIV_BY_OP=1 dit "je provisionne par operateur" ; start_bridge_mode
    # la pose. C est la seule chose que la detection ne pouvait pas deviner.
    [ "${OSMO_PRIV_BY_OP:-0}" = "1" ] && { printf '%s' "$op"; return; }
    # Meme detection que start-native.sh : le couple /.dockerenv +
    # /etc/docker-entrypoint-cmd identifie un conteneur DE CE DEPOT ; le cgroup
    # sert de repli pour un conteneur quelconque.
    if [ -f /.dockerenv ] || grep -qa 'docker\|containerd' /proc/1/cgroup 2>/dev/null; then
        printf '%s' "$op"; return
    fi
    local n="${OSMO_WAN_NODE:-${WAN_NODE_ID:-}}"
    [ -n "$n" ] || n="$(awk -F= '/^OSMO_WAN_NODE=/{gsub(/[ \r\t]/,"",$2);v=$2} END{print v}' \
                        "${ROLE_FILE:-/etc/osmo-role}" 2>/dev/null)"
    [ -n "$n" ] || n="$(sed -n 's/^PLAN_NODE=//p' "${OSMOCOM_CFG:-/etc/osmocom}/radio-plan.env" 2>/dev/null | tail
     ↪  -1)"
    case "$n" in [1-9]) ;; *) n=1 ;; esac
    printf '%s' "$n"
}
op_private_ip()   { echo "192.168.$(($(_osmo_priv_index "$1") + 1)).10"; }
op_private_gw()   { echo "192.168.$(($(_osmo_priv_index "$1") + 1)).1"; }
op_private_net()  { echo "192.168.$(($(_osmo_priv_index "$1") + 1)).0/24"; }
op_container()    { echo "osmo-operator-$1"; }
op_rctx_msc()     { echo $(( $1 * 100 + 10 )); }
op_rctx_stp()     { echo $(( $1 * 100 + 20 )); }
op_rctx_bsc()     { echo $(( $1 * 100 + 30 )); }
op_rctx_inter()   { echo $(( $1 * 100 + 50 )); }

# ── Le PLMN : le MCC est le PAYS, donc le NOEUD ; le MNC, l'operateur ──────────
# Un MCC fige a 001 pour tout le monde ne laissait que le MNC pour distinguer
# les reseaux, et toute valeur ecrite en dur ailleurs - les IMSI des MS
# l'etaient - retombait sur un PLMN plausible mais etranger : le mobile de
# l'operateur 2 presentait 001-01, son reseau diffusait 001-02, et le Location
# Updating restait sans reponse, pile entiere debout.
# Le noeud, c'est ce que --node passe a start-native.sh. Avec --node-per-op
# chaque conteneur est son propre noeud, donc son propre pays, et le MNC
# retombe a 01 ; sans lui, les conteneurs sont N operateurs d'un meme pays.
op_plmn_node()  { if [ "${WAN_NODE_PER_OP:-0}" = "1" ]; then echo $(( ${WAN_NODE_ID:-1} + $1 - 1 )); else echo
 ↪  "${WAN_NODE_ID:-1}"; fi; }
op_plmn_index() { if [ "${WAN_NODE_PER_OP:-0}" = "1" ]; then echo 1; else echo "$1"; fi; }
# printf '%03d' sur une valeur vide ou nulle rend "000", et osmo-msc REFUSE
# "network country code 000" : il abandonne la lecture a cette ligne et systemd
# le relance trois fois par seconde jusqu'a y renoncer. Le lanceur, lui,
# affichait une croix puis "Core Osmocom pret". Trois chiffres, une heure de
# recherche dans le SS7. Un numero de noeud a zero est un bug, pas un MCC : on
```

```
# le borne et on le dit, une fois.
_plmn_borne() {                 # $1 = valeur, $2 = plancher, $3 = quoi
    case "$1" in
        ''|*[!0-9]*) echo "  ATTENTION : $3 illisible ('$1') - ramene a $2" >&2; echo "$2" ;;
       *) if [ "$1" -lt "$2" ]; then echo "  ATTENTION : $3 vaut $1 - ramene a $2" >&2; echo "$2"; else echo "$1";
        ↪  fi ;;
    esac
}
op_mcc()        { printf '%03d' "$(_plmn_borne "$(op_plmn_node "$1")" 1 "numero de noeud (MCC)")"; }
op_mnc()        { printf '%02d' "$(_plmn_borne "$(op_plmn_index "$1")" 1 "rang d'operateur (MNC)")"; }

# Noms sans espace : la commande VTY "short name" d'osmo-msc prend un mot, et
# un nom coupe en deux y passe pour un argument surnumeraire.
# ── Les jeux de donnees du banc : data/*.txt ──────────────────────────────
# Noms d'operateur, pays, etat civil et adresses ne sont pas ecrits dans ce
# script : ils vivent dans data/, un par ligne, editables sans toucher au code.
# On les MELANGE au chargement - c'est la le tirage au hasard - puis on les
# fige pour toute la duree du lancement : un operateur qui changerait de nom ou
# de pays entre deux lignes du journal ne serait plus le meme reseau.
OSMO_DATA_DIR="${OSMO_DATA_DIR:-$START_REPO/data}"

# $1 = nom du tableau a remplir, $2 = fichier, le reste = valeurs de secours.
# Le depot peut etre incomplet (une ISO n'embarque pas toujours data/) : sans
# repli, un banc entier s'arreterait sur un fichier d'agrement absent.
_load_pool() {
    local -n __out=$1
    local __file=$2; shift 2
    __out=()
    if [ -r "$__file" ]; then
        if command -v shuf >/dev/null 2>&1; then
            mapfile -t __out < <(grep -vE '^[[:space:]]*(#|$)' "$__file" | shuf)
        else
            mapfile -t __out < <(grep -vE '^[[:space:]]*(#|$)' "$__file")
        fi
    fi
    [ "${#__out[@]}" -gt 0 ] || __out=("$@")
}
declare -a POOL_OP POOL_PAYS POOL_NOMS POOL_ADR
_load_pool POOL_OP   "$OSMO_DATA_DIR/op.txt"        OsmoOP1 OsmoOP2 OsmoOP3
_load_pool POOL_PAYS "$OSMO_DATA_DIR/pays.txt"      Syldavie Bordurie Poldevie
_load_pool POOL_NOMS "$OSMO_DATA_DIR/noms.txt"      "Jean Dupont" "Marie Martin"
_load_pool POOL_ADR  "$OSMO_DATA_DIR/adresses.txt"  "rue des Ondes Courtes, Fontaine-sur-Onde"

op_default_name() { echo "${POOL_OP[$(( ($1 - 1) % ${#POOL_OP[@]} ))]}"; }
op_country()      { local n; n=$(op_plmn_node "$1"); echo "${POOL_PAYS[$(( (n - 1) % ${#POOL_PAYS[@]} ))]}"; }

# WAN helpers
wan_sip_port()    { echo $(( WAN_SIP_BASE + ($1 - 1) * 2 )); }
wan_rtp_start()   { echo $(( WAN_RTP_BASE + ($1 - 1) * WAN_RTP_PER_OP )); }
wan_rtp_end()     { echo $(( WAN_RTP_BASE + $1 * WAN_RTP_PER_OP - 1 )); }

wan_sms_port()    { echo $(( 7890 + $1 - 1 )); }
# Un WAN est actif - legacy OU mesh. Sert partout ou la question est "faut-il
# ouvrir les ports WAN", independamment du mecanisme choisi.
wan_active()      { [ "$WAN_ENABLED" = "true" ] || [ "${WAN_MESH:-0}" = "1" ]; }

# Linphone helpers - port SIP/RTP expose sur le host
# ── PLUS AUCUN DECALAGE DE PORTS ──────────────────────────────────────────
# [2026-08-31] Il y avait ici une detection : "5060 est-il pris ? alors decale
# les publications de +100". Elle ne servait qu a une chose - eviter que le
# 5060 publie d un conteneur n entre en conflit avec l Asterisk du natif.
#
# On ne publie PLUS aucun port SIP/RTP pour les conteneurs : ils sont sur un
# bridge routable et se joignent en direct, a leur propre adresse, sur le port
# nominal (verifie : 172.20.0.12:5060 et 172.20.0.13:5060 repondent 401 depuis
# l hote, sans la moindre publication). Le conflit ayant disparu avec sa cause,
```

```
# le decalage n a plus d objet - et il tirait avec lui une sonde instantanee,
# donc une course : lancee juste apres l arret du banc, elle voyait 5060 libre,
# ne decalait rien, et le conteneur mourait quand le natif remontait.
#
# Les fonctions ci-dessous gardent leur forme (generate_configs.sh s en sert
# encore pour ecrire les configs) mais rendent desormais les valeurs NOMINALES.
# OSMO_PORT_OFFSET=N reste disponible pour qui voudrait retablir un decalage.
PORT_OFFSET="${OSMO_PORT_OFFSET:-0}"
# Le RTP suit la meme regle : plage nominale, 200 ports par operateur.
linphone_sip_port()  { echo $(( 5060  + ${PORT_OFFSET:-0}      + ($1 - 1) )); }
linphone_rtp_start() { echo $(( 30000 + ${PORT_OFFSET:-0} * 10 + ($1 - 1) * 200 )); }
linphone_rtp_end()   { echo $(( 30000 + ${PORT_OFFSET:-0} * 10 + $1 * 200 - 1 )); }

# Global - detecte avant la boucle operateurs
HOST_IP="127.0.0.1"
ALSA_OUTPUT="${ALSA_OUTPUT:-default}"
ALSA_INPUT="${ALSA_INPUT:-default}"
HOST_AUDIO_RELAY=""
WSLG_TCP_PORT="${WSLG_TCP_PORT:-4713}"

banner() {
    echo -e "${CYAN}"
    echo "╔═══════════════════════════════════════════════════════════╗"
    echo "║          Osmocom GSM/EGPRS Virtual Network                ║"
    echo "║         Multi-Operator SS7 + osmo-gapk Audio              ║"
    echo "╚═══════════════════════════════════════════════════════════╝"
    echo -e "${NC}"
}

# ── Whiptail helpers ──────────────────────────────────────────────
WT_BACKTITLE="Osmocom GSM/EGPRS Virtual Network"
WT_WIDTH=72

check_whiptail() {
    if ! command -v whiptail >/dev/null 2>&1; then
        echo -e "${RED}whiptail introuvable.${NC} Installez-le :" >&2
        echo -e "  ${CYAN}apt-get install -y whiptail${NC}  (Debian/Ubuntu)" >&2
        echo -e "  ${CYAN}dnf install -y newt${NC}          (Fedora/RHEL)" >&2
        exit 1
    fi
}

# ── MODE NON INTERACTIF : LE QUESTIONNAIRE REPOND TOUT SEUL ──────
# [2026-08-31] Chaque wt_* ouvre une boite whiptail et ATTEND. C est parfait au
# terminal, et fatal partout ailleurs : lance par une icone du bureau, par
# start-multi.sh ou par un script, start-docker.sh s arretait sur « Nombre
# d'operateurs (1-36) » - un dialogue que personne ne voit et que personne ne
# valide. Le lancement paraissait bloque ou mort, alors qu il attendait une
# reponse.
#
# On repond donc AUTOMATIQUEMENT, avec la valeur que la boite proposait deja :
#   wt_input -> son defaut ($3), celui qui est pre-rempli dans le champ ;
#   wt_menu  -> la PREMIERE entree, celle sur laquelle le curseur se pose ;
#   wt_yesno -> non, parce que les boites sont en --defaultno ;
#   wt_msg   -> ecrit sur stderr au lieu d attendre un <Ok>.
# Aucune de ces reponses n est inventee : c est exactement ce qu on obtient en
# appuyant sur Entree.
#
# DEUX DECLENCHEURS. OSMO_NONINTERACTIVE=1 le force (start-multi.sh le pose).
# Et l absence de TTY sur l entree standard le declenche seule : c est la
# signature d un appel par icone, par cron ou par pipe - la ou aucun dialogue
# ne peut aboutir. Au terminal, rien ne change : les questions sont posees.
_wt_auto() { [ "${OSMO_NONINTERACTIVE:-0}" = "1" ] || [ ! -t 0 ]; }

wt_input() {
    if _wt_auto; then
```

```
        echo "[auto] $1 : ${3}" >&2
        printf '%s' "$3"; return 0
    fi
    whiptail --backtitle "$WT_BACKTITLE" --title "$1" \
        --inputbox "$2" 10 "$WT_WIDTH" "$3" 3>&1 1>&2 2>&3
}

wt_yesno() {
    if _wt_auto; then echo "[auto] $1 : non" >&2; return 1; fi
    whiptail --backtitle "$WT_BACKTITLE" --title "$1" \
        --defaultno --yesno "$2" 11 "$WT_WIDTH" 3>&1 1>&2 2>&3
}

wt_menu() {
    local title="$1" prompt="$2"; shift 2
    local nitems=$(( $# / 2 ))
    if _wt_auto; then
        echo "[auto] $title : ${1}" >&2
        printf '%s' "$1"; return 0
    fi
    whiptail --backtitle "$WT_BACKTITLE" --title "$title" \
        --menu "$prompt" 20 "$WT_WIDTH" "$nitems" "$@" 3>&1 1>&2 2>&3
}

wt_msg() {
    if _wt_auto; then echo "$1" >&2; return 0; fi
    whiptail --backtitle "$WT_BACKTITLE" --msgbox "$1" 10 "$WT_WIDTH"
}

# ── Loopback audio cote session utilisateur ──────────────────────
# ── Session audio de l'hote : QUI, et OU ─────────────────────────
# [2026-08-12] Ces deux fonctions codaient "nirvana" en dur et cherchaient
# /home/<user>/osmo-operator/loopback.sh - un chemin QUI N'EXISTE PAS (le script
# est network/loopback.sh). Resultat : enable_user_loopback echouait sur toute
# machine sans utilisateur nirvana, et disable_user_loopback (appele a l'arret)
# etait deja du code mort ICI. On resout les deux a l'execution.
#
# Le critere du bon utilisateur n'est pas son nom mais la PRESENCE de son socket
# pulse : c'est la seule chose qui distingue une session graphique vivante d'un
# compte qui existe. Meme logique que session_user() de wslg-audio-bridge.sh,
# etendue au balayage de /run/user/*.
session_pulse_user() {
    local u uid sock
    for u in "${HOST_PULSE_USER:-}" "${SUDO_USER:-}" "$(logname 2>/dev/null || true)" "${USER:-}"; do
        [ -n "$u" ] || continue
        uid="$(id -u "$u" 2>/dev/null)" || continue
        [ -S "/run/user/${uid}/pulse/native" ] && { echo "$u"; return 0; }
    done
    # Dernier recours : proprietaire du premier socket pulse trouve. Couvre le
    # `sudo -i` (SUDO_USER perdu) et les sessions sans logname (cron, service).
    for sock in /run/user/*/pulse/native; do
        [ -S "$sock" ] || continue
        uid="${sock#/run/user/}"; uid="${uid%%/*}"
        u="$(getent passwd "$uid" 2>/dev/null | cut -d: -f1)"
        [ -n "$u" ] && { echo "$u"; return 0; }
    done
    return 1
}

# Le script de loopback vit dans le depot, pas dans un $HOME.
find_loopback_script() {
    local d c
    d="$START_REPO"
    for c in "$d/network/loopback.sh" "$d/loopback.sh"; do
        [ -x "$c" ] && { echo "$c"; return 0; }
    done
```

```
    return 1
}

enable_user_loopback() {
    local target_user target_uid target_runtime loopback_script
    target_user="$(session_pulse_user)" || {
        echo -e "  ${YELLOW}[audio] aucune session PulseAudio utilisateur - loopback ignore${NC}"
        return 0
    }
    loopback_script="$(find_loopback_script)" || {
        echo -e "  ${YELLOW}[audio] network/loopback.sh introuvable - loopback ignore${NC}"
        return 0
    }
    target_uid="$(id -u "$target_user")"
    target_runtime="/run/user/${target_uid}"
    echo -e "${GREEN}=== [audio] Loopback user session ===${NC}"
    echo -e "  user    : ${CYAN}${target_user}${NC}"
    echo -e "  runtime : ${CYAN}${target_runtime}${NC}"
    echo -e "  script  : ${CYAN}${loopback_script}${NC}"
    sudo -u "$target_user" \
        XDG_RUNTIME_DIR="$target_runtime" \
        PULSE_SERVER="unix:${target_runtime}/pulse/native" \
        "$loopback_script" enable >/dev/null 2>&1 || {
            echo -e "  ${RED}[FAIL]${NC} enable loopback"
            return 1
        }
    echo -e "[ OK ]${NC} loopback active"
}

disable_user_loopback() {
    local target_user target_uid target_runtime loopback_script
    target_user="$(session_pulse_user)"       || return 0
    loopback_script="$(find_loopback_script)" || return 0
    target_uid="$(id -u "$target_user")"
    target_runtime="/run/user/${target_uid}"
    sudo -u "$target_user" \
        XDG_RUNTIME_DIR="$target_runtime" \
        PULSE_SERVER="unix:${target_runtime}/pulse/native" \
        "$loopback_script" disable || true
}

build_alsa_args() {
    local alsa_args=""
    local src_asound="$START_REPO/configs/asound.conf"
    local host_asound="/tmp/osmocom-alsa/asound.conf"
    mkdir -p /tmp/osmocom-alsa
    # Docker cree un REPERTOIRE quand la source d'un bind -v n'existe pas :
    # si un run precedent a laisse ce repertoire, le mount sur /etc/asound.conf
    # (un fichier) echoue avec "not a directory". On le nettoie avant la copie.
    if [ -d "$host_asound" ]; then
        rm -rf "$host_asound"
    fi
    if [ -d /dev/snd ]; then
        alsa_args="${alsa_args} --device /dev/snd"
        if getent group audio >/dev/null 2>&1; then
            alsa_args="${alsa_args} --group-add $(getent group audio | cut -d: -f3)"
        fi
    fi
    local has_pulse="true"
    # ── LE SOCKET PULSE DOIT ETRE MONTE, PAS SEULEMENT NOMME ──────────────
    # [2026-08-31] Cette ligne posait PULSE_SERVER=unix:/run/pulse/native sans
    # que RIEN ne monte ce socket dans le conteneur. Le chemin existe sur
    # l hote, pas chez l invite : le plugin ALSA-pulse ouvrait donc dans le
    # vide, et gapk mourait a l initialisation -
    #     pulse.c:242 (pulse_connect) PulseAudio: Unable to connect: Connection refused
    #     pq_alsa.c:168 Couldn't init ALSA device 'gsm_out': Connection refused
```

```
    #     gapk_io.c:468 Failed to initialize GAPK I/O
    # soit un appel parfaitement etabli et TOTALEMENT muet, sans une ligne qui
    # parle d audio dans les logs du coeur. Verifie le 31/08 dans
    # osmo-operator-1 : `docker inspect` ne montrait aucun montage pulse, et
    # `PULSE_SERVER=unix:/run/pulse/native pactl info` echouait dans le
    # conteneur la ou tcp:172.20.0.1:4713 repondait.
    #
    # On monte donc le socket quand il existe - c est le chemin le plus court
    # et le moins bavard - et on retombe sur le relais TCP sinon. Le socket est
    # en srwxrwxrwx et le module unix est en auth-anonymous=1 : l utilisateur
    # osmocom du conteneur peut s y connecter.
    local _pa_sock=""
    for _s in /var/run/pulse/native /run/pulse/native "/run/user/$(id -u)/pulse/native"; do
        [ -S "$_s" ] && { _pa_sock="$_s"; break; }
    done
    if [ -n "$_pa_sock" ]; then
        alsa_args="${alsa_args} -v ${_pa_sock}:/run/pulse/native"
        alsa_args="${alsa_args} -e PULSE_SERVER=unix:/run/pulse/native"
    elif [ -n "${HOST_AUDIO_RELAY:-}" ]; then
        alsa_args="${alsa_args} -e PULSE_SERVER=${HOST_AUDIO_RELAY}"
    else
        # Ni socket ni relais : on le DIT. Sans ce message, le silence de
        # l appel restait la seule trace du probleme.
        alsa_args="${alsa_args} -e PULSE_SERVER=tcp:${INTER_NET_GATEWAY:-172.20.0.1}:${WSLG_TCP_PORT:-4713}"
        echo -e "  ${YELLOW}Audio : aucun socket pulse trouve et relais non arme - repli TCP a l aveugle${NC}" >&2
    fi
    local asound_ok="false"
    if [ -f "$src_asound" ]; then
        if cp -f "$src_asound" "$host_asound" 2>/dev/null && [ -f "$host_asound" ]; then
            asound_ok="true"
        else
            echo -e "  ${YELLOW}Audio : copie de asound.conf vers ${host_asound} impossible${NC}" >&2
        fi
    fi
    if [ "$asound_ok" = "true" ]; then
        alsa_args="${alsa_args} -v ${host_asound}:/etc/asound.conf:rw"
        alsa_args="${alsa_args} -e ALSA_OUTPUT=gsm_out -e ALSA_INPUT=gsm_in"
        alsa_args="${alsa_args} -e ALSA_CARD=gsm_out -e GAPK_ALSA_DEV=gsm_out"
        if [ "$has_pulse" = "true" ]; then
            echo -e "  ${GREEN}Audio : PulseAudio + asound.conf${NC}" >&2
        else
            echo -e "  ${YELLOW}Audio : asound.conf sans PulseAudio${NC}" >&2
        fi
    else
        alsa_args="${alsa_args} -e ALSA_OUTPUT=default -e ALSA_INPUT=default"
        alsa_args="${alsa_args} -e ALSA_CARD=default -e GAPK_ALSA_DEV=default"
        echo -e "  ${YELLOW}Audio : configs/asound.conf absent, fallback default${NC}" >&2
    fi
    echo "$alsa_args"
}

# ── Audio hote : expose le pulse de l'hote en TCP ──────────────────────
ensure_host_audio_relay() {
    [ -n "${HOST_AUDIO_RELAY_DONE:-}" ] && return 0
    HOST_AUDIO_RELAY_DONE=1
    local bridge="$START_REPO/scripts/wslg-audio-bridge.sh"
    [ -x "$bridge" ] || { echo -e "  ${YELLOW}[host-audio] $bridge introuvable${NC}" >&2; return 0; }
    if "$bridge" host-relay; then
        # [2026-08-31] ON VERIFIE QUE LE PORT REPOND, on ne se fie plus au seul
        # code de retour. `host-relay` rend 0 des que le chargement du module a
        # ete DEMANDE - pas quand pulseaudio accepte des connexions. On
        # annoncait donc "relai pret" pendant que gapk, dans le conteneur,
        # recevait :
        #     lib pulse.c:242:(pulse_connect) PulseAudio: Unable to connect:
        #         Connection refused
        #     pq_alsa.c:168 Couldn't init ALSA device 'gsm_out': Connection refused
```

```
        #     gapk_io.c:468 Failed to initialize GAPK I/O
        # -> appel etabli et TOTALEMENT muet, avec un demarrage qui affichait
        # "relai pret" en vert. Constate le 31/08, cinq minutes apres un boot :
        # la course se gagne ou se perd selon ce que la machine fait par
        # ailleurs, ce qui en fait un "ca marche une fois sur deux".
        # gapk n init son I/O QU UNE FOIS : rater cette fenetre, c est le
        # silence pour tout l appel.
        local _p=0 _i
        for _i in $(seq 1 30); do
            if (exec 3<>"/dev/tcp/${INTER_NET_GATEWAY}/${WSLG_TCP_PORT:-4713}") 2>/dev/null; then
                exec 3>&- 2>/dev/null; _p=1; break
            fi
            sleep 1
        done
        if [ "$_p" = "1" ]; then
            HOST_AUDIO_RELAY="tcp:${INTER_NET_GATEWAY}:${WSLG_TCP_PORT:-4713}"
            echo -e "  ${GREEN}[host-audio] relai pret → ${HOST_AUDIO_RELAY}${NC}"
        else
            HOST_AUDIO_RELAY=""
            echo -e "  ${RED}[host-audio] relai charge mais ${INTER_NET_GATEWAY}:${WSLG_TCP_PORT:-4713} NE REPOND
             ↪  PAS apres 30 s${NC}" >&2
            echo -e "  ${YELLOW}             → l audio du conteneur sera muet. Verifier :${NC}" >&2
            echo -e "  ${YELLOW}               pactl list modules | grep -A2 native-protocol-tcp${NC}" >&2
        fi
    else
        HOST_AUDIO_RELAY=""
        echo -e "  ${YELLOW}[host-audio] relai non ouvert - fallback carte locale dans le conteneur${NC}"
    fi

    # [2026-08-12] Sens MONTANT (micro). Le relai ci-dessus ne couvre que la
    # descente (gsm_audio.monitor → HP de l'hote). Le micro, lui, vient du
    # navigateur, qui capture la source PAR DEFAUT de l'hote - laquelle peut
    # tres bien etre une entree fantome qui ne sort que du zero (vecu ici :
    # defaut non mute a 93 % et pourtant silence total, cf. host_mic()).
    # On choisit donc la source a l'oreille, pas au nom. Non fatal.
    "$bridge" host-mic || true
}

# ═══════════════════════════════════════════════════════════════════════
# Build
# ═══════════════════════════════════════════════════════════════════════
# [2026-08-31] QUICK PAR DEFAUT, et sans poser la question.
# Le defaut etait 0 - "normal (--no-cache)" - et, faute de choix explicite,
# start-docker.sh ouvrait une boite whiptail (l.2578, choose_build_mode) pour la
# demander. Deux consequences :
#   - la recompilation Osmocom complete, ~40 minutes, etait ce qu on obtenait
#     en repondant trop vite, alors que l image ne change quasiment jamais ;
#   - la QUESTION elle-meme rendait le script inutilisable depuis une icone ou
#     un appel scripte : il s arretait sur un dialogue que personne ne voit.
# Le cache docker est la bonne reponse dans la vie de tous les jours ; on ne
# reconstruit a neuf que quand on le DEMANDE.
#
# Comment forcer une reconstruction complete, les deux marchent :
#     ./launch/start-docker.sh normal ...        (argument positionnel)
#     OSMO_QUICK=0 ./launch/start-docker.sh ...  (variable d environnement)
QUICK="${OSMO_QUICK:-1}"
# EXPLICIT a 1 meme sans demande : c est ce qui supprime le dialogue. Le passer
# a 0 ici ferait revenir la question a chaque lancement non interactif.
QUICK_EXPLICIT=1

build_run_image() {
    # [2026-09-03] docker compose v2 (compose.yaml, service "run") quand il est
    # la - c est lui qui porte les build-args et le nom d image - et docker
    # build tel quel sinon. Le resultat est le meme : osmocom-run depuis
    # Dockerfile.run, par-dessus osmocom-nitb.
    local _dir; _dir="${OSMO_REPO_DIR:-$START_REPO}"
```

```
    local _nc=""; [ "${QUICK:-0}" = "1" ] || _nc="--no-cache"
    if [ "${QUICK:-0}" = "1" ]; then
        echo -e "${GREEN}Build de l'image run...${NC} ${YELLOW}(quick - cache docker reutilise)${NC}"
    else
        echo -e "${GREEN}Build de l'image run...${NC} ${CYAN}(normal - --no-cache)${NC}"
    fi
    if docker compose version >/dev/null 2>&1 && [ -f "$_dir/compose.yaml" ]; then
        ( cd "$_dir" && QEMU_CACHE_BUST=$(date +%s) IMAGE_RUN="$IMAGE_RUN" \
          docker compose -f "$_dir/compose.yaml" build $_nc run )
    else
        docker build $_nc --build-arg QEMU_CACHE_BUST=$(date +%s) -f Dockerfile.run -t "$IMAGE_RUN" .
    fi
    echo -e "${GREEN}Image '$IMAGE_RUN' prete.${NC}"
}

check_image() {
    if ! docker image inspect "$IMAGE_RUN" &>/dev/null; then
        echo -e "${RED}Image '$IMAGE_RUN' introuvable - build en cours...${NC}"
        build_run_image
    fi
}

# ═══════════════════════════════════════════════════════════════════════
# Generation dynamique - pjsip interop trunks
# ═══════════════════════════════════════════════════════════════════════
generate_pjsip_interop_trunks() {
    local op_id=$1
    local n_operators=$2
    # ── LES OPERATEURS SOUS LA BASE SONT LES NATIFS ──────────────────
    # [2026-08-31] La boucle partait DE la base. Avec OP_ID_BASE=2 - les
    # conteneurs prenant les rangs 2..N, le 1 restant au natif sur l hote -
    # op2 ne fabriquait qu un trunk vers op3 : RIEN vers le natif. Un appel
    # 100201 -> 100101 tombait dans le fourre-tout _X. de [interop_out] et
    # sortait en Congestion, sans qu une ligne dise que la route n existe pas.
    # Meme trou que sms-routing.conf, sur le chemin VOIX.
    for remote_op in $(seq 1 $(( ${OP_ID_BASE:-1} - 1 ))) \
                     $(seq "${OP_ID_BASE:-1}" "$(( ${OP_ID_BASE:-1} + n_operators - 1 ))"); do
        [ "$remote_op" -eq "$op_id" ] && continue
        local remote_ip
        # Un operateur sous la base est NATIF : il tourne sur l hote, pas dans
        # un conteneur. Vu d un conteneur, l hote est la passerelle du bridge
        # docker - la meme adresse que le pont audio et que sms-routing.conf.
        # NB : match=172.20.0.1 n avale pas les REGISTER Linphone, qui arrivent
        # par la meme passerelle : pjsip.conf pose endpoint_identifier_order=
        # username,ip,anonymous - le username passe avant l IP.
        if [ "$remote_op" -lt "${OP_ID_BASE:-1}" ]; then
            remote_ip="${INTER_NET_GATEWAY:-172.20.0.1}"
        else
            remote_ip=$(op_backbone_ip "$remote_op")
        fi
        cat <<EOF

[interop-identify-op${remote_op}]
type=identify
endpoint=interop_trunk_op${remote_op}
match=${remote_ip}

[interop_trunk_op${remote_op}]
type=endpoint
transport=transport-udp
context=interop_in
disallow=all
allow=gsm
allow=ulaw
aors=interop_trunk_op${remote_op}
direct_media=no
```

```
rtp_symmetric=yes
force_rport=yes
media_encryption=no

[interop_trunk_op${remote_op}]
type=aor
contact=sip:${remote_ip}:5060
qualify_frequency=15
qualify_timeout=5.0
EOF
    done
}

generate_extensions_interop_out() {
    local op_id=$1
    local n_operators=$2
    # Le prefixe porte le NOEUD : MSISDN = <noeud>00<operateur><rang>. Fige a
    # "600", le motif n'accrochait plus rien des que le numero commencait par le
    # numero de noeud, et les appels inter-operateurs sortaient en Congestion
    # sans qu'une ligne ne designe le motif.
    local _pfx; _pfx="$(osmo_msisdn_pfx "$(osmo_node_id)")"
    cat <<'EOF'
[interop_out]

EOF
    # Un seul motif : les MSISDN font six chiffres, <noeud>00<operateur><rang>.
    # deux motifs _<op>XXXX / _<op>XXXXX visaient l'ancien plan a cinq chiffres,
    # ou le premier chiffre du numero ETAIT l'operateur - plus rien ne matchait.
    # ── LES OPERATEURS SOUS LA BASE SONT LES NATIFS ──────────────────────
    # [2026-08-31] La boucle partait DE la base. Avec OP_ID_BASE=2 - les
    # conteneurs prenant les rangs 2..N, le 1 restant au natif sur l hote -
    # op2 ne fabriquait qu un trunk vers op3 : RIEN vers le natif. Un appel
    # 100201 -> 100101 tombait dans le fourre-tout _X. de [interop_out] et
    # sortait en Congestion, sans qu une ligne dise que la route n existe pas.
    # Meme trou que sms-routing.conf, sur le chemin VOIX.
    for remote_op in $(seq 1 $(( ${OP_ID_BASE:-1} - 1 ))) \
                     $(seq "${OP_ID_BASE:-1}" "$(( ${OP_ID_BASE:-1} + n_operators - 1 ))"); do
        [ "$remote_op" -eq "$op_id" ] && continue
        cat <<EOF
exten => _${_pfx}${remote_op}XX,1,NoOp(=== INTEROP OUT Op${remote_op}: \${EXTEN} ===)
 same => n,Dial(PJSIP/\${EXTEN}@interop_trunk_op${remote_op},,rT)
 same => n,Congestion()
 same => n,Hangup()

EOF
    done

    # ── LES SOFTPHONES AUSSI, ET SANS PREFIXE ────────────────────────────
    # [2026-08-31] Depuis [internal], un Linphone ne pouvait joindre un autre
    # operateur qu'en composant 9 + numero (_9. du gabarit - dont l'exemple
    # « 920001 pour joindre 20001 sur Op2 » date d'ailleurs du plan a CINQ
    # chiffres, abandonne). Compose tel quel, 100201 tombait sur le fourre-tout
    # _X. de [internal] et partait au MSC LOCAL, qui le rendait en
    #     rx MNCC_SETUP_REQ for unknown subscriber number '100201'
    # avant de le ressortir par MNCC vers Asterisk. Le detour aboutissait, mais
    # il consommait une transaction MNCC pour rien et faisait dependre un appel
    # SIP->SIP du bon vouloir du MSC.
    # On declare donc la route explicitement. _<pfx><op>XX est PLUS SPECIFIQUE
    # que _X. : Asterisk la choisit d'office, sans qu'il faille toucher au
    # fourre-tout du gabarit. Gosub(sub-record) comme pour _9., sinon ces
    # appels-la seraient les seuls a ne pas etre enregistres.
    printf '[internal]\n\n'
    for remote_op in $(seq 1 $(( ${OP_ID_BASE:-1} - 1 ))) \
                     $(seq "${OP_ID_BASE:-1}" "$(( ${OP_ID_BASE:-1} + n_operators - 1 ))"); do
        [ "$remote_op" -eq "$op_id" ] && continue
        cat <<EOF
```

```
exten => _${_pfx}${remote_op}XX,1,NoOp(=== SIP -> INTER-OP Op${remote_op}: \${EXTEN} - CallerID:
 ↪  \${CALLERID(all)} ===)
 same => n,Gosub(sub-record,s,1(\${EXTEN}))
 same => n,Goto(interop_out,\${EXTEN},1)

EOF
    done
    # PAS de fourre-tout _X. ici : le gabarit configs/extensions.conf en declare
    # deja un dans ce meme contexte. Deux extensions identiques dans un contexte
    # font refuser la seconde, avec six avertissements a chaque chargement :
    #     add_priority: Unable to register extension '_X.' priority 1
    #                   in 'interop_out', already in use
    # Rien ne cassait - le fourre-tout du gabarit restait en place - mais ces
    # lignes noyaient celles qui comptent.
}

# ═══════════════════════════════════════════════════════════════════════
# SMS routing fallback
# ═══════════════════════════════════════════════════════════════════════
_generate_sms_routing_conf_fallback() {
    # 'local i' n'est PAS cosmetique. Cette fonction est appelee, via
    # apply_config_templates, DEPUIS la boucle "for i in ..." qui demarre les
    # operateurs. Sans local, sa propre boucle ecrase la variable de l'appelant
    # et la laisse a n_operators : tous les conteneurs suivants calculaient
    # alors leurs ports avec le MEME indice, d'ou
    #     Bind for 0.0.0.0:5082 failed: port is already allocated
    # au deuxieme conteneur - et, avant l'erreur, deux operateurs annonces sur
    # les memes SIP/RTP/SMS.
    local i
    local op_id=$1 n_operators=$2
    printf '# sms-routing.conf - Fallback\n\n[local]\noperator_id = %s\nsc_address  =
     ↪  1999001%s444\n\n[operators]\n' "$op_id" "$op_id"
    # ── LES OPERATEURS SOUS LA BASE SONT LES NATIFS ──────────────────────
    # [2026-08-31] start-multi.sh pose OP_ID_BASE=2 : les conteneurs prennent
    # les rangs 2..N et l'operateur 1 reste NATIF, sur l'hote. Comme la boucle
    # part de OP_ID_BASE, il n'apparaissait ni dans [operators] ni dans
    # [routes] - releve dans osmo-operator-2 : « 2 = ... / 3 = ... » et rien
    # pour le 1. Un SMS vers 100101 sortait donc en « No route for
    # destination », et le natif etait injoignable depuis tout le banc.
    # Il existe pourtant : il n'est simplement pas dans un conteneur. On le
    # pointe sur la passerelle du bridge docker, qui EST l'hote vu d'un
    # conteneur - la meme adresse que le pont audio (172.20.0.1).
    local _natif_gw="${INTER_NET_GATEWAY:-172.20.0.1}"
    for i in $(seq 1 $(( ${OP_ID_BASE:-1} - 1 ))); do printf '%s = %s\n' "$i" "$_natif_gw"; done
    for i in $(seq "${OP_ID_BASE:-1}" "$(( ${OP_ID_BASE:-1} + n_operators - 1 ))"); do
        printf '%s = %s\n' "$i" "$(op_backbone_ip "$i")"
    done
    printf '\n[routes]\n'
    for i in $(seq 1 $(( ${OP_ID_BASE:-1} - 1 ))); do
        for ms in 1 2; do printf '%s = %s\n' "$(osmo_msisdn "$(osmo_node_id)" "$i" "$ms")" "$i"; done
    done
    for i in $(seq "${OP_ID_BASE:-1}" "$(( ${OP_ID_BASE:-1} + n_operators - 1 ))"); do
        for ms in 1 2; do printf '%s = %s\n' "$(osmo_msisdn "$(osmo_node_id)" "$i" "$ms")" "$i"; done   # MSISDN
         ↪  exacts <noeud>00<op><ms> - PAS de concatenation
    done
    printf '\n[relay]\nport = 7890\nconnect_timeout = 10\nretry_count = 3\nretry_delay = 5\n'
}

# ═══════════════════════════════════════════════════════════════════════
# ──────────────────────────────────────────────────────────────────────
# Configuration du reseau - deleguee a ./generate_configs.sh
#
# [2026-08-03] Ce fichier portait ici une copie de 132 lignes de
# apply_config_templates(), jumelle de celle de lib/gabarits.sh (70 lignes).
# Deux implementations de la meme chose, qui divergeaient en silence : celle-ci
# copiait tout scripts/, l'autre une liste blanche perimee. Elles vivent
```

```
# desormais dans generate_configs.sh, une seule fois.
#
# generate_configs.sh cree globals.conf a la racine s'il n'existe pas - et ne
# l'ecrase JAMAIS ensuite. Tes reglages survivent donc a tous les runs. Pour
# repartir des valeurs d'usine : ./generate_configs.sh --force
# ──────────────────────────────────────────────────────────────────────────
# start-docker.sh n'a pas de variable de repertoire (il travaille en chemins relatifs
# depuis la racine du depot) : on resout ici, sans rien supposer.
#
# BASH_SOURCE ne suffit pas : build-iso.sh extrait ce fichier en bibliotheque
# ($WORK/start.lib.sh) puis la source, et BASH_SOURCE designe alors la COPIE,
# dans un repertoire de travail ou generate_configs.sh n'existe pas.
# OSMO_REPO_DIR laisse l'appelant nommer la vraie racine du depot.
_GC_SH="${OSMO_REPO_DIR:-$START_REPO}/generate_configs.sh"
"$_GC_SH" >/dev/null || true
. "$_GC_SH"


# ── Mise a jour forcee des trois arbres, AVANT run.sh ──────────────────────
# [2026-08-12] Les trois depots sont clones et `git pull`-es DANS LE Dockerfile,
# donc a l'etat du BUILD. Entre deux builds d'image (11,5 Go, plusieurs dizaines
# de minutes) un `git push` n'atteignait jamais un conteneur neuf : on lancait un
# run sur du code perime sans le voir. On les rafraichit donc a chaque demarrage,
# avant que run.sh ne lise quoi que ce soit.
#
# ORDRE : cette fonction DOIT etre appelee apres `docker run` et AVANT le
# `docker exec ... run.sh`. Apres, run.sh aurait deja lu l'ancien code.
#
# `--ff-only`, PAS de `reset --hard` : le depot qosmo du conteneur recoit des
# commits faits a la main pendant une session (c'est le mode de travail : on
# corrige, on compile, on commite dans le conteneur). Un reset dur les
# effacerait sans le dire. En avance-rapide seule, une divergence s'ARRETE et se
# VOIT au lieu d'etre ecrasee en silence.
#
# qosmo est RECOMPILE apres le pull : mettre la source a jour sans relier le
# binaire ne change rien a ce qui tourne - et c'est indiscernable d'un pull qui
# aurait echoue. Le juge d'un binaire a jour reste `stat -L /proc/<pid>/exe`.
#
# Non bloquant (reseau absent = on continue), mais BRUYANT : un pull rate doit
# se lire dans la sortie du lanceur, pas se deviner.
force_update_trees() {
    local c=$1
    echo -e "  ${GREEN}[*] Mise a jour forcee des depots (avant run.sh)...${NC}"
    for repo in /opt/GSM/qosmo /opt/GSM/osmo-operator /opt/GSM/osmo-egprs-web; do
        if ! docker exec "$c" test -d "$repo/.git" 2>/dev/null; then
            echo -e "    ${YELLOW}$repo : pas un depot git - ignore${NC}"
            continue
        fi
        local before after
        before=$(docker exec "$c" git -C "$repo" rev-parse --short HEAD 2>/dev/null || echo '?')
        if docker exec "$c" git -C "$repo" pull --ff-only 2>&1 | tail -2 | sed 's/^/    /'; then
            after=$(docker exec "$c" git -C "$repo" rev-parse --short HEAD 2>/dev/null || echo '?')
            if [ "$before" = "$after" ]; then
                echo -e "    ${CYAN}$repo${NC} : deja a jour ($after)"
            else
                echo -e "    ${GREEN}$repo${NC} : $before -> $after"
                # ── LE SERVEUR WEB TOURNE DEJA, ET NODE A DEJA LU SON FICHIER ──
                # [2026-08-31] Exactement le piege du ninja ci-dessous, sur un
                # autre maillon : le pull change le FICHIER, pas le PROCESSUS.
                # osmo-egprs-web.service est une unite systemd, donc lancee au
                # BOOT du conteneur - avant cet appel - et node ne lit server.js
                # qu'une fois. Mesure du 31/08 dans osmo-operator-2 :
                #     node demarre 18:58:18   server.js reecrit 18:58:21
                # Le disque portait le correctif Kc (grep kcPollVty : 2), le
                # process tournait encore sur l'ancien code, et le dashboard
                # affichait « Kc absent » - panne indiscernable d'un pull rate.
```

```
                # On ne relance QUE si le pull a reellement bouge.
                if [ "$repo" = /opt/GSM/osmo-egprs-web ]; then
                    if docker exec "$c" systemctl restart osmo-egprs-web 2>/dev/null; then
                        echo -e "    ${GREEN}osmo-egprs-web.service relance${NC} (le dashboard charge le nouveau
                         ↪  code)"
                    else
                        echo -e "    ${YELLOW}osmo-egprs-web.service : relance KO - le dashboard garde l'ancien
                         ↪  code${NC}"
                    fi
                fi
            fi
        else
            echo -e "    ${RED}$repo : pull KO (divergence ou reseau) - le run part sur $before${NC}"
        fi
    done
    # Recompilation de QEMU : sans elle le pull ci-dessus ne change RIEN au
    # binaire qui tourne. ninja ne reconstruit que ce qui a bouge.
    echo -e "  ${GREEN}[*] Recompilation QEMU (ninja)...${NC}"
    if docker exec "$c" bash -c 'cd /opt/GSM/qosmo/build && ninja' >/dev/null 2>&1; then
        echo -e "    ${GREEN}qemu-system-arm relie${NC} ($(docker exec "$c" stat -L -c %y
         ↪  /opt/GSM/qosmo/build/qemu-system-arm 2>/dev/null | cut -c1-19))"
    else
        echo -e "    ${RED}ninja KO - le run utilisera le binaire de l'image${NC}"
        docker exec "$c" bash -c 'cd /opt/GSM/qosmo/build && ninja 2>&1 | tail -15' | sed 's/^/      /' || true
    fi
}

build_vol_args() {
    local tmpdir=$1
    local vol_args=""
    [ -d "$tmpdir/osmocom" ] && vol_args="$vol_args -v $tmpdir/osmocom:/etc/osmocom"
    for f in "$tmpdir/asterisk"/*.conf; do
        [ -f "$f" ] || continue
        vol_args="$vol_args -v $f:/etc/asterisk/$(basename "$f")"
    done
    if [ -f "$tmpdir/bb/mobile.cfg" ]; then
        vol_args="$vol_args -v $tmpdir/bb/mobile.cfg:/root/.osmocom/bb/mobile.cfg"
    fi
    if [ -f "$tmpdir/bb/mobile_group1.cfg" ]; then
        vol_args="$vol_args -v $tmpdir/bb/mobile_group1.cfg:/root/.osmocom/bb/mobile_group1.cfg"
    fi
    echo "$vol_args"
}

# ═══════════════════════════════════════════════════════════════════
# TUN hote
# ═══════════════════════════════════════════════════════════════════
prepare_host_tun() {
    echo -e "${GREEN}[*] Configuration TUN sur l'hote...${NC}"
    modprobe tun 2>/dev/null || true
    mkdir -p /dev/net
    if [ ! -c /dev/net/tun ]; then
        mknod /dev/net/tun c 10 200
        chmod 666 /dev/net/tun
    fi
    ip link del apn0 2>/dev/null || true
    ip tuntap add dev apn0 mode tun
    ip addr add 176.16.32.0/24 dev apn0
    ip link set apn0 up
}

# ═══════════════════════════════════════════════════════════════════
# Inter-STP
# ═══════════════════════════════════════════════════════════════════
start_inter_stp() {
    local n_operators=$1
```

```
local tmpdir
tmpdir=$(mktemp -d)
local inter_cfg="${tmpdir}/osmo-stp-interop.cfg"
if [ "${WAN_MESH:-0}" = "1" ]; then
    # Plan WAN : un AS par couple (noeud, operateur), point codes
    # 1.<noeud><op>.<role>. Le hub doit connaitre TOUS les noeuds, pas
    # seulement les operateurs locaux - sinon les ASP distants s'attachent
    # a un AS qui n'existe pas et sont rejetes sans explication.
    echo -e "${GREEN}Generation config inter-STP WAN (${WAN_NODE_COUNT} noeuds × ${n_operators})...${NC}"
    bash ./helpers/create_interop.sh --wan "$WAN_NODE_COUNT" "$n_operators" "$inter_cfg" > /dev/null
else
    echo -e "${GREEN}Generation config inter-STP (${n_operators} operateurs)...${NC}"
    bash ./helpers/create_interop.sh "$n_operators" "$inter_cfg" > /dev/null
fi
if [ ! -f "$inter_cfg" ]; then
    echo -e "${RED}Echec generation config inter-STP${NC}"; exit 1
fi
# Le hub ne fait que router du M3UA : l'image osmocom-stp (Dockerfile.stp)
# lui suffit et pese une fraction d'osmocom-run. On ne la CONSTRUIT pas ici
# a l'insu de l'appelant - si elle manque, on reste sur l'image complete,
# qui contient aussi osmo-stp.
local stp_image="$IMAGE_RUN"
if [ "${BUILD_STP:-0}" = "1" ] && ! docker image inspect "$IMAGE_STP" >/dev/null 2>&1; then
    echo -e "  ${GREEN}Construction de ${IMAGE_STP} (Dockerfile.stp)...${NC}"
    docker build -f "$START_REPO/Dockerfile.stp" -t "$IMAGE_STP" "$START_REPO" \
        || echo -e "  ${YELLOW}⚠ echec - on restera sur ${IMAGE_RUN}${NC}"
fi
if docker image inspect "$IMAGE_STP" >/dev/null 2>&1; then
    stp_image="$IMAGE_STP"
    echo -e "  ${CYAN}image ${IMAGE_STP} (STP seul)${NC}"
elif [ "${WAN_MESH:-0}" = "1" ]; then
    echo -e "  ${YELLOW}image ${IMAGE_STP} absente - on utilise ${IMAGE_RUN}.${NC}"
    echo -e "  ${CYAN}Pour la construire : docker build -f Dockerfile.stp -t ${IMAGE_STP} .${NC}"
fi

echo -e "${GREEN}Lancement inter-STP @ ${INTER_STP_IP}:2908 (PC 0.0.0)...${NC}"
docker rm -f "$INTER_STP_CONTAINER" &>/dev/null || true
# --restart, PAS --rm : avec --rm, un reboot de l'hote ne laissait pas le hub
# arrete, il le faisait DISPARAITRE - au retour, plus rien a redemarrer, et
# des ASP qui ouvrent leur SCTP vers personne. Les deux options sont
# exclusives (docker refuse la combinaison au parsing), et le "docker rm -f"
# ci-dessus tient deja le role de --rm : aucun reliquat du lancement d'avant.
# ── 2908 PUBLIE SUR LA BOUCLE LOCALE : LE RACCORD DE L OPERATEUR NATIF ──
# [2026-08-31] Sans cette ligne, un operateur NATIF (celui de
# start-native.sh, hors conteneur) ne pouvait PAS s attacher au hub.
# Son /etc/osmocom/osmo-stp.cfg declare
#       asp asp-to-inter 2908 2910 m3ua
#        remote-ip 127.0.0.1
# ce qui est le bon choix - c est la seule adresse dont on soit sur - mais
# RIEN n ecoutait sur 127.0.0.1:2908 cote hote : le hub vit dans le
# conteneur, sur le reseau prive, et ne publiait aucun port
# ("docker port osmo-inter-stp" : vide). L ASP ouvrait donc son SCTP vers
# personne, et le diagnostic sortait tres loin de la cause -
#     Aucune association SCTP vers ...:2908 (0/1)
#     as-inter : AS_DOWN  /  Route par defaut -> inter-STP absente
# ce qui envoie chercher un hub en panne ou un filtrage, alors que le hub
# va bien et que rien ne le rendait joignable.
#
# /sctp, PAS /tcp : M3UA roule sur SCTP. Docker sait publier ce protocole
# (verifie sur ce banc : noyau avec module sctp, et
# "docker run -p 127.0.0.1:PORT:PORT/sctp" accepte).
#
# Lie a 127.0.0.1 et pas a 0.0.0.0 : ce point d entree ne sert QU au natif,
# sur la meme machine. L exposer sur toutes les interfaces ouvrirait le
# coeur SS7 au LAN - checks/wan_ss7_check.sh signale d ailleurs "un port
# M3UA expose sur l interface publique" comme un defaut a part entiere.
```

```
    #
    # L autre solution possible etait de repointer l ASP natif sur
    # 172.20.0.10 avec local-ip 172.20.0.1. Ecartee : ces deux adresses
    # n existent que TANT QUE le bridge docker est monte, et elles disparaissent
    # avec les conteneurs - on aurait remplace une adresse absente par deux
    # adresses intermittentes, dans une config que start-native.sh regenere.
    docker run -d \
        --restart unless-stopped \
        --name "$INTER_STP_CONTAINER" \
        --hostname "$INTER_STP_CONTAINER" \
        --network "$INTER_NET" \
        --ip "$INTER_STP_IP" \
        -p "127.0.0.1:2908:2908/sctp" \
        --cap-add NET_ADMIN \
        -v "${inter_cfg}:/etc/osmocom/osmo-stp-interop.cfg" \
        --entrypoint bash \
        "$stp_image" \
        -c "sleep infinity" > /dev/null
    # tmux si l'image en a un, sinon un simple processus detache.
    #
    # osmocom-stp est l'image "STP seul" : elle ne porte que osmo-stp et ses
    # bibliotheques - c'est tout son interet, 877 Mo contre 11 Go. tmux n'y est
    # pas, et l'exec echouait alors sur "tmux: executable file not found in
    # $PATH". Le conteneur tournait pourtant (sleep infinity) : un hub bien
    # demarre, mais SANS osmo-stp dedans - il n'ecoutait rien, et les noeuds ne
    # s'attachaient a personne.
    #
    # tmux n'apportait ici que le confort de s'attacher a la session. La
    # barriere qui suit ne lit que /tmp/osmo-stp.log : un processus detache qui
    # redirige vers ce meme fichier remplit le contrat, et l'image complete
    # garde son tmux quand elle en a un.
    if docker exec "$INTER_STP_CONTAINER" sh -c 'command -v tmux >/dev/null 2>&1'; then
        docker exec "$INTER_STP_CONTAINER" \
            tmux new-session -d -s stp \
            "osmo-stp -c /etc/osmocom/osmo-stp-interop.cfg 2>&1 | tee /tmp/osmo-stp.log"
    else
        docker exec -d "$INTER_STP_CONTAINER" \
            sh -c 'exec osmo-stp -c /etc/osmocom/osmo-stp-interop.cfg > /tmp/osmo-stp.log 2>&1'
    fi
    echo -ne "${GREEN}[*] Attente demarrage inter-STP"
    local retries=25
    while [ $retries -gt 0 ]; do
        if docker exec "$INTER_STP_CONTAINER" \
            grep -qE "listening|m3ua.*[Ss]erver|bound" /tmp/osmo-stp.log 2>/dev/null; then
            echo -e " ${GREEN}✓${NC}"
            return 0
        fi
        if [ "$(docker inspect -f '{{.State.Running}}' "$INTER_STP_CONTAINER" 2>/dev/null)" != "true" ]; then
            echo -e " ${RED}CRASH${NC}"
            docker logs "$INTER_STP_CONTAINER"
            exit 1
        fi
        echo -n "."
        sleep 1
        ((retries--)) || true
    done
    echo -e " ${YELLOW}(timeout log, on continue)${NC}"
}

wait_inter_stp_ready() {
    local n_operators=$1
    echo -ne "${GREEN}[*] Verification stabilisation inter-STP (routes SS7)${NC}"
    for i in {1..15}; do sleep 1; echo -n "."; done
    echo -e " ${GREEN}✓${NC}"
    return 0
}
```

```
wait_bb_vty() {
    local container="$1"
    local timeout=120
    local elapsed=0
    echo -ne "  Attente BB VTY 4247 "
    while ! docker exec "$container" bash -c "echo >/dev/tcp/127.0.0.1/4247" 2>/dev/null; do
        sleep 2; elapsed=$((elapsed + 2)); echo -n "."
        if [ "$elapsed" -ge "$timeout" ]; then echo -e " ${RED}TIMEOUT${NC}"; return 1; fi
    done
    echo -e " ${GREEN}OK${NC}"
}

wait_stp_vty() {
    local container="$1"
    local timeout=60
    local elapsed=0
    echo -ne "  Attente STP VTY 4239 "
    while ! docker exec "$container" bash -c "echo >/dev/tcp/127.0.0.1/4239" 2>/dev/null; do
        sleep 2; elapsed=$((elapsed + 2)); echo -n "."
        if [ "$elapsed" -ge "$timeout" ]; then echo -e " ${RED}TIMEOUT${NC}"; return 1; fi
    done
    echo -e " ${GREEN}OK${NC}"
}

# ═══════════════════════════════════════════════════════════════════
# WAN Interop
# ═══════════════════════════════════════════════════════════════════
setup_wan_interop() {
    local n_local=$1
    local n_remote=$2
    local script_path
    script_path="$START_REPO/network/setup-wan-interop.sh"
    if [ ! -f "$script_path" ]; then
        echo -e "${RED}[WAN] network/setup-wan-interop.sh introuvable${NC}"
        return 1
    fi
    chmod +x "$script_path"
    bash "$script_path" "$WAN_LOCAL_IP" "$WAN_REMOTE_IP" "$n_local" "$n_remote"
}

# ═══════════════════════════════════════════════════════════════════
# WAN mesh (--wan)
# ═══════════════════════════════════════════════════════════════════
# Renseigne la table des noeuds : sans interaction si WAN_NODES est deja pose
# (option --wan-nodes ou variable d'environnement), par questions sinon.
wan_mesh_configure() {
    local n_operators=$1
    WAN_OPS="$n_operators"

    # Le point code WAN vaut 1.<noeud><op>.<role> : le champ central du format
    # ITU 3-8-3 tient sur 8 bits, soit 255. Avec 12 operateurs, le noeud 9
    # donnerait "912" - un point code invalide, refuse par osmo-stp au
    # demarrage avec une erreur de parsing qui ne dit rien du nombre
    # d'operateurs. On arrete ici, en nommant la cause.
    if [ "$n_operators" -gt 9 ]; then
        echo -e "${RED}[WAN] ${n_operators} operateurs : le plan de point codes WAN${NC}" >&2
        echo -e "${RED}      (1.<noeud><op>.<role>) n'en admet que 9 par noeud.${NC}" >&2
        echo -e "${CYAN}      Repartissez-les sur plusieurs noeuds, ou lancez sans --wan.${NC}" >&2
        exit 1
    fi

    # --virtualbox : le segment, les VM et la table sont fabriques AVANT tout le
    # reste, puis relus comme n'importe quelle table. Le WAN qui suit ne sait
    # pas - et n'a pas a savoir - que ses pairs sont des VM.
    if [ "${VBOX_INTERCO:-0}" = "1" ] && [ -z "${WAN_NODES:-}" ]; then
```

```
    local _vb=("$START_REPO/network/setup-vbox-interco.sh"
               --host-node "$VBOX_HOST_NODE" --conf "$WAN_CONF_FILE")
    [ -n "${VBOX_NODES:-}" ] && _vb+=(--nodes "$VBOX_NODES")
    bash "${_vb[@]}" || { echo -e "${RED}[WAN] interconnexion VirtualBox echouee${NC}" >&2; exit 1; }
    wan_nodes_load "$WAN_CONF_FILE" || exit 1
    WAN_NODES="$(wan_nodes_spec)"
    WAN_NODE_ID="$VBOX_HOST_NODE"
fi

wan_menu_table

if [ -n "${WAN_NODES:-}" ]; then
    wan_nodes_parse "$WAN_NODES" || exit 1
    if [ "${WAN_NODE_ID:-0}" = "0" ]; then
        # La detection par IP echoue par construction quand les premiers
        # noeuds sont des CONTENEURS : leurs adresses (172.20.0.11, .12)
        # appartiennent au backbone docker, pas a l'hote. L'hote n'est donc
        # dans la table sous aucune de ses adresses - et il n'a pas a y
        # etre : il porte des noeuds, il n'en est pas un.
        #
        # Son numero est celui de son PREMIER conteneur, c'est-a-dire 1
        # dans le modele "conteneur i = noeud i". On ne s'arrete donc plus :
        # on le pose, et on ne demande --wan-id que si la table ne contient
        # meme pas ce noeud-la.
        if wan_nodes_detect_self 2>/dev/null; then
            :
        elif [ -n "${WAN_IP[1]:-}" ]; then
            WAN_NODE_ID=1
            echo -e "  ${CYAN}[WAN] cette machine porte les conteneurs : noeud de base 1${NC}"
        else
            echo -e "${RED}[WAN] table vide ou sans noeud 1 : passez --wan-id N${NC}" >&2
            exit 1
        fi
    fi
    wan_nodes_validate || exit 1
else
    # Une table deja posee sur cette machine sert de valeurs par defaut :
    # relancer le lab ne redemande donc pas tout, il propose l'existant.
    wan_nodes_load "$WAN_CONF_FILE" 2>/dev/null || true
    wan_nodes_prompt || exit 1
fi

# ── Ce que la table WAN ne dit pas, et qu'il faut pourtant savoir ────────
# Elle donne les noeuds et leurs adresses. Elle ne dit ni ou est le hub SS7,
# ni si les conteneurs de CETTE machine sont ses operateurs ou des noeuds a
# part entiere. Sans ces reponses, --wan partait sur ses defauts - hub
# local, conteneurs = operateurs - et sur un banc ou le hub est une VM
# dediee, cela donnait un second hub inutile et des ASP attaches au mauvais.
wan_menu_complements

WAN_ENABLED="false"          # exclusif du WAN legacy
WAN_LOCAL_IP="$(wan_local_ip)"
echo ""
wan_nodes_summary
if [ -n "${WAN_HUB_IP:-}" ]; then
    echo -e "  inter-STP : ${CYAN}${WAN_HUB_IP}${NC}  (hors table : ce n'est pas un noeud)"
else
    echo -e "  inter-STP : porte par le noeud ${CYAN}${WAN_HUB_NODE}${NC}"
fi
[ "${WAN_NODE_PER_OP:-0}" = "1" ] \
    && echo -e "  conteneurs: ${CYAN}un noeud chacun${NC}, a partir du noeud ${CYAN}${WAN_NODE_ID}${NC}"
# Qui tourne ICI, et qui est seulement DECLARE. La distinction n'a rien de
# cosmetique : un noeud declare doit deja tourner ailleurs, sinon ses
# abonnes figurent dans les tables SMS et le dialplan sans que personne ne
# reponde - un appel qui sonne dans le vide se diagnostique mal.
for _nid in ${WAN_NODE_LIST[@]+"${WAN_NODE_LIST[@]}"}; do
```

```
        if [ "$_nid" = "${WAN_NODE_ID:-0}" ] || \
           { [ "${WAN_NODE_PER_OP:-0}" = "1" ] && [ "$_nid" -gt "${WAN_NODE_ID:-0}" ]; }; then
            printf '    noeud %-2s %-16s %s\n' "$_nid" "${WAN_IP[$_nid]}" "conteneur lance ici"
        else
            printf '    noeud %-2s %-16s %s\n' "$_nid" "${WAN_IP[$_nid]}" "declare (doit tourner ailleurs)"
        fi
    done
    echo ""
}

# ── La table WAN, batie dans l'ordre ou on la pense ──────────────────────
# D'abord COMBIEN d'operateurs en tout, puis combien tournent ICI en conteneur,
# puis les autres un par un. Les conteneurs prennent les premiers rangs -
# osmo-operator-1 est le noeud 1, -2 le noeud 2 - de sorte que le nom du
# conteneur donne son numero de noeud, sans table a consulter. Les operateurs
# distants (une VM deja demarree) prennent les rangs suivants : on ne demande
# que ce qu'on ne peut pas deduire, leur adresse et leur indicatif.
wan_menu_table() {
    command -v whiptail >/dev/null 2>&1 || return 0
    [ -t 0 ] || return 0
    [ -n "${WAN_NODES:-}" ] && return 0            # deja donne en CLI
    [ "${#OPERATOR_DECLS[@]}" -gt 0 ] && return 0  # deja declare par --operator

    local total cont k ip ind spec=""
    total=$(wt_input "WAN" "Nombre d'operateurs du WAN, tous sites confondus (1-9) :" "3") || return 0
    [[ "$total" =~ ^[1-9]$ ]] || { wt_msg "Nombre invalide : $total"; return 0; }

    cont=$(wt_input "WAN" "Combien tournent ICI, en conteneur ? (0-${total})" "$(( total > 1 ? total - 1 : total
    ↪  ))") || return 0
    [[ "$cont" =~ ^[0-9]$ ]] && [ "$cont" -le "$total" ] \
        || { wt_msg "Entre 0 et ${total}."; return 0; }

    # Les conteneurs : adresse du backbone, indicatif du rang. Rien a demander.
    for k in $(seq 1 "$cont"); do
        spec="${spec}${spec:+ }${k}:$(op_backbone_ip "$k"):$(( k * 11 ))"
    done
    # Les autres : eux seuls ont besoin d'une reponse.
    for k in $(seq $(( cont + 1 )) "$total"); do
        # Defaut : la VM du banc, seul noeud distant du montage courant.
        _defip=""; [ "$k" = 3 ] && _defip="192.168.1.123"
        ip=$(wt_input "Operateur distant ${k}/${total}" \
             "Adresse du noeud ${k} (une VM, une autre machine) :" "$_defip") || return 0
        [ -n "$ip" ] || { wt_msg "Adresse vide : noeud ${k} ignore."; continue; }
        ind=$(wt_input "Operateur distant ${k}/${total}" \
              "Indicatif (prefixe d'appel) du noeud ${k} :" "$(( k * 11 ))") || return 0
        spec="${spec}${spec:+ }${k}:${ip}:${ind:-$(( k * 11 ))}"
    done

    WAN_NODES="$spec"
    OPERATOR_COUNT_HINT="$cont"
    WAN_NODE_ID="${WAN_NODE_ID:-1}"
    [ "$cont" -ge 1 ] && WAN_NODE_PER_OP=1 && WAN_NODE_PER_OP_GIVEN=1
    echo -e "  ${CYAN}Table WAN :${NC} $WAN_NODES"
}

# ── Les questions WAN que la table ne porte pas ──────────────────────────
# Chacune se saute des que l'option correspondante a ete donnee en ligne de
# commande : une invocation scriptee reste muette, une invocation a la main est
# guidee. Sans terminal (CI, cron), on garde les defauts plutot que d'attendre
# une reponse que personne ne donnera.
wan_menu_complements() {
    command -v whiptail >/dev/null 2>&1 || return 0
    [ -t 0 ] || return 0

    if [ -z "${WAN_HUB_IP:-}" ] && [ "${WAN_HUB_NODE_GIVEN:-0}" != "1" ]; then
        local _c
```

```
        _c=$(wt_menu "Inter-STP" "Ou se trouve le hub SS7 (PC 0.0.0) ?" \
            "ip"    "sur une machine dediee, hors table (une VM par exemple)" \
            "ici"   "sur CETTE machine - un conteneur inter-STP sera lance" \
            "noeud" "sur un autre noeud de la table WAN") || return 0
        case "$_c" in
            ip)
                local _ip
                _ip=$(wt_input "Inter-STP" "Adresse du hub (ex: 192.168.1.49) :" "${WAN_HUB_IP:-}") || return 0
                if [ -n "$_ip" ]; then WAN_HUB_IP="$_ip"
                else wt_msg "Adresse vide : le hub restera local."; fi ;;
            noeud)
                local _n
                _n=$(wt_input "Inter-STP" "Numero du noeud qui porte le hub :" "${WAN_HUB_NODE:-1}") || return 0
                [ -n "$_n" ] && WAN_HUB_NODE="$_n" ;;
            *)  WAN_HUB_NODE="$WAN_NODE_ID" ;;
        esac
    fi

    if [ "${WAN_NODE_PER_OP_GIVEN:-0}" != "1" ]; then
        if wt_yesno "Conteneurs" \
"Chaque conteneur doit-il etre un NOEUD a part entiere ?

Oui : conteneur 1 -> noeud ${WAN_NODE_ID}, conteneur 2 -> noeud $((WAN_NODE_ID + 1))...
      (leurs adresses backbone doivent etre joignables des autres noeuds)
Non : ce sont les operateurs du noeud ${WAN_NODE_ID}."; then
            WAN_NODE_PER_OP=1
        fi
    fi
}

# Applique la table (Asterisk, iptables, SMS) une fois les containers debout.
wan_mesh_apply() {
    local n_operators=$1
    local script_path="$START_REPO/network/setup-wan-mesh.sh"
    if [ ! -f "$script_path" ]; then
        echo -e "${RED}[WAN] network/setup-wan-mesh.sh introuvable${NC}"; return 1
    fi
    # ── Le 4e champ de la table : combien d'operateurs porte chaque noeud ────
    # WAN_NOPS n'avait AUCUN producteur dans le depot : la table sortait
    # toujours a trois champs, setup-wan-mesh.sh se repliait sur une supposition
    # et fabriquait - ou supprimait - des trunks au jugee. On l'ecrit ici, ou
    # l'information est certaine : avec --op-is-node chaque noeud EST un
    # conteneur et porte donc un operateur ; sinon, notre noeud les porte tous.
    # wan_nodes_spec n'ecrit le champ que lorsqu'il vaut autre chose que 1, donc
    # une table de conteneurs garde exactement la forme que les autres lecteurs
    # (build-iso.sh, set_stp_ip.sh, les diagnostics) lui connaissent.
    local _k
    if [ "${WAN_NODE_PER_OP:-0}" = "1" ]; then
        for _k in $(seq "${OP_ID_BASE:-1}" "$(( ${OP_ID_BASE:-1} + n_operators - 1 ))"); do
            WAN_NOPS[$(( ${WAN_NODE_ID:-1} + _k - 1 ))]=1
        done
    else
        WAN_NOPS[${WAN_NODE_ID:-1}]="$n_operators"
    fi

    if [ "${WAN_NODE_PER_OP:-0}" = "1" ]; then
        # UN SEUL passage, avec --op-is-node : le script associe alors
        # l'operateur i au noeud i et recalcule SES pairs. Un passage par noeud,
        # comme on le faisait, ecrivait dans TOUS les conteneurs a chaque fois :
        # le dernier gagnait, l'un d'eux se routait vers lui-meme et l'autre
        # n'avait aucun maillage.
        bash "$script_path" --docker --op-is-node \
            --nodes "$(wan_nodes_spec)" --id 1 --ops "$n_operators" \
            || echo -e "${RED}[WAN] setup-wan-mesh.sh a echoue${NC}"
    else
        bash "$script_path" --docker \
```

```
            --nodes "$(wan_nodes_spec)" --id "$WAN_NODE_ID" --ops "$n_operators" \
            || echo -e "${RED}[WAN] setup-wan-mesh.sh a echoue${NC}"
    fi

}

# ── Rendre au trafic sortant son adresse d'origine ──────────────────────
# Docker masque tout ce qui quitte un bridge : vues du LAN, les associations des
# conteneurs semblent toutes venir de l'hote. Ce n'est PAS ce qui empeche un ASP
# de rejoindre son AS : l'inter-STP est en "accept-asp-connections
# dynamic-permitted" (helpers/create_interop.sh) et apparie par ROUTING CONTEXT,
# pas par adresse source - il n'a aucun bloc "match=", contrairement a ce qui
# etait ecrit ici. Preserver l'adresse reste utile pour autre chose : un
# diagnostic qui nomme l'operateur au lieu de montrer trois fois l'hote
# (checks/diag-interstp.sh), et un pare-feu qui peut distinguer les sources.
# network/setup-docker-lan-route.sh pose donc un RETURN en tete de POSTROUTING.
#
# POURQUOI LE REJOUER ICI
# Docker REINSERE ses MASQUERADE en tete a chaque reseau cree - et start-docker.sh en
# cree un par operateur. Le RETURN se retrouve alors DERRIERE eux et ne sert
# plus a rien : "iptables -L" le montre toujours, et pourtant le hub revoit
# l'adresse de l'hote. On le remet donc en premiere position une fois TOUS les
# reseaux crees.
reassert_docker_lan_return() {
    command -v iptables >/dev/null 2>&1 || return 0
    [ "$(id -u)" -eq 0 ] || return 0

    local lan_ip lan_net
    lan_ip="${HOST_LAN_IP:-}"
    [ -n "$lan_ip" ] || lan_ip="$(ip route show default 2>/dev/null | awk '/^default/{print $9; exit}')"
    [ -n "$lan_ip" ] || lan_ip="$(hostname -I 2>/dev/null | awk '{print $1}')"
    [ -n "$lan_ip" ] || return 0
    lan_net="${lan_ip%.*}.0/24"

    # Si LAN et backbone se recouvrent, la regle porterait sur du trafic interne
    # au bridge : on s'abstient plutot que de deviner.
    [ "${lan_net%%/*}" = "${INTER_NET_SUBNET%%/*}" ] && return 0

    # ── LA REGLE RETURN EST RETIREE, PAS POSEE ──────────────────────────
    # [2026-08-26] Elle preservait l'adresse source des conteneurs vers le LAN :
    # l'inter-STP voyait 172.20.0.11 et 172.20.0.12 au lieu de l'hote. C'etait
    # l'intention. Le resultat etait l'inverse : ASP_DOWN des deux cotes et
    # "connect failed (-110)" en boucle.
    #
    # MESURE, plutot que raisonnement. Capture pendant une reconnexion forcee :
    #   wlp0s20f3 Out  172.20.0.12.2910 > 192.168.1.49.2908  [INIT]
    #   wlp0s20f3 Out  192.168.1.49.2908 > 172.20.0.12.2910  [INIT ACK]
    # L'INIT part, le hub repond - et l'INIT ACK n'est JAMAIS recu en entree par
    # l'hote (tcpdump -Q in : rien). Il est adresse a 172.20.0.12, une adresse
    # que l'hote doit ROUTER, et le chemin ne la lui remet pas.
    #
    # Ce n'est PAS le drop de Docker 29 dans la table raw, contrairement a ce
    # qu'on a cru : ses compteurs sont restes a 0 pendant toute la mesure.
    #   ip daddr 172.20.0.12 iifname != "br-..." counter packets 0 bytes 0 drop
    #
    # Sans la regle, le trafic ressort masque en 192.168.1.31 : le hub repond a
    # l'adresse PROPRE de l'hote, qui la recoit, et conntrack rend le paquet au
    # conteneur. Verifie : as-inter AS_ACTIVE sur les deux operateurs, en huit
    # secondes.
    #
    # Et l'on ne perd rien : l'inter-STP ne distingue pas ses noeuds par leur
    # adresse source. Il est en "accept-asp-connections dynamic-permitted"
    # (helpers/create_interop.sh) et les apparie par ROUTING CONTEXT - 1150,
    # 2150, 3150. Deux noeuds derriere la meme adresse gardent des contextes
    # distincts, donc des AS distincts.
    #
```

```
# On RETIRE donc la regle si un lancement precedent l'a laissee.
iptables -t nat -D POSTROUTING -s "$INTER_NET_SUBNET" -d "$lan_net" -j RETURN 2>/dev/null \
    && echo -e "  ${CYAN}[WAN] regle RETURN retiree : les noeuds sortent masques (l'inter-STP les apparie par
    ↪  routing context)${NC}" \
    || true

# ── Le FORWARD, sans quoi le routage ne sert a rien ──────────────────
# Docker pose un FORWARD par defaut a DROP : un paquet du LAN vers le
# backbone (ou l'inverse) est jete avant d'atteindre le conteneur, sans
# refus ni trace - cote VM, on ne voit qu'un silence. DOCKER-USER est
# consultee AVANT les regles de docker et il ne la reecrit pas.
# network/setup-docker-lan-route.sh pose ces deux memes regles, mais aucun
# chemin automatique ne l'appelle : elles n'existaient donc sur le banc que
# si on avait pense a lancer ce script a la main. On les pose ici, ou la
# fonction est deja rejouee apres chaque reseau cree.
iptables -N DOCKER-USER 2>/dev/null || true
local _spec
for _spec in "-s ${lan_net} -d ${INTER_NET_SUBNET}" "-s ${INTER_NET_SUBNET} -d ${lan_net}"; do
    # shellcheck disable=SC2086
    if iptables -C DOCKER-USER $_spec -j ACCEPT 2>/dev/null; then
        continue
    fi
    # shellcheck disable=SC2086
    if iptables -I DOCKER-USER 1 $_spec -j ACCEPT 2>/dev/null; then
        echo -e "  ${CYAN}[WAN] FORWARD autorise (${_spec})${NC}"
    fi
done

# ── Les traductions DEJA etablies ────────────────────────────────────
# La regle ne vaut que pour les nouvelles associations. Celles qui ont ete
# masquees avant elle gardent leur traduction tant qu'elles vivent : le hub
# continue de les voir sous l'adresse de l'hote, et l'on croit la regle
# inoperante alors qu'elle est bien la. On efface donc ces entrees - les
# piles se reconnectent, cette fois avec leur vraie adresse.
#
# SAUF quand docker 29 a pose son DROP. Sa table "ip raw" jette, en
# prerouting, tout paquet entrant vers le backbone qui n'arrive pas par le
# bridge : une association qui repart avec sa vraie adresse source ne recoit
# alors plus aucune reponse et reste en COOKIE_WAIT. Purger conntrack dans
# cet etat coupe des ASP qui ne tiennent QUE par le masquage, pour les faire
# renaitre muets - c'est exactement la panne qu'on croit corriger.
local _nft_drop=0
if command -v nft >/dev/null 2>&1; then
    if nft list table ip raw 2>/dev/null \
       | grep -F "${INTER_NET_SUBNET%.*}." | grep -q 'drop'; then
        _nft_drop=1
    fi
fi
if [ "$_nft_drop" = "1" ]; then
    echo -e "  ${YELLOW}[WAN] DROP nft de docker actif sur ${INTER_NET_SUBNET} :${NC}"
    echo -e "  ${YELLOW}      purge conntrack SAUTEE (elle couperait les ASP attaches).${NC}"
    echo -e "  ${CYAN}      Backbone en mode routed pour le lever : OSMO_DOCKER_ROUTED=1${NC}"
elif command -v conntrack >/dev/null 2>&1; then
    # BORNEE a la destination LAN. Sans -d, la meme commande effacait aussi
    # les flux 172.20.0.x <-> 172.20.0.x, c'est-a-dire tout le SS7
    # intra-backbone : les ASP du hub local tombaient d'un seul coup, et le
    # message parlait pourtant de "traductions heritees".
    local _purged
    _purged="$(conntrack -D -s "$INTER_NET_SUBNET" -d "$lan_net" 2>/dev/null | grep -c . || true)"
    echo -e "  ${CYAN}[WAN] ${_purged} traduction(s) heritee(s) effacee(s) vers ${lan_net}${NC}"
else
    # Le cas de l'hote du banc. Le bloc etait saute sans un mot : on croyait
    # les traductions effacees alors qu'elles vivaient encore, et le hub
    # continuait de voir l'adresse de l'hote sans que rien ne l'explique.
    echo -e "  ${YELLOW}[WAN] conntrack absent : les associations deja masquees gardent${NC}"
    echo -e "  ${YELLOW}      leur traduction jusqu'a leur reconnexion (apt install conntrack)${NC}"
```

```
    fi
}

# ============================================================================
# PoC QEMU - entree par defaut du premier menu
# ============================================================================
start_qemu_poc() {
    echo -e "${GREEN}PoC QEMU Calypso - bring-up minimal + qosmo-grgsm/run.sh...${NC}"
    QEMU_POC=1
    start_bridge_mode
}

# ============================================================================
# Mode virtual (multi-operateurs SS7 - ex-bridge)
# ============================================================================
start_bridge_mode() {
    # Ce mode CREE les conteneurs : le segment prive se derive de l operateur,
    # pas du noeud (voir _osmo_priv_index).
    export OSMO_PRIV_BY_OP=1
    local n_operators
    declare -A OP_MCC OP_MNC OP_NAME OP_MS

    if [ "${QEMU_POC:-0}" = "1" ]; then
        # -- PoC QEMU ------------------------------------------------------
        n_operators=1
        # Meme base que les boucles : sans ca, un OP_ID_BASE=2 ferait chercher
        # OP_MCC[2] alors que tout est pose en [1] - tableaux vides et
        # operateur sans MCC, sans que rien ne le signale.
        _b="${OP_ID_BASE:-1}"
        OP_MCC[$_b]="001"; OP_MNC[$_b]="01"; OP_NAME[$_b]="OsmoQEMU"; OP_MS[$_b]=2
        # PoC a un seul operateur : le PLMN de reference, celui des IMSI du
        # depot. On ne le derive pas du noeud, il n'y a pas de WAN ici.
        WAN_ENABLED="false"
        # --wan reste valable en PoC QEMU : un noeud = un operateur, c'est
        # exactement la maquette "N machines qui s'appellent".
        [ "${WAN_MESH:-0}" = "1" ] && wan_mesh_configure 1
        # BRIDGE_NO_PROCESS=0 : la radio et le mobile demarrent avec le coeur
        # (voir le commentaire "NO-ATTACH, PLUS NO-PROCESS" plus bas).
        PHY_MODE="faketrx"; BRIDGE_NO_PROCESS="${BRIDGE_NO_PROCESS:-0}"; BRIDGE_QEMU=1
        ENCRYPTION="${ENCRYPTION:-a5 1}"
        echo -e "  ${CYAN}[PoC QEMU] 1 operateur · no-process + qosmo/run.sh · ${ENCRYPTION}${NC}"
    else
        # -- Mode interactif -----------------------------------------------
        # Deja repondu dans le menu WAN (« combien tournent ICI ») : on ne
        # repose pas la question, deux reponses divergentes donneraient une
        # table et un nombre de conteneurs qui ne se correspondent plus.
        if [ -n "${OPERATOR_COUNT_HINT:-}" ]; then
            n_operators="$OPERATOR_COUNT_HINT"
            echo -e "  ${CYAN}Conteneurs a lancer : ${n_operators} (--operators / menu WAN)${NC}"
        else
            # 3 par defaut : c'est la taille du banc (deux conteneurs et une
            # VM). Un defaut qui correspond au montage courant evite la moitie
            # des saisies, et surtout evite qu'un oubli produise une table plus
            # courte que la realite - un noeud absent de la table n'existe pour
            # personne, et son absence ne se voit qu'a l'appel qui n'aboutit pas.
            n_operators=$(wt_input "Operateurs" "Nombre d'operateurs (1-36) :" "3") || exit 1
        fi
        n_operators=${n_operators:-2}
        if ! [[ "$n_operators" =~ ^[0-9]+$ ]] || [ "$n_operators" -lt 1 ] || [ "$n_operators" -gt 36 ]; then
            wt_msg "Nombre invalide (1-36)."; exit 1
        fi

        # -- LE PLAN DE NOEUDS SE DECIDE AVANT LE PLMN ---------------------
        # Ce bloc etait 50 lignes plus bas, APRES le menu des operateurs. Or
        # c'est lui qui repond a "chaque conteneur est-il un noeud a part
        # entiere ?", donc lui qui fixe WAN_NODE_PER_OP - et le MCC vaut le
```

```
# numero de NOEUD. Le menu calculait donc les PLMN alors que la reponse
# n'etait pas encore connue : tout le monde recevait le MCC du noeud 1.
# Constate sur le banc - "Operateur 2 : MCC=001 MNC=02" annonce juste au-
# dessus de "STP PC 1.21.2", c'est-a-dire noeud 2. Deux plans dans le
# meme ecran, le point code juste et le PLMN faux.
# Les questions de WAN passent donc devant. Elles ne dependent que du
# nombre d'operateurs, connu ci-dessus.
# ── WAN ──────────────────────────────────────────────────────────
if [ "${WAN_MESH:-0}" = "1" ]; then
    wan_mesh_configure "$n_operators"
elif wt_yesno "WAN Interop" "Activer WAN vers un serveur distant (2 serveurs, prefixe 66) ?"; then
    WAN_ENABLED="true"
    local auto_ip=""
    auto_ip=$(ip route get 1.1.1.1 2>/dev/null | awk '{for(i=1;i<=NF;i++) if($i=="src") print $(i+1)}' |
     ↪  head -1) || true
    [ -z "$auto_ip" ] && auto_ip=$(hostname -I 2>/dev/null | awk '{print $1}') || true
    local wan_local wan_remote wan_nremote wan_pfx
    wan_local=$(wt_input "WAN" "IP publique locale :" "$auto_ip") || exit 1
    WAN_LOCAL_IP="${wan_local:-$auto_ip}"
    [ -z "$WAN_LOCAL_IP" ] && WAN_ENABLED="false"
    if [ "$WAN_ENABLED" = "true" ]; then
        wan_remote=$(wt_input "WAN" "IP publique distante :" "") || exit 1
        WAN_REMOTE_IP="$wan_remote"
        [ -z "$WAN_REMOTE_IP" ] && WAN_ENABLED="false"
    fi
    if [ "$WAN_ENABLED" = "true" ]; then
        wan_nremote=$(wt_input "WAN" "Nb operateurs distants :" "$n_operators") || exit 1
        WAN_N_REMOTE="${wan_nremote:-$n_operators}"
        wan_pfx=$(wt_input "WAN" "Prefixe WAN :" "$WAN_PREFIX") || exit 1
        WAN_PREFIX="${wan_pfx:-$WAN_PREFIX}"
        echo -e "  ${GREEN}WAN: ${WAN_LOCAL_IP} ↔ ${WAN_REMOTE_IP} (local=${n_operators}
         ↪  remote=${WAN_N_REMOTE} prefix=${WAN_PREFIX})${NC}"
    fi
fi

local use_defaults="N"
wt_yesno "Valeurs par defaut" "Valeurs par defaut pour tous (MCC=001/002/... MNC=01/02/...) ?" &&
 ↪  use_defaults="O"

local same_ms_all="N"
if [ "$use_defaults" != "O" ]; then
    wt_yesno "MS" "Meme nombre de MS pour tous les operateurs ?" && same_ms_all="O"
fi

declare -A OP_MCC OP_MNC OP_NAME OP_MS

if [ "$use_defaults" = "O" ]; then
    local common_ms
    common_ms=$(wt_input "MS" "MS par operateur (1-64) :" "8") || exit 1
    common_ms=${common_ms:-8}
    if ! [[ "$common_ms" =~ ^[0-9]+$ ]] || [ "$common_ms" -lt 1 ] || [ "$common_ms" -gt 64 ]; then
     ↪  common_ms=8; fi
    for i in $(seq "${OP_ID_BASE:-1}" "$(( ${OP_ID_BASE:-1} + n_operators - 1 ))"); do
        # Un PLMN par operateur, sur les DEUX composantes. Le MCC
        # etait fige a 001 pour tout le monde : seul le MNC separait
        # les reseaux, et la moindre valeur ecrite en dur ailleurs
        # (les IMSI des MS l'etaient) retombait sur un PLMN plausible
        # mais etranger. 001/002/003 rend la confusion visible.
        OP_MCC[$i]=$(op_mcc "$i"); OP_MNC[$i]=$(op_mnc "$i")
        OP_NAME[$i]="$(op_default_name "$i")"; OP_MS[$i]=$common_ms
    done
else
    if [ "$same_ms_all" = "O" ]; then
        local common_ms
        common_ms=$(wt_input "MS" "MS par operateur (1-64) :" "8") || exit 1
        common_ms=${common_ms:-8}
```

```
            if ! [[ "$common_ms" =~ ^[0-9]+$ ]] || [ "$common_ms" -lt 1 ] || [ "$common_ms" -gt 64 ]; then
             ↪  common_ms=8; fi
            for i in $(seq "${OP_ID_BASE:-1}" "$(( ${OP_ID_BASE:-1} + n_operators - 1 ))"); do
                local mcc mnc name dmcc dmnc dname
                dmcc=$(op_mcc "$i"); dmnc=$(op_mnc "$i"); dname="$(op_default_name "$i")"
               mcc=$(wt_input "Operateur ${i}" "MCC (pays = noeud $(op_plmn_node "$i"), $(op_country "$i")) :"
                ↪  "$dmcc") || exit 1; OP_MCC[$i]=${mcc:-$dmcc}
                mnc=$(wt_input "Operateur ${i}" "MNC :" "$dmnc") || exit 1; OP_MNC[$i]=${mnc:-$dmnc}
                name=$(wt_input "Operateur ${i}" "Nom :" "$dname") || exit 1; OP_NAME[$i]=${name:-$dname}
                OP_MS[$i]=$common_ms
            done
        else
            for i in $(seq "${OP_ID_BASE:-1}" "$(( ${OP_ID_BASE:-1} + n_operators - 1 ))"); do
                local mcc mnc name n_ms dmcc dmnc dname
                dmcc=$(op_mcc "$i"); dmnc=$(op_mnc "$i"); dname="$(op_default_name "$i")"
               mcc=$(wt_input "Operateur ${i}" "MCC (pays = noeud $(op_plmn_node "$i"), $(op_country "$i")) :"
                ↪  "$dmcc") || exit 1; OP_MCC[$i]=${mcc:-$dmcc}
                mnc=$(wt_input "Operateur ${i}" "MNC :" "$dmnc") || exit 1; OP_MNC[$i]=${mnc:-$dmnc}
                name=$(wt_input "Operateur ${i}" "Nom :" "$dname") || exit 1; OP_NAME[$i]=${name:-$dname}
                # 2 par defaut, pas 1 : c est le minimum pour une radio
                # MIXTE - un faketrx et un QEMU Calypso, comme le natif.
                # A 1, l operateur n avait qu un seul MS et le mode hybride
                # n avait plus rien a repartir.
                n_ms=$(wt_input "Operateur ${i}" "Nombre de MS (1-64) :" "2") || exit 1; OP_MS[$i]=${n_ms:-2}
               if ! [[ "${OP_MS[$i]}" =~ ^[0-9]+$ ]] || [ "${OP_MS[$i]}" -lt 1 ] || [ "${OP_MS[$i]}" -gt 64 ];
                ↪  then OP_MS[$i]=1; fi
            done
        fi
    fi


    # ── RAN / Encryption ──────────────────────────────────────────
    # BRIDGE_NO_PROCESS=0 : la radio et le mobile demarrent avec le coeur
    # (voir le commentaire "NO-ATTACH, PLUS NO-PROCESS" plus bas).
    PHY_MODE="faketrx"; BRIDGE_NO_PROCESS="${BRIDGE_NO_PROCESS:-0}"; BRIDGE_QEMU=1
    ENCRYPTION="${ENCRYPTION:-a5 1}"
    if [ -n "$HANDOFF_MODE_FROM_ENV" ]; then
        # [2026-08-31] L APPELANT A IMPOSE LE MODE : on ne lui repose pas la
        # question. Sans cette branche, seul --wan pouvait choisir l hybride
        # et tout autre lancement retombait sur choose_ran_host - un menu.
        # En non interactif ce menu s auto-repondait par sa PREMIERE entree,
        # donc "full-grgsm" : un HANDOFF_MODE=faketrx-qemu passe en
        # environnement etait lu (HANDOFF_MODE_FROM_ENV=1, l.137) puis
        # ignore ici. On ne pouvait pas obtenir la radio mixte hors WAN.
        echo -e "  ${GREEN}[RAN] mode impose par l appelant : ${CYAN}${HANDOFF_MODE}${NC}"
    elif [ "${WAN_MESH:-0}" = "1" ]; then
        # Un noeud de WAN est une maquette complete : il lui faut une radio
        # des DEUX types pour que l'appel inter-noeud soit vraiment porte de
        # bout en bout. D'ou l'hybride par defaut - un faketrx + un QEMU
        # Calypso - au lieu du menu. HANDOFF_MODE=... en prefixe le change.
        HANDOFF_MODE="faketrx-qemu"; HANDOFF_QEMU_CHOICE="faketrx-qemu"
        echo -e "  ${GREEN}[RAN] WAN → hybride par defaut : ${CYAN}1 faketrx + 1 QEMU Calypso${NC}"
    else
        choose_ran_host
    fi
fi

# ── Detection IP hote ─────────────────────────────────────────────
HOST_IP=$(ip route get 1.1.1.1 2>/dev/null \
    | awk '{for(i=1;i<=NF;i++) if($i=="src") print $(i+1)}' | head -1) || true
if [ -z "$HOST_IP" ]; then
    HOST_IP=$(hostname -I 2>/dev/null | awk '{print $1}') || true
fi
HOST_IP="${HOST_IP:-127.0.0.1}"
REAL_UID=$(id -u "${SUDO_USER:-$(logname 2>/dev/null || echo root)}")
echo -e "${GREEN}IP hote : ${CYAN}${HOST_IP}${NC}  (Linphone)${NC}"
```

```
echo ""

# ── Ou joindre le hub SS7, vu d'ici ──────────────────────────────
# Le hub est UNIQUE pour tout le WAN. Deux cas, et le mauvais donne un ASP
# qui ne s'attache jamais :
#   ce noeud heberge le hub  → les conteneurs le joignent en interne (172.20.0.10)
#   un autre noeud l'heberge → ils sortent sur le WAN, vers SON adresse
WAN_STP_TARGET="$INTER_STP_IP"
HUB_IS_LOCAL=1
if [ -n "${WAN_HUB_IP:-}" ]; then
    # Hub designe par son adresse : il n'est pas un noeud, et rien ne se
    # lance ici. C'est le cas d'une VM inter-STP dediee.
    WAN_STP_TARGET="$WAN_HUB_IP"; HUB_IS_LOCAL=0
elif [ "${WAN_MESH:-0}" = "1" ] && [ "${WAN_NODE_ID}" != "${WAN_HUB_NODE}" ]; then
    WAN_STP_TARGET="${WAN_IP[$WAN_HUB_NODE]:-}"; HUB_IS_LOCAL=0
    if [ -z "$WAN_STP_TARGET" ]; then
        echo -e "${RED}[SS7] le noeud ${WAN_HUB_NODE} (hub) n'est pas dans la table WAN${NC}" >&2
        echo -e "${RED}      (ou designez-le par son adresse : --hub-ip ADRESSE)${NC}" >&2
        exit 1
    fi
fi

# ── Reseau backbone ──────────────────────────────────────────────
# OSMO_DOCKER_ROUTED=1 : bridge en mode "routed" (docker >= 28). Docker
# cesse alors de masquer ce qui sort du backbone, et surtout n'installe pas
# dans sa table "ip raw" le DROP qui jette, en prerouting, tout paquet
# entrant vers 172.20.0.x n'arrivant pas par le bridge. Avec ce DROP, un
# conteneur joint depuis le LAN a sa VRAIE adresse ne recoit jamais de
# reponse : son association SCTP reste en COOKIE_WAIT, sans un mot dans les
# journaux. C'est le seul reglage qui leve les deux a la fois.
# Optionnel et non fatal : docker < 28 ne connait pas l'option et refuse la
# creation, on retombe alors en silence sur le reseau ordinaire (masque),
# celui avec lequel le banc a toujours tourne.
if docker network inspect "$INTER_NET" &>/dev/null; then
    echo -e "  ${CYAN}Reseau backbone ${INTER_NET} deja present${NC}"
else
    if [ "${OSMO_DOCKER_ROUTED:-0}" = "1" ] \
       && docker network create --subnet="$INTER_NET_SUBNET" --gateway="$INTER_NET_GATEWAY" \
            -o com.docker.network.bridge.gateway_mode_ipv4=routed "$INTER_NET" &>/dev/null; then
        echo -e "  ${GREEN}✓ Reseau backbone ${INTER_NET} (${INTER_NET_SUBNET}) cree - mode routed${NC}"
    elif docker network create --subnet="$INTER_NET_SUBNET" --gateway="$INTER_NET_GATEWAY" "$INTER_NET"
     ↪  &>/dev/null; then
        [ "${OSMO_DOCKER_ROUTED:-0}" = "1" ] \
            && echo -e "  ${YELLOW}  (mode routed refuse par ce docker - reseau ordinaire)${NC}"
        echo -e "  ${GREEN}✓ Reseau backbone ${INTER_NET} (${INTER_NET_SUBNET}) cree${NC}"
    else
        echo -e "  ${RED}✗ Echec creation reseau backbone ${INTER_NET} (${INTER_NET_SUBNET})${NC}"; exit 1
    fi
fi

# ── SMS Routing ──────────────────────────────────────────────────
SMS_ROUTING_DIR=$(mktemp -d)
if declare -f sms_routing_generate_all > /dev/null 2>&1; then
    local _ms_counts=()
    for i in $(seq "${OP_ID_BASE:-1}" "$(( ${OP_ID_BASE:-1} + n_operators - 1 ))"); do
     ↪  _ms_counts+=("${OP_MS[$i]}"); done
    sms_routing_generate_all "$n_operators" "$SMS_ROUTING_DIR" "${_ms_counts[@]}"
    sms_routing_summary "$n_operators" "${_ms_counts[@]}"
fi

# ── Inter-STP ────────────────────────────────────────────────────
if [ "$HUB_IS_LOCAL" = "1" ]; then
    start_inter_stp "$n_operators"
    wait_inter_stp_ready "$n_operators"
else
    echo -e "${CYAN}[SS7] hub porte par le noeud ${WAN_HUB_NODE} (${WAN_STP_TARGET}) - pas de hub ici.${NC}"
```

```
    echo -e "${CYAN}      Les ASP de ce noeud s'y attacheront en M3UA/SCTP 2908.${NC}"
fi

# ── HLR subscribers ──────────────────────────────────────────────
local all_subscribers_file
all_subscribers_file=$(mktemp)
echo "# op_id:ms_idx:imsi:msisdn:ki" > "$all_subscribers_file"
# ── L'annuaire : un etat civil pour chaque IMSI ──────────────────
# Un banc ne rend que des numeros a 15 chiffres, et l'on finit par lire
# "001010001000002" comme un mot de passe : impossible de dire de tete a
# quel operateur, a quel pays, a quel telephone il appartient. Cette table
# met un nom et une adresse en face de chaque abonne. Elle est FABRIQUEE -
# personnages et rues n'existent pas - mais elle n'invente pas l'IMSI :
# c'est exactement celui qui part au HLR trois lignes plus bas, dans la
# meme boucle, a partir de la meme variable.
# Les fiches sont tirees dans data/noms.txt et data/adresses.txt, melanges
# au chargement. On les parcourt EN ORDRE a partir de la liste melangee
# plutot que de retirer au sort a chaque abonne : deux abonnes du meme banc
# ne peuvent alors pas porter le meme nom tant que le fichier n'est pas
# epuise - et deux homonymes dans une trace, c'est une heure perdue.
local ANNUAIRE_FILE="${OSMO_ANNUAIRE:-/var/tmp/osmo-annuaire.csv}"
# ── L'annuaire du SS7 : le meme service, un etage plus bas ───────
# L'annuaire d'abonnes met un nom sur un MSISDN. Celui-ci met un nom sur un
# POINT CODE. On lit "1.21.2" dans une trace M3UA sans savoir de quelle
# machine il s'agit, ni quel operateur elle porte, ni vers quel hub elle
# s'attache - et les trois se deduisent d'un plan qui n'est ecrit nulle
# part ailleurs que dans la tete de celui qui a lance le banc.
# Il est produit par la MEME boucle que les configurations : ce n'est pas
# une documentation a cote, c'est le releve de ce qui vient d'etre ecrit.
local ANNUAIRE_SS7="${OSMO_ANNUAIRE_SS7:-/var/tmp/osmo-annuaire-ss7.csv}"
printf
 ↪  'noeud;operateur;pays;mcc;mnc;pc_msc;pc_stp;pc_bsc;rctx_msc;rctx_stp;rctx_bsc;rctx_inter;hub;as_inter;asp_inter;backbon
 ↪  \
    > "$ANNUAIRE_SS7"
local _pick=0
printf 'imsi;msisdn;operateur;pays;mcc;mnc;nom;adresse;ki\n' > "$ANNUAIRE_FILE"
for op_id in $(seq "${OP_ID_BASE:-1}" "$(( ${OP_ID_BASE:-1} + n_operators - 1 ))"); do
    local mcc="${OP_MCC[$op_id]}" mnc="${OP_MNC[$op_id]}" n_ms="${OP_MS[$op_id]}"
    local _pays; _pays="$(op_country "$op_id")"
    for ms_idx in $(seq 1 "$n_ms"); do
        local msin; msin=$(printf '%04d%06d' "${op_id}" "${ms_idx}")
        local imsi="${mcc}${mnc}${msin}"
        # MSISDN a SIX chiffres : <noeud>00<operateur><rang>.
        # L'ancien plan, op * 10000 + rang, commencait a 10001 - cinq
        # chiffres dont le premier EST le numero d'operateur, ce qui melait
        # le plan d'abonnes au plan de composition (les services du dialplan
        # sont a 100, 200, 500, 600, 700) et obligeait chaque motif local a
        # dependre de l'operateur. Le prefixe 600 qui l'a remplace etait
        # commun a TOUTES les machines : deux noeuds y revendiquaient les
        # memes numeros. Le premier chiffre porte donc le noeud, et le
        # routage se lit dans le numero : 100101, 100102, 200101...
        local msisdn; msisdn=$(osmo_msisdn "$(osmo_node_id)" "$op_id" "$ms_idx")
        local ki; ki=$(printf '00112233445566778899aabbccdd%02x%02x' "${ms_idx}" "${op_id}")
        echo "${op_id}:${ms_idx}:${imsi}:${msisdn}:${ki}" >> "$all_subscribers_file"
        printf '%s;%s;%s;%s;%s;%s;%s;%s %s;%s\n' \
            "$imsi" "$msisdn" "${OP_NAME[$op_id]}" "$_pays" "$mcc" "$mnc" \
            "${POOL_NOMS[$(( _pick % ${#POOL_NOMS[@]} ))]}" \
            "$(( RANDOM % 97 + 1 ))" \
            "${POOL_ADR[$(( _pick % ${#POOL_ADR[@]} ))]}" \
            "$ki" >> "$ANNUAIRE_FILE"
        _pick=$(( _pick + 1 ))
    done
done
local total_subs; total_subs=$(( $(wc -l < "$all_subscribers_file") - 1 ))
echo -e "${GREEN}Abonnes total : ${total_subs}${NC}"
# Aperçu : les premieres fiches, puis le chemin du fichier complet. Une
```

```
# table de 8 MS par operateur noierait le reste du demarrage.
awk -F';' 'NR==1 {next}
           NR<=9 {printf "  %-16s %-6s %-16s %-12s %-22s %s\n", $1,$2,$3,$4,$7,$8}
           END {if (NR>9) printf "  ... et %d autres\n", NR-9}' "$ANNUAIRE_FILE"
echo -e "  ${CYAN}Annuaire complet : ${ANNUAIRE_FILE}${NC}  (aussi /etc/osmocom/annuaire.csv dans chaque
 ↪  conteneur)"
echo ""

# ── Table rase sur la serie d'operateurs ──────────────────────────────
# Le "docker rm -f" plus bas ne vise que le conteneur de MEME NOM. Un
# reliquat d'un lancement precedent - trois operateurs hier, deux
# aujourd'hui, ou une serie interrompue en cours de route - garde ses
# publications de ports et fait echouer la creation du suivant :
#     Bind for 0.0.0.0:5082 failed: port is already allocated
# Le message nomme le port, jamais le conteneur qui le tient : on cherche
# longtemps. Cette fonction RECREE toute la serie, on efface donc d'abord.
local _stale
_stale="$(docker ps -aq --filter "name=^osmo-operator-" 2>/dev/null)"
if [ -n "$_stale" ]; then
    echo -e "  ${CYAN}Nettoyage : $(echo "$_stale" | wc -l) conteneur(s) operateur d'un lancement
     ↪  precedent${NC}"
    # shellcheck disable=SC2086
    docker rm -f $_stale >/dev/null 2>&1 || true
fi

# ── Demarrage sequentiel des operateurs ───────────────────────────────
for i in $(seq "${OP_ID_BASE:-1}" "$(( ${OP_ID_BASE:-1} + n_operators - 1 ))"); do
    local container_name net_name subnet gateway container_ip inter_local_ip rctx_inter
    local n_groups last_group_ip

    container_name=$(op_container "$i")
    net_name="gsm-net-op${i}"
    subnet=$(op_private_net "$i")
    gateway=$(op_private_gw "$i")
    container_ip=$(op_private_ip "$i")
    inter_local_ip=$(op_backbone_ip "$i")
    rctx_inter=$(op_rctx_inter "$i")
    n_groups=$(( (${OP_MS[$i]} + 7) / 8 ))
    last_group_ip="127.0.0.${n_groups}"

    echo -e "${CYAN}── Operateur ${i} : ${OP_NAME[$i]} ($(op_country "$i"), MCC=${OP_MCC[$i]}
     ↪  MNC=${OP_MNC[$i]}) ──${NC}"
    echo -e "  Backbone   : ${CYAN}${inter_local_ip}${NC}  Prive : ${CYAN}${container_ip}${NC}"
    local _node_i="$WAN_NODE_ID" _op_i="$i"
    if [ "${WAN_NODE_PER_OP:-0}" = "1" ]; then
        _node_i=$(( WAN_NODE_ID + i - 1 ))
        _op_i=1
        # Le numero de noeud vit dans 1..9 : au-dela, le point code ecrit
        # ici (1.<noeud><op>.<role>) sort du format ITU 3-8-3, aucun AS
        # correspondant n'existe au hub (helpers/create_interop.sh borne a
        # 9 noeuds), et network/set-node-id.sh comme start-native.sh
        # refusent l'argument. Mieux vaut le dire ici, avant de fabriquer
        # une identite SS7 que personne ne connaitra.
        if [ "$_node_i" -gt 9 ] || [ "$_node_i" -lt 1 ]; then
            echo -e "${RED}[WAN] --node-per-op : l'operateur ${i} tomberait sur le noeud ${_node_i}${NC}" >&2
           echo -e "${RED}      Les numeros de noeud vont de 1 a 9. Avec --wan-id ${WAN_NODE_ID},"' '"${NC}" >&2
            echo -e "${RED}      le maximum est $(( 10 - WAN_NODE_ID )) operateur(s) : baissez --wan-id,"'
             ↪  '"${NC}" >&2
            echo -e "${RED}      reduisez --operators, ou retirez --node-per-op.${NC}" >&2
            exit 1
        fi
    fi
    local _pc_mid="$i" _rctx_inter=""
    if [ "${WAN_MESH:-0}" = "1" ]; then
        _pc_mid="${_node_i}${_op_i}"
        _rctx_inter=$(( _node_i * 1000 + _op_i * 100 + 50 ))
```

```
fi

# L'identite AFFICHEE est celle qui sera ECRITE. Cette ligne montrait
# "1.<op>.2" en dur et le routing context du plan local : elle annoncait
# donc 1.1.2/150 alors que la config recevait 1.21.2/2150. On cherche
# longtemps une panne quand le journal dit le contraire du fichier.
echo -e "  STP PC     : 1.${_pc_mid}.2  RCTX : ${_rctx_inter:-$rctx_inter}  MS : ${CYAN}${OP_MS[$i]}${NC}"

# Le releve SS7 de CET operateur, avec les valeurs qui partent dans sa
# configuration deux lignes plus bas - pas une reconstitution.
printf '%s;%s;%s;%s;%s;1.%s.1;1.%s.2;1.%s.3;%s;%s;%s;%s;%s;as-n%sop%s;asp-n%sop%s;%s;%s\n' \
    "$_node_i" "${OP_NAME[$i]}" "$(op_country "$i")" \
    "${OP_MCC[$i]}" "${OP_MNC[$i]}" \
    "$_pc_mid" "$_pc_mid" "$_pc_mid" \
    "$(op_rctx_msc "$i")" "$(op_rctx_stp "$i")" "$(op_rctx_bsc "$i")" \
    "${_rctx_inter:-$rctx_inter}" "${WAN_STP_TARGET:-127.0.0.1}" \
    "$_node_i" "$_op_i" "$_node_i" "$_op_i" \
    "$inter_local_ip" "$container_ip" >> "$ANNUAIRE_SS7"

if docker network inspect "$net_name" &>/dev/null; then
    echo -e "  Reseau     : ${CYAN}${net_name}${NC} (deja present)"
elif docker network create --subnet="$subnet" --gateway="$gateway" "$net_name" &>/dev/null; then
    echo -e "  Reseau     : ${GREEN}✓ ${net_name} (${subnet}) cree${NC}"
else
    echo -e "  ${RED}✗ Echec creation reseau ${net_name} (${subnet})${NC}"; exit 1
fi

# ── La regle de non-masquage, AVANT que la pile ne se connecte ────────
# Une decision de NAT se prend sur le PREMIER paquet d'une association :
# conntrack la conserve ensuite pour toute sa duree. Poser la regle en
# fin de lancement arrivait donc trop tard - l'ASP s'etait deja attache,
# masque, et le hub le voyait sous l'adresse de l'hote jusqu'a sa
# prochaine reconnexion. Deux operateurs y devenaient indiscernables.
#
# Ici, elle est reaffirmee apres CHAQUE reseau cree (docker reinsere ses
# MASQUERADE en tete a chacun) et donc avant tout demarrage de pile.
reassert_docker_lan_return >/dev/null

local tmpdir
tmpdir=$(mktemp -d)
# Plan de point codes. Hors WAN : 1.<op>.<role>, comme toujours. En WAN :
# le numero de noeud entre dedans, sans quoi deux machines presentent au
# hub des adresses SS7 identiques - routage faux, et silencieux.
# Seul RCTX_INTER est surcharge : c'est le seul routing context que le
# hub voit. Ceux du MSC et du BSC restent internes au conteneur.
#
# UN NOEUD PAR CONTENEUR (--node-per-op)
# Par defaut, la machine EST le noeud et ses conteneurs en sont les
# operateurs : 1.<noeud><op>.<role>. C'est juste quand une machine
# heberge un operateur complet.
#
# Mais des que les conteneurs sont joignables chacun a SON adresse
# (172.20.0.11, .12... routees depuis le LAN), ce ne sont plus des
# operateurs d'un meme site : ce sont des noeuds a part entiere, au meme
# titre qu'une VM. Les traiter en operateurs obligeait alors a inscrire
# la MACHINE dans la table WAN - une entree qui ne correspond a aucun
# equipement, et qui recevait pourtant un indicatif et des routes.
#
# Avec --node-per-op, le conteneur i devient le noeud (base + i - 1),
# porte l'operateur 1 de ce noeud, et son point code suit : 1.<n>1.<role>.
RCTX_INTER_OVERRIDE="$_rctx_inter" \
apply_config_templates "$tmpdir" \
    "$container_ip" "$gateway" \
    "$i" "1.${_pc_mid}.1" "1.${_pc_mid}.2" "1.${_pc_mid}.3" \
    "${OP_MCC[$i]}" "${OP_MNC[$i]}" "${OP_NAME[$i]}" \
    "$WAN_STP_TARGET" "no shutdown" \
```

```
    "$n_operators"

# L'annuaire voyage avec la pile : c'est dans le conteneur qu'on lit
# un IMSI dans un journal, pas sur l'hote.
[ -f "$ANNUAIRE_FILE" ] && cp "$ANNUAIRE_FILE" "$tmpdir/osmocom/annuaire.csv"
[ -f "$ANNUAIRE_SS7" ]  && cp "$ANNUAIRE_SS7"  "$tmpdir/osmocom/annuaire-ss7.csv"

# ── Les fiches du dialplan ──────────────────────────────────
# Asterisk ne sait pas lire un CSV. On lui ecrit donc les memes fiches
# sous la forme qu'il comprend : une extension par MSISDN dans le
# contexte sub-annuaire (voir configs/extensions.conf). L'appel arrive
# alors sur le combine avec un nom, et non avec cinq chiffres.
{
    printf '; annuaire.conf - genere par start-docker.sh, ne pas editer a la main.\n'
    printf '; Une fiche par abonne du banc. Le repli _X. est dans extensions.conf.\n'
    printf '[sub-annuaire]\n'
    awk -F';' 'NR>1 && $2 != "" {
        gsub(/[^A-Za-z0-9 ._-]/, "", $7); gsub(/[^A-Za-z0-9 ._-]/, "", $3)
        gsub(/[^A-Za-z0-9 ._-]/, "", $4); gsub(/[^A-Za-z0-9 .,_-]/, "", $8)
        printf "exten => %s,1,Set(CALLERID(name)=%s)\n", $2, $7
        printf " same => n,Set(ANNU_OP=%s)\n", $3
        printf " same => n,Set(ANNU_PAYS=%s)\n", $4
        printf " same => n,Set(ANNU_ADR=%s)\n", $8
        printf " same => n,Set(ANNU_IMSI=%s)\n", $1
        printf " same => n,Set(CDR(userfield)=%s / %s / %s)\n", $7, $3, $4
        printf " same => n,NoOp(=== ANNUAIRE %s = %s, %s (%s) ===)\n", $2, $7, $3, $4
        printf " same => n,Return()\n"
    }' "$ANNUAIRE_FILE"
} > "$tmpdir/asterisk/annuaire.conf" 2>/dev/null || true

local vol_args alsa_args
vol_args=$(build_vol_args "$tmpdir")
alsa_args=$(build_alsa_args)

# [2026-08-12] Le chown est OBLIGATOIRE, pas cosmetique. Ce repertoire est
# monte sur /var/log/osmocom et les demons tournent en osmocom (999:1000
# dans l'image). Cree en root:root, osmo-hlr ne peut pas creer son fichier
# de log → "% Unable to create file '/var/log/osmocom/osmo-hlr.log'" →
# echec du parse de sa config → boucle de restart systemd → osmo-start.sh
# `exit 1` ("HLR indispensable") → run.sh meurt (set -e) et n'atteint
# aucun de ses etages suivants. Un simple probleme de droits fait donc
# tomber tout le pipeline, audio compris.
mkdir -p /tmp/osmocom-logs/op${i}
chown 999:1000 /tmp/osmocom-logs/op${i} 2>/dev/null || true

# ── Ports a exposer ─────────────────────────────────────────
local port_args=""

# ── SIP/RTP : PLUS AUCUNE PUBLICATION SUR L HOTE ────────────
# [2026-08-31] On publiait le 5060 et la plage RTP de chaque conteneur
# sur l hote. C etait la SEULE source du conflit de ports : l operateur
# natif tient 5060/udp (Asterisk) et 5061 (osmo-sip-connector) en
# permanence, puisqu il tourne EN MEME TEMPS que les conteneurs.
#
# On a d abord decale les publications (+100). Ca marchait, mais ca
# reposait sur une sonde "5060 est-il pris ?" prise a un instant t -
# donc sur une course : lancee juste apres l arret du banc, la sonde
# voyait le port libre, ne decalait rien, et le conteneur mourait sur
#     failed to bind host port 0.0.0.0:5061/udp: address already in use
# en restant en etat "Created", des que le natif remontait.
#
# La publication etait de toute facon INUTILE : les conteneurs sont sur
# un bridge routable depuis l hote. On les joint en direct, a leur
# propre adresse, sur le port nominal. Supprimer la publication
# supprime le conflit, l offset ET la course, d un coup.
local _cip; _cip="$(op_backbone_ip "$i")"
```

```
echo -e "  Linphone   : ${CYAN}${_cip}:5060${NC}  RTP 30000-30199 (direct, sans publication)"

# WAN en plus si active (legacy ou mesh)
if wan_active; then
    local sip_port rtp_start rtp_end sms_port
    sip_port=$(wan_sip_port "$i")
    rtp_start=$(wan_rtp_start "$i")
    rtp_end=$(wan_rtp_end "$i")
    port_args="${port_args} -p ${sip_port}:5060/udp -p ${sip_port}:5060/tcp -p
     ↪  ${rtp_start}-${rtp_end}:${rtp_start}-${rtp_end}/udp"
    echo -e "  WAN        : SIP ${sip_port} RTP ${rtp_start}-${rtp_end}"
    if [ "${WAN_MESH:-0}" = "1" ]; then
        # Un port SMS PAR operateur : le relais distant doit joindre le
        # container du BON operateur, or son injection MT passe par le
        # HLR local a son container. Un port unique livrerait tous les
        # SMS au premier operateur, qui ne connait pas les abonnes des
        # autres - "MSISDN not found in HLR", sans autre trace.
        sms_port=$(wan_sms_port "$i")
        port_args="${port_args} -p ${sms_port}:7890/tcp"
        echo -e "  WAN SMS    : ${sms_port} → relay 7890"
    fi
fi

docker rm -f "$container_name" 2>/dev/null || true
ensure_host_audio_relay

# --restart, PAS --rm : au reboot, les conteneurs operateurs ne se
# retrouvaient pas arretes mais DISPARUS - avec leur base HLR alimentee
# et tout ce qui n'est pas monte depuis l'hote. Le banc etait a remonter
# entierement apres une simple coupure. Le "docker rm -f" juste au-dessus
# rend --rm inutile, et docker refuse --rm avec --restart au parsing.
#
# OSMO_ROLE=operator : le conteneur regenere ses configs quand run.sh
# rejoue apply_config_templates, et le rattrapage d'identite SS7 de
# generate_configs.sh s'arrete net sur un role vide - sans rien
# reappliquer et sans rien dire. Le conteneur repartait alors sur le
# point code du gabarit, 1.1.2 pour TOUS, donc deux equipements a la
# meme adresse SS7 et aucun ASP attache.
# shellcheck disable=SC2086
# ── --security-opt apparmor=unconfined : SANS LUI, PID 1 MEURT ──────
# [2026-09-04] SYMPTOME : osmo-operator-2 en "Restarting (255)" a
# l'infini, et l'operateur 3 jamais cree. Diagnostic : le conteneur
# execute /scripts/entrypoint.sh, qui finit par `exec systemd`. systemd,
# en PID 1, commence par monter ses systemes de fichiers d'API - et
# celui-la echoue :
#     Failed to mount tmpfs (type tmpfs) on /run/lock
#         (MS_NOSUID|MS_NODEV|MS_NOEXEC "mode=1777,size=5242880"):
#         Permission denied
#     [!!!!!!] Failed to mount API filesystems.
#     Exiting PID 1...
# Le message ne sort JAMAIS dans `docker logs` : systemd ecrit sur
# /dev/console, pas sur stdout. `docker logs` s'arrete apres
# "Created symlink ... docker-entrypoint.service" et on ne voit qu'un
# code 255 sans explication. Il faut `docker run -t` pour le lire.
#
# CAP_SYS_ADMIN est deja la, et ne suffit pas : ce n'est pas seccomp qui
# refuse le mount, c'est APPARMOR. Le profil docker-default d'Ubuntu
# interdit les operations de montage quelles que soient les
# capabilities. Un conteneur qui fait tourner systemd doit en sortir.
# Verifie a la main sur ce banc : memes drapeaux + apparmor=unconfined
# -> "Queued start job for default target multi-user.target", conteneur
# Up ; sans lui -> exit 255 en boucle.
# [2026-10-24] SYS_NICE : c54x_exe/run.sh lance la chaine (DSP, QEMU, osmocon,
# mobile, pont) sous `nice -n -10` (RT_NICE) ; sans la capability chaque etape
# crachait « nice: cannot set niceness: Permission denied » et tournait sans priorite.
# shellcheck disable=SC2086
```

```
docker run -d \
    --restart unless-stopped \
    --name "$container_name" \
    --hostname "$container_name" \
    --network "$INTER_NET" \
    --ip "$inter_local_ip" \
    --cap-add NET_ADMIN \
    --cap-add SYS_ADMIN \
    --cap-add NET_RAW \
    --cap-add SYS_NICE \
    --security-opt apparmor=unconfined \
    --ulimit rtprio=18 \
    --shm-size=8g \
    --cgroupns host \
    --device /dev/net/tun:/dev/net/tun \
    $alsa_args \
    $port_args \
    -v /sys/fs/cgroup:/sys/fs/cgroup:rw \
    -v /tmp/osmocom-logs/op${i}:/var/log/osmocom \
    --tmpfs /tmp:exec,rw,mode=1777,size=2g \
    --tmpfs /run:exec,size=64m \
    --tmpfs /run/lock \
    -e OPERATOR_ID="$i" \
    -e OSMO_ROLE=operator \
    -e N_MS="${OP_MS[$i]}" \
    -e CONTAINER_IP="$container_ip" \
    -e GATEWAY_IP="$gateway" \
    -e INTER_STP_IP="${WAN_STP_TARGET:-$INTER_STP_IP}" \
    -e OSMO_HUB_IP="${WAN_STP_TARGET:-$INTER_STP_IP}" \
    -e WAN_NODE_ID="${_node_i}" \
    -e OSMO_WAN_NODE="${_node_i}" \
    -e HOST_IP="${HOST_IP}" \
    -e SIP_HOST_PORT="5060" \
    -e PHY_MODE="${PHY_MODE}" \
    -e HOST_AUDIO_RELAY="${HOST_AUDIO_RELAY}" \
    $vol_args \
    "$IMAGE_RUN" \
    sleep infinity

docker network connect --ip "$container_ip" "$net_name" "$container_name"

# ── UN OPERATEUR MORT N'EMPORTE PLUS LES SUIVANTS ──────────────────
# [2026-09-04] CE QUI EST ARRIVE : osmo-operator-2 repartait en boucle
# (systemd tue par AppArmor, voir le commentaire du docker run
# ci-dessus). Tout ce qui suit dans ce tour de boucle parle au conteneur
# par `docker exec`, qui repond "container is restarting" et sort en
# erreur - et start-docker.sh tourne sous `set -eu`. Le pipeline
# d'alimentation du HLR (`... | docker exec -i "$c" bash`) rendait donc
# un statut non nul, et LE SCRIPT ENTIER s'arretait la, en silence.
# Resultat observe : un op2 en "Restarting (255)", AUCUN conteneur
# osmo-operator-3, pas meme son reseau gsm-net-op3 - et rien dans la
# sortie qui dise pourquoi le troisieme manque.
#
# La panne d'un operateur doit rester LA PANNE DE CET OPERATEUR. On
# verifie donc qu'il tourne vraiment avant de lui parler ; s'il est
# mort, on le DIT (avec les deux commandes qui donnent la vraie cause,
# celle que `docker logs` cache) et on passe au suivant.
_st="$(docker inspect -f '{{.State.Status}}' "$container_name" 2>/dev/null || echo absent)"
if [ "$_st" != running ]; then
    _rc="$(docker inspect -f '{{.State.ExitCode}}' "$container_name" 2>/dev/null || echo '?')"
    echo -e "  ${RED}[!] ${container_name} n'est pas parti (etat: ${_st}, code: ${_rc}) - operateur
    ↪  SAUTE${NC}"
    echo -e "      ${YELLOW}systemd ecrit son erreur sur /dev/console : elle n'est PAS dans docker
    ↪  logs.${NC}"
    echo -e "      ${CYAN}docker logs ${container_name} | tail${NC}"
```

```
    echo -e "      ${CYAN}docker run --rm -t <memes drapeaux> ${IMAGE_RUN} sleep infinity${NC}   (pour la
     ↪  voir)"
    FAILED_OPS="${FAILED_OPS:-} ${container_name}"
    continue
fi

force_update_trees "$container_name"

# ── LE CONTENEUR S'ARRETE AU COEUR ──────────────────────────────────
# [2026-08-31] NO-ATTACH, PLUS NO-PROCESS.
# no-process arretait run.sh au coeur : STP, HLR, MSC, BSC, MGW, GGSN,
# SGSN, PCU montaient, puis plus rien - ni PHY, ni mobile, ni Asterisk,
# ni SMSC. Il fallait lancer start-native.sh A LA MAIN derriere, et en
# attendant le conteneur affichait
#     [no-process] mobile non lance - attente 4247 sautee
# c est-a-dire un operateur sans radio ni abonne joignable, alors que
# tout paraissait "pret".
#
# La pile complete demarre donc maintenant. Ce n est PAS un retour a
# l attachement bloquant : run.sh ne s attache jamais de lui-meme - il
# lance tout dans tmux et se contente d imprimer la commande d attache
# (scripts/run.sh, "Attach : tmux -S ... attach"). Le pipeline QEMU
# delegue, lui, prend CALYPSO_NO_ATTACH=1 (qosmo-grgsm/run.sh l.36).
#
# BRIDGE_NO_PROCESS=1 ./launch/start-docker.sh retablit l ancien comportement quand on
# veut vraiment le coeur seul.
local run_cmd="RUN_NO_PROCESS=${BRIDGE_NO_PROCESS:-0} CALYPSO_NO_ATTACH=1 /etc/osmocom/run.sh"
echo -e "  ${GREEN}[*] Lancement run.sh...${NC}"
# DANS UN TMUX, pas en simple processus detache.
#
# Le terminal courant est reserve a osmo-operator-1 (handoff QEMU plus
# bas) : c'est celui qu'on regarde. Les autres tournaient en
# "docker exec -d", c'est-a-dire sans session a laquelle se rattacher -
# on ne pouvait plus que LIRE leur journal, jamais reprendre la main sur
# la pile qui tourne.
#
# Une session nommee "osmo" rend chaque operateur joignable a la
# demande, sans rien afficher tant qu'on ne le demande pas :
#     docker exec -ti osmo-operator-2 tmux attach -t osmo
# (ou par tools/vty-menu.sh, qui liste les conteneurs)
#
# tee : le journal reste ecrit meme quand personne n'est attache, sinon
# un incident survenu hors session serait perdu. Repli sans tmux pour
# une image qui n'en aurait pas - le comportement d'avant, a l'identique.
# UN SEUL chemin de demarrage par conteneur.
# En mode QEMU, chaque conteneur est demarre par start-native.sh - le
# premier dans le terminal courant, les autres en session tmux. Lancer
# run.sh EN PLUS donnait deux piles concurrentes dans le meme
# conteneur : la seconde commencait par "--stop" et coupait ce que la
# premiere venait de monter. L'ASP de l'operateur 2 s'attachait, puis
# tout s'arretait - "Client connected" suivi d'un silence.
# run.sh EST conservé : c'est lui qui monte le coeur, et c'est de son
# HLR que depend l'etape suivante - l'attente du port 4258 puis le
# provisionnement des abonnes. L'en priver figeait le lancement sur
# "Attente HLR" pour un demarrage prevu plus tard.
#
# Les conteneurs qui doivent porter la pile COMPLETE la relancent
# ensuite via start-native.sh, une fois les abonnes en base : la base
# HLR survit au redemarrage, le travail n'est pas perdu.
docker exec -d "$container_name" bash -c "mkdir -p /var/log/osmocom && { ${run_cmd}; } >
 ↪  /var/log/osmocom/run.sh.log 2>&1"

# Attente HLR
echo -ne "  ${GREEN}[*] Attente HLR (4258)${NC}"
local retry=0
while ! docker exec "$container_name" bash -c "echo >/dev/tcp/127.0.0.1/4258" 2>/dev/null; do
```

```
    sleep 2; echo -n "."; retry=$((retry + 1))
    if [ $retry -ge 45 ]; then echo -e " ${RED}TIMEOUT${NC}"; break; fi
done
echo -e " ${GREEN}✓${NC}"

# ── CHAQUE HLR NE PORTE QUE SES PROPRES ABONNES ──────────────────
# On y versait TOUS les abonnes du banc, tous operateurs confondus. Cela
# simulait un roaming qui n'existe pas : un abonne de l'operateur 1
# inscrit dans le HLR de l'operateur 2 y est joignable sans qu'aucun
# lien MAP n'ait ete etabli. Et surtout, la liste devenait illisible -
# six abonnes chez un operateur qui n'en a que deux, dont on ne savait
# plus lesquels etaient les siens.
# Un vrai roaming passe par le SS7, pas par une copie de la base.
local _mine; _mine="$(awk -F: -v op="$i" '$1 == op' "$all_subscribers_file")"
local _n_mine; _n_mine="$(printf '%s\n' "$_mine" | grep -c . || true)"
echo -e "  ${GREEN}[*] Alimentation HLR Op${i} (${_n_mine} abonnes)...${NC}"

# ── ET ON EFFACE CE QUI N'EST PLUS AU PLAN ──────────────────────
# hlr.db survit aux relances. Un changement de plan - MCC par noeud,
# MSISDN a six chiffres - laisse donc derriere lui des abonnes d'un plan
# mort : IMSI que plus aucun mobile ne presente, et surtout MSISDN qu'ils
# RETIENNENT. osmo-hlr impose l'unicite du numero, si bien que le bon
# abonne ne peut plus recuperer le sien - l'erreur est alors
# "cannot update MSISDN", qui ne nomme pas celui qui le tient.
local _garder; _garder="$(printf '%s\n' "$_mine" | cut -d: -f3 | paste -sd'|' -)"
{
    echo "#!/bin/bash"
    echo "cat > /tmp/hlr_feed.vty << 'VTYCMDS'"
    echo "enable"
    if [ -n "$_garder" ]; then
        echo "__PURGE__${_garder}"
    fi
    while IFS=: read -r feed_op feed_ms feed_imsi feed_msisdn feed_ki; do
        [[ "$feed_op" =~ ^#.*$ ]] && continue
        echo "subscriber imsi ${feed_imsi} create"
        echo "subscriber imsi ${feed_imsi} update msisdn ${feed_msisdn}"
        echo "subscriber imsi ${feed_imsi} update aud2g comp128v1 ki ${feed_ki}"
    done <<< "$_mine"
    echo "end"
    echo "VTYCMDS"
} | docker exec -i "$container_name" bash

# ON NE COMPTE PAS LES REFUS. Le compte n'a pas de valeur ici : osmo-hlr
# prefixe d'un '%' TOUTES ses reponses, succes compris, et un
# "Subscriber already exists" est la reponse NORMALE d'une relance -
# le banc est reprovisionne a chaque demarrage. Toute variante de ce
# comptage finissait par signaler un probleme la ou il n'y en a pas,
# et un controle qui crie au loup ne sert plus a rien le jour ou il a
# raison.
#
# La seule preuve qu'un abonne est servi, c'est sa fiche : on RELIT la
# base. Un abonne sans cle est pire qu'un abonne absent - osmo-msc le
# trouve, reclame ses vecteurs d'authentification, n'obtient rien, et
# rejette le mobile avec un motif qui accuse la carte SIM.
local _sans_cle
# La purge : on LISTE avec sqlite (le VTY n'a pas de "show all" filtrable)
# et on SUPPRIME par le VTY - osmo-hlr tient un cache en memoire, un
# DELETE direct dans la base ne l'en previendrait pas.
docker exec "$container_name" bash -c '
    garder="$(sed -n "s/^__PURGE__//p" /tmp/hlr_feed.vty)"
    sed -i "/^__PURGE__/d" /tmp/hlr_feed.vty
    if [ -n "$garder" ] && command -v sqlite3 >/dev/null 2>&1; then
        sqlite3 /var/lib/osmocom/hlr.db "select imsi from subscriber;" 2>/dev/null \
          | grep -vE "^($garder)$" \
          | sed "s|^|subscriber imsi |; s|$| delete|" >> /tmp/hlr_purge.vty
    fi
```

```
        if [ -s /tmp/hlr_purge.vty ]; then
            { echo enable; cat /tmp/hlr_purge.vty; } > /tmp/hlr_purge_full.vty
            if command -v nc >/dev/null 2>&1; then
                (sleep 1; cat /tmp/hlr_purge_full.vty; sleep 2) | nc -q2 127.0.0.1 4258 2>/dev/null
            else
                (sleep 1; cat /tmp/hlr_purge_full.vty; sleep 3) | telnet 127.0.0.1 4258 2>/dev/null
            fi
            echo "  purge : $(grep -c . /tmp/hlr_purge.vty) abonne(s) hors plan supprime(s)" >&2
        fi
        rm -f /tmp/hlr_purge.vty /tmp/hlr_purge_full.vty
        if command -v nc >/dev/null 2>&1; then
            (sleep 1; cat /tmp/hlr_feed.vty; sleep 2) | nc -q2 127.0.0.1 4258 2>/dev/null
        else
            (sleep 1; cat /tmp/hlr_feed.vty; sleep 3) | telnet 127.0.0.1 4258 2>/dev/null
        fi
        rm -f /tmp/hlr_feed.vty
    ' >/dev/null 2>&1 || true

    _sans_cle="$(docker exec "$container_name" bash -c '
        command -v sqlite3 >/dev/null 2>&1 || exit 0
        sqlite3 /var/lib/osmocom/hlr.db "
            select count(*) from subscriber s
            left join auc_2g a on a.subscriber_id = s.id
            where a.subscriber_id is null;" 2>/dev/null' 2>/dev/null)" || true

    if [ "${_sans_cle:-0}" -gt 0 ]; then
        echo -e "  ${RED}✗ HLR Op${i} : ${_sans_cle} abonne(s) SANS CLE - ils ne pourront pas
         ↪  s'authentifier${NC}"
        echo -e "    ${CYAN}docker exec ${container_name} sqlite3 /var/lib/osmocom/hlr.db \"select s.imsi from
         ↪  subscriber s left join auc_2g a on a.subscriber_id=s.id where a.subscriber_id is null;\"${NC}"
    else
        echo -e "  ${GREEN}✓ HLR Op${i} alimente (${_n_mine} abonnes, tous avec cle)${NC}"
    fi

    # Attente dernier groupe - sautee en no-process
    if [ "${BRIDGE_NO_PROCESS:-0}" = "1" ]; then
        echo -e "  ${YELLOW}[no-process] mobile non lance - attente 4247 sautee${NC}"
    else
        echo -ne "  ${GREEN}[*] Attente groupe (${last_group_ip}:4247)${NC}"
        retry=0
        while ! docker exec "$container_name" bash -c "echo >/dev/tcp/${last_group_ip}/4247" 2>/dev/null; do
            sleep 2; echo -n "."; retry=$((retry + 1))
            if [ $retry -ge 90 ]; then echo -e " ${RED}TIMEOUT${NC}"; break; fi
        done
        echo -e " ${GREEN}OK${NC}"
        sleep 3
    fi
    echo -e "  ${GREEN}✓${NC} ${container_name} pret"
    echo ""
done

rm -f "$all_subscribers_file"

# ── Attente relays SMS ──────────────────────────────────────────
if [ "${BRIDGE_NO_PROCESS:-0}" != "1" ] && declare -f sms_routing_wait_ready > /dev/null 2>&1; then
    for i in $(seq "${OP_ID_BASE:-1}" "$(( ${OP_ID_BASE:-1} + n_operators - 1 ))"); do
        sms_routing_wait_ready "$(op_container "$i")" 90 || true
    done
fi

# ── WAN ─────────────────────────────────────────────────────────
if [ "${WAN_MESH:-0}" = "1" ]; then
    wan_mesh_apply "$n_operators"
elif [ "$WAN_ENABLED" = "true" ]; then
    setup_wan_interop "$n_operators" "$WAN_N_REMOTE"
fi
```

```
# ── Resume ──────────────────────────────────────────────────────────
echo ""
echo -e "${GREEN}${BOLD}Stack multi-operateurs demarree !${NC}"
echo ""
echo -e "  Inter-STP @ ${CYAN}${INTER_STP_IP}:2908${NC}  PC=0.0.0"
for i in $(seq "${OP_ID_BASE:-1}" "$(( ${OP_ID_BASE:-1} + n_operators - 1 ))"); do
    local rctx bb_ip n_groups
    rctx=$(op_rctx_inter "$i"); bb_ip=$(op_backbone_ip "$i")
    n_groups=$(( (${OP_MS[$i]} + 7) / 8 ))
    echo -e "  Op${i} ${OP_NAME[$i]}  STP 1.${i}.2 @ ${bb_ip}  RCTX ${rctx}  [${OP_MS[$i]} MS]"
done

echo ""
echo -e "  ${BOLD}Linphone (depuis l'hote) :${NC}"
for i in $(seq "${OP_ID_BASE:-1}" "$(( ${OP_ID_BASE:-1} + n_operators - 1 ))"); do
    local bb_ip; bb_ip=$(op_backbone_ip "$i")
    echo -e "    Op${i}: ${CYAN}${bb_ip}:5060${NC}"
done
echo -e "    linphone_A / tester → 100  |  linphone_B / testerB → 200"

if [ "${WAN_MESH:-0}" = "1" ]; then
    echo ""
    echo -e "  ${BOLD}WAN mesh :${NC} noeud ${CYAN}${WAN_NODE_ID}${NC}/${WAN_NODE_COUNT} - indicatif
     ↪  ${CYAN}$(wan_local_ind)${NC}"
    for _r in "${WAN_NODE_LIST[@]}"; do
        [ "$_r" = "$WAN_NODE_ID" ] && continue
        echo -e "    ${CYAN}${WAN_IND[$_r]}${NC}10001 → MS 10001 op1 du noeud ${_r} (${WAN_IP[$_r]})"
    done
elif [ "$WAN_ENABLED" = "true" ]; then
    echo ""
    echo -e "  ${BOLD}WAN :${NC} ${CYAN}${WAN_LOCAL_IP}${NC} (${n_operators} ops) ↔
     ↪  ${CYAN}${WAN_REMOTE_IP}${NC} (${WAN_N_REMOTE} ops)  prefix=${WAN_PREFIX}"
fi
echo ""

# ── LES CHECKS, JUSTE APRES LE LANCEMENT ────────────────────────────
# [2026-08-31] Ils tournent ICI depuis que la pile complete demarre
# (no-attach au lieu de no-process). Avant, il n y avait rien a mesurer : le
# coeur seul tournait, ni PHY ni mobile ni Asterisk, et "Stack demarree !"
# etait tout ce qu on obtenait - un banc annonce pret dont personne
# n avait verifie qu il portait un appel.
#
#   ss7_check.sh        ASP, AS, routes, SSN - la couche qui porte les appels.
#   operator_summary.sh ce que chaque operateur a REELLEMENT : abonnes
#                       HLR/VLR, BTS, contextes PDP. --live lui fait generer
#                       son propre dump ; sans ce drapeau il analyserait
#                       celui d un lab precedent et decrirait un autre banc.
#
# Chemins RELATIFS, comme ./helpers/prepare_host.sh plus haut : ce script
# suppose deja d etre lance depuis la racine du depot.
# Ni l un ni l autre ne doit faire echouer le lancement - la pile est
# debout, on RAPPORTE son etat, d ou le || true.
# OSMO_SKIP_CHECKS=1 les saute (start-multi.sh le pose : il enchaine ses
# propres verifications juste apres, inutile de jouer ss7_check deux fois).
if [ "${OSMO_SKIP_CHECKS:-0}" != "1" ]; then
    if [ -r ./checks/ss7_check.sh ]; then
        echo -e "${GREEN}${BOLD}── Check SS7 ──${NC}"
        bash ./checks/ss7_check.sh || true
        echo ""
    else
        echo -e "  ${YELLOW}checks/ss7_check.sh introuvable - check SS7 saute${NC}"
    fi
    if [ -r ./checks/operator_summary.sh ]; then
        echo -e "${GREEN}${BOLD}── Check operateurs ──${NC}"
        bash ./checks/operator_summary.sh --live || true
```

```
        echo ""
    else
        echo -e "  ${YELLOW}checks/operator_summary.sh introuvable - resume saute${NC}"
    fi
fi

# --virtualbox : le "set" ne se limite pas a cette machine. Les VM sont
# creees avant (setup-vbox-interco.sh) ; on les demarre ici, une fois que le
# hub et les operateurs locaux sont debout - un noeud qui monte avant le hub
# voit sa SCTP refusee.
if [ "${VBOX_INTERCO:-0}" = "1" ]; then
    local _lab="$START_REPO/tools/vbox-wan-lab.sh"
    if [ -x "$_lab" ] || [ -r "$_lab" ]; then
        echo ""
        echo -e "${GREEN}[*] Demarrage des VM du WAN...${NC}"
        bash "$_lab" start || echo -e "  ${YELLOW}⚠ demarrage des VM incomplet${NC}"
    fi
fi

# ICI, et pas plus tot : docker reinsere ses MASQUERADE en tete de
# POSTROUTING a CHAQUE reseau cree, et il y en a un par operateur. Rejouee
# depuis l'application du WAN, la regle se retrouvait derriere eux - donc
# inerte, alors qu'"iptables -L" la montrait toujours. A cet endroit, tous
# les reseaux existent : plus rien ne passera devant.
reassert_docker_lan_return

# L'adresse par laquelle les VM doivent router vers les conteneurs. Elle
# change avec l'interface active - un cable debranche, et c'est le wifi qui
# repond, sous une autre adresse. L'afficher ici evite d'aller la chercher
# apres coup, quand plus rien ne s'attache et que rien n'a "bouge".
local _hostip
_hostip="${HOST_LAN_IP:-$(ip route show default 2>/dev/null | awk '/^default/{print $9; exit}')}"
if [ -n "$_hostip" ] && [ "${WAN_MESH:-0}" = "1" ]; then
    echo -e "\n  ${BOLD}Cote VM${NC}, pour joindre les conteneurs :"
    echo -e "    ${CYAN}bash network/set_ip_vm.sh --via ${_hostip} --persist${NC}"
fi

echo -e "\n${GREEN}${BOLD}Stack prete !${NC}"

# ── Mode QEMU : handoff vers start-native.sh ──────────────────────────
if [ "${BRIDGE_QEMU:-0}" = "1" ]; then
    local _qemu_container; _qemu_container=$(op_container 1)
    echo -e "  ${CYAN}[*] run.sh calypso → ${_qemu_container} (terminal courant)${NC}"
    echo -e "  ${CYAN}[*] mode=${HANDOFF_MODE} pipeline=${HANDOFF_QEMU_CHOICE} chiffrement='${ENCRYPTION}'
     ↪  (choisis sur l'hote, NO_MENU)${NC}"

    # Le start-native lance DANS le conteneur doit connaitre son noeud : il
    # reverifie (et corrige) l'identite SS7 avant de demarrer la pile, au
    # lieu de dependre de ce que start-docker.sh a ecrit dans les configs.
    # Ces arguments sont fabriques par la fonction _node_args() plus bas,
    # une par conteneur. La variable `_node_args` qui vivait ici n'etait
    # LUE nulle part : une seconde version, figee sur "--op 1", du meme
    # calcul. Supprimee pour qu'il n'en reste qu'une.

    local _wan_env=""
    if [ "${WAN_MESH:-0}" = "1" ]; then
        _wan_env="WAN_NODES='$(wan_nodes_spec)' WAN_NODE_ID='${WAN_NODE_ID}' WAN_OPS='${n_operators}'"
        echo -e "  ${CYAN}[*] WAN mesh transmis au conteneur (noeud ${WAN_NODE_ID}, indicatif
         ↪  $(wan_local_ind))${NC}"
    elif [ "$WAN_ENABLED" = "true" ]; then
        _wan_env="WAN_REMOTE_IP='${WAN_REMOTE_IP}' WAN_PREFIX='${WAN_PREFIX}'
         ↪  WAN_N_REMOTE='${WAN_N_REMOTE:-$n_operators}'"
        echo -e "  ${CYAN}[*] SMS inter-WAN → ${WAN_REMOTE_IP} (prefixe ${WAN_PREFIX})${NC}"
    fi

    # ── LANCEMENT DE LA PILE RADIO ──────────────────────────────────────
```

```
# Les conteneurs se sont arretes au COEUR (no-process) : STP, HLR, MSC,
# BSC, MGW, GGSN, SGSN, PCU sont debout, mais ni PHY, ni mobile, ni
# Asterisk, ni SMSC. start-docker.sh enchaine maintenant start-native.sh sur
# chaque conteneur pour monter la pile radio - la sortie [ OK ] defile
# dans le terminal courant, on voit ce qui se passe.
#
# LE COMPORTEMENT DEPEND DU NOMBRE DE CONTENEURS (voir plus bas) :
#   - un seul   : on lance PUIS on s'attache au tmux "calypso" ;
#   - plusieurs : on lance chacun en NO-ATTACH, les [ OK ] defilent.
#
# Les commandes executees ne sont pas indicatives : ce sont EXACTEMENT
# celles que start-docker.sh imprimait auparavant a recopier a la main - meme
# identite de noeud, meme hub, meme table WAN.

# Les arguments de noeud d'un conteneur : sa place dans le plan WAN.
_node_args() {                # $1=indice du conteneur
    # Le rang d'operateur n'est 1 que si CHAQUE conteneur est un noeud.
    # Sinon les conteneurs partagent le noeud et se distinguent par
    # leur rang : figer "--op 1" faisait reecrire par set-node-id.sh
    # les trois configs du conteneur 2 avec l'identite du conteneur 1
    # (meme point code, meme routing context) - le hub, en override,
    # donnait alors tout le trafic au dernier attache. Meme convention
    # que le plan ecrit plus haut (_node_i / _op_i).
    local _n _op
    if [ "${WAN_NODE_PER_OP:-0}" = "1" ]; then
        _n=$(( ${WAN_NODE_ID:-1} + $1 - 1 )); _op=1
    else
        _n="${WAN_NODE_ID:-1}"; _op="$1"
    fi
    # Repli sur le noeud 1, TOUJOURS. Deux trous se referment ici :
    #   - sans WAN (`./launch/start-docker.sh` sans argument), la fonction n'imprimait
    #     RIEN : start-native.sh partait sans identite et montait la
    #     pile sur les point codes du gabarit (1.1.2), les memes pour
    #     tout le monde ;
    #   - avec WAN mais WAN_NODE_ID vide, elle imprimait `--node --op 1`
    #     — bash recolle les mots, start-native.sh lisait "--op" comme
    #     numero de noeud et sortait sur
    #     « [ KO ] --node (un chiffre de 1 a 9) ».
    # Un noeud unique EST le noeud 1 : c'est deja ce que start-docker.sh se
    # donne lui-meme (WAN_NODE_ID="${WAN_NODE_ID:-1}" plus haut), on
    # transmet donc la meme reponse au conteneur au lieu de rien.
    [[ "$_n"  =~ ^[1-9]$ ]] || _n=1
    [[ "$_op" =~ ^[1-9]$ ]] || _op=1
    printf -- '--node %s --op %s' "$_n" "$_op"
    # Le hub n'a de sens qu'en mesh : hors mesh il n'y a personne a
    # joindre, et un `--hub-ip` vide decalerait les arguments suivants.
    [ "${WAN_MESH:-0}" = "1" ] && [ -n "${WAN_STP_TARGET:-}" ] && \
        printf -- ' --hub-ip %s' "$WAN_STP_TARGET"
    return 0
}

echo ""

# La commande start-native.sh d'un conteneur : identite de noeud + hub
# + table WAN. Rien d'indicatif ici, ce sont EXACTEMENT les arguments
# que start-docker.sh executait quand il imprimait les lignes a recopier.
_hcmd() {                     # $1=indice du conteneur -> imprime la commande
    local _na; _na="$(_node_args "$1")"
    # [2026-10-24] `unset PONT_PY` : l'image (Dockerfile.run) le figeait sur
    # pont.py (grgsm). En DSP, c54x_exe/run.sh lancait alors pont.py
    # --dsp-port 6702, refuse, et aucun burst n'arrivait au DSP.
    local _cmd="cd /opt/GSM/osmo-operator && unset PONT_PY && NO_MENU=1"
    # [2026-08-31] CALYPSO_NO_ATTACH SUIT LE HANDOFF.
    # L attache ne venait ni d ici ni de start-native.sh, mais du
    # qosmo-grgsm/run.sh qu il appelle en bout de chaine :
    #     NO_ATTACH="${CALYPSO_NO_ATTACH:-0}"      (run.sh l.36)
```

```
    #     [ $TTY -eq 1 ] && [ "$NO_ATTACH" != 1 ] && exec tmux attach
    # Le defaut est donc D ATTACHER, et `docker exec -t` fournit
    # justement un TTY. Resultat : meme avec deux conteneurs - donc dans
    # la branche NO-ATTACH de ce script - la fenetre finissait dans le
    # tmux du conteneur, start-multi.sh ne reprenait jamais la main, et
    # ni les checks ni le bilan ne s executaient.
    # La variable posee plus haut sur /etc/osmocom/run.sh ne suffisait
    # pas : elle ne traversait pas ce handoff-ci.
    [ "${OSMO_NO_ATTACH:-0}" = "1" ] && _cmd="$_cmd CALYPSO_NO_ATTACH=1"
    _cmd="$_cmd MODE='${HANDOFF_MODE}' QEMU_CHOICE='${HANDOFF_QEMU_CHOICE}'"
    # Le chiffrement de l'HOTE, pas une constante : les configs du
    # conteneur sont deja ecrites avec lui (OSMO_NO_REGEN=1 la-bas), et
    # 08-gabarits.sh refuse de demarrer si les deux different.
    # [2026-10-24] CALYPSO_BRIDGE n'est plus impose : 05-run-sh.sh le met a
    # `none` en DSP (le pont est celui de c54x_exe/run.sh) et a `pont` en
    # grgsm. Force a `pont`, le pont grgsm prenait 5700-5702 avant le DSP.
    _cmd="$_cmd ENCRYPTION='${ENCRYPTION}' CALYPSO_MODE=shunt_legit"
    [ -n "$_wan_env" ] && _cmd="$_cmd ${_wan_env}"
    _cmd="$_cmd ./start-native.sh ${_na} --force"
    printf '%s' "$_cmd"
}

local _c _ct _cmd
if [ "$n_operators" -eq 1 ] && [ "${OSMO_NO_ATTACH:-0}" != "1" ]; then
    # -- UN SEUL CONTENEUR : ON LANCE PUIS ON S'ATTACHE ----------------
    # [2026-08-31] ...SAUF si l appelant a demande le contraire.
    # A un seul conteneur, ce bloc s attachait TOUJOURS a la session
    # tmux du conteneur et n en ressortait plus. C est le bon geste
    # quand on tape la commande soi-meme ; c est le mauvais quand le
    # lancement vient d une icone ou d un script : le terminal reste
    # bloque dans le tmux, le script appelant n a jamais la main, et
    # rien de ce qui devait suivre - les checks, le bilan - ne
    # s execute. Constate depuis start-multi.sh, dont la fenetre
    # finissait sur les panneaux de osmo-operator-1.
    # OSMO_NO_ATTACH=1 saute l attache ; la pile reste debout dans sa
    # session calypso, on s y rattache a la main quand on veut.
    # start-docker.sh lance lui-meme la pile radio, dans le terminal courant :
    # les [ OK ] de start-native.sh puis de run.sh defilent sous les
    # yeux, et une fois le coeur monte on S'ATTACHE a la session tmux
    # "calypso" du conteneur - on reste dedans, on regarde la pile
    # vivre. Ctrl-b puis d pour se detacher sans rien arreter.
    _ct="$(op_container 1)"; _cmd="$(_hcmd 1)"
    echo -e "  ${BOLD}Lancement de la pile radio sur ${_ct} (terminal courant)...${NC}"
    echo ""
    docker exec -ti "$_ct" bash -c "$_cmd"
    echo ""
    echo -e "  ${BOLD}Attache a la session tmux ${CYAN}calypso${NC}${BOLD} de ${_ct}${NC}  ${CYAN}(Ctrl-b
    ↪  puis d pour se detacher)${NC}"
    echo ""
    docker exec -ti "$_ct" tmux attach -t calypso
else
    # -- PLUSIEURS CONTENEURS : NO-ATTACH, ET DETACHE ------------------
    # On ne peut pas s'attacher a N sessions tmux dans un seul terminal.
    #
    # [2026-08-31] `docker exec -t` REMPLACE PAR `-d`. Le `-t` allouait
    # un pty et le rendait au retour du script ; Docker le fermait, et
    # le noyau envoyait SIGHUP a tout le groupe de premier plan - donc a
    # TOUS les demons que run_modules lance en `cmd &` (un shell non
    # interactif n'a pas de controle de tache : ils restent dans le
    # groupe du script, dont ce pty est le terminal de controle). QEMU
    # sortait sur "terminating on signal 1", fake_trx et tcpdump avec
    # lui, et les deux `mobile` sur "Layer2 socket failed" - une
    # quinzaine de secondes apres un demarrage annonce reussi, sans que
    # rien ne relie la panne au lanceur. Les modules posent desormais
    # `setsid` (voir run_modules/_lib/radio.sh, bloc SIGHUP) ; `-d`
    # supprime la cause en amont : pas de pty, donc pas de hangup.
```

```
            #
            # CE QU'ON PERD : les [ OK ] ne defilent plus ici. Ils sont donc
            # rediriges, DANS le conteneur, vers /var/log/osmocom/start-direct.log
            # - chemin deja monte sur l'hote (/tmp/osmocom-logs/opN), rien n'est
            # perdu. Meme convention que le lancement de run.sh plus haut.
            # CE QU'ON GARDE : le sequencement. `wait_bb_vty` ne rend la main
            # qu'une fois la pile du conteneur en ligne, donc un conteneur a la
            # fois, et "Toutes les piles tournent" reste vrai a l'impression.
            echo -e "  ${BOLD}${n_operators} conteneurs : lancement detache - journal
             ↪  ${CYAN}/var/log/osmocom/start-direct.log${NC}${BOLD} dans chaque conteneur.${NC}"
            echo ""
            for _c in $(seq "${OP_ID_BASE:-1}" "$(( ${OP_ID_BASE:-1} + n_operators - 1 ))"); do
                _ct="$(op_container "$_c")"; _cmd="$(_hcmd "$_c")"
                echo -e "  ${GREEN}──── ${_ct} ──────────────────────────────────${NC}"
                docker exec -d "$_ct" bash -c "mkdir -p /var/log/osmocom && { ${_cmd}; } >
                 ↪  /var/log/osmocom/start-direct.log 2>&1"
                wait_bb_vty "$_ct" || echo -e "    ${RED}pile non prete - voir
                 ↪  /var/log/osmocom/start-direct.log${NC}"
                echo ""
            done
            echo -e "  ${BOLD}Toutes les piles tournent.${NC} Pour en suivre une :"
            echo -e "    ${CYAN}docker exec -ti <conteneur> tmux attach -t calypso${NC}"
            echo -e "  ${CYAN}Ctrl-b puis d${NC} pour se detacher sans rien arreter."
            echo ""
        fi
    fi
}

# ═══════════════════════════════════════════════════════════════════
# Mode net-host
# ═══════════════════════════════════════════════════════════════════
start_host_mode() {
    echo -e "${GREEN}Demarrage en mode net-host (1 operateur)...${NC}"
    local src_ip gw_ip
    gw_ip=$(ip route get 1 | awk '{print $3; exit}')
    src_ip=$(ip route get 1 | awk '{print $7; exit}')
    [ -z "$src_ip" ] && { echo -e "${RED}Impossible de detecter l'IP hote.${NC}"; exit 1; }
    HOST_IP="$src_ip"

    local n_ms
    n_ms=$(wt_input "net-host" "Nombre de MS :" "1") || exit 1
    n_ms=${n_ms:-1}

    prepare_host_tun
    docker rm -f egprs &>/dev/null || true

    SMS_ROUTING_DIR=$(mktemp -d)
    if declare -f sms_routing_generate > /dev/null 2>&1; then
        sms_routing_generate 1 1 "$SMS_ROUTING_DIR" "$n_ms"
    fi

    local tmpdir; tmpdir=$(mktemp -d)
    apply_config_templates "$tmpdir" \
        "$src_ip" "$gw_ip" \
        "1" "1.1.1" "1.1.2" "1.1.3" \
        "001" "01" "OsmoGSM" \
        "127.0.0.1" "shutdown" "1"

    local vol_args alsa_args
    vol_args=$(build_vol_args "$tmpdir")
    alsa_args=$(build_alsa_args)

    ensure_host_audio_relay
    local nethost_relay=""
    [ -n "$HOST_AUDIO_RELAY" ] && nethost_relay="tcp:127.0.0.1:${WSLG_TCP_PORT}"
```

```
    # --rm CONSERVE ici, contrairement aux conteneurs operateurs. Celui-la a
    # pour commande run.sh et non "sleep infinity" : il s'arrete des que run.sh
    # rend la main, et --restart le relancerait alors en boucle - sur le reseau
    # de l'HOTE, en concurrence avec le gnome-terminal qui ouvre sa propre
    # session juste apres. Il depend en outre de prepare_host_tun (apn0), qui
    # n'existe plus apres un reboot : un redemarrage automatique repartirait
    # dans un environnement incomplet. Ce mode se relance donc a la main.
    # shellcheck disable=SC2086
    docker run -d --rm --name egprs --net host \
        --cap-add NET_ADMIN --cap-add SYS_ADMIN --cap-add NET_RAW --cap-add SYS_NICE \
        --security-opt apparmor=unconfined \
        --shm-size=8g \
        --cgroupns host \
        --device /dev/net/tun:/dev/net/tun \
        $alsa_args \
        -v /sys/fs/cgroup:/sys/fs/cgroup:rw \
        --tmpfs /run --tmpfs /run/lock --tmpfs /tmp:exec,rw,mode=1777,size=2g \
        -e CONTAINER_IP="$src_ip" -e GATEWAY_IP="$gw_ip" \
        -e OPERATOR_ID="1" -e N_MS="$n_ms" \
        -e INTER_STP_IP="127.0.0.1" \
        -e HOST_IP="${HOST_IP}" -e SIP_HOST_PORT="5060" \
        -e ALSA_OUTPUT="${ALSA_OUTPUT}" -e ALSA_INPUT="${ALSA_INPUT}" \
        -e HOST_AUDIO_RELAY="${nethost_relay}" \
        $vol_args \
        "$IMAGE_RUN" /root/run.sh

    wait_bb_vty "egprs"
    sleep 3

    local _du="${SUDO_USER:-$(logname 2>/dev/null || echo "$USER")}"
    local _duid; _duid=$(id -u "$_du" 2>/dev/null)
    local _disp="${DISPLAY:-:0}"
    local _xauth="${XAUTHORITY:-/home/$_du/.Xauthority}"
    local _hscript="/tmp/osmo-gnome-host.sh"
    cat > "$_hscript" <<'EOF'
#!/usr/bin/env bash
printf '\033]0;net-host - egprs\007'
echo "=== net-host - egprs run.sh ==="
exec sudo docker exec -ti egprs /bin/bash -c "/root/run.sh; exec bash"
EOF
    chmod +x "$_hscript"
    # ── OSMO_NO_TERM=1 : LA PILE MONTE, SANS FENETRE ──────────────────────
    # [2026-09-04] Lance par une unite systemd (osmo-multi.service, et l icone
    # du bureau qui passe par elle), il n y a ni ecran ni personne devant : un
    # gnome-terminal surgirait au milieu de la session, et son contenu serait
    # perdu a la fermeture. On lance alors run.sh detache dans le conteneur, et
    # tout est dans le journal du conteneur (`docker logs egprs`).
    if [ "${OSMO_NO_TERM:-0}" = "1" ]; then
        echo -e "  ${CYAN}[*] run.sh dans egprs (OSMO_NO_TERM=1 : pas de terminal)${NC}"
        docker exec -d egprs /bin/bash -c "/root/run.sh > /var/log/osmocom/run.sh.log 2>&1"
        sleep 0.3
        return 0
    fi
    echo -e "  ${CYAN}[*] run.sh → gnome-terminal${NC}"
    sudo -u "$_du" \
        DISPLAY="$_disp" XAUTHORITY="$_xauth" \
        XDG_RUNTIME_DIR="/run/user/${_duid}" \
        DBUS_SESSION_BUS_ADDRESS="unix:path=/run/user/${_duid}/bus" \
        gnome-terminal -- bash "$_hscript" &
    sleep 0.3
}

# ═══════════════════════════════════════════════════════════════════════
stop_all() {
    echo -e "${YELLOW}Arret de tous les containers Osmocom...${NC}"
    docker ps -a --filter "name=osmo-" --format "{{.Names}}" | xargs -r docker rm -f 2>/dev/null || true
```

```
    docker ps -a --filter "name=egprs"  --format "{{.Names}}" | xargs -r docker rm -f 2>/dev/null || true
    for _chain in OSMO_WAN_INTEROP OSMO_WAN_MESH; do
        iptables -t nat -D PREROUTING -j "$_chain" 2>/dev/null || true
        iptables -t nat -F "$_chain" 2>/dev/null || true
        iptables -t nat -X "$_chain" 2>/dev/null || true
    done
    echo -e "${GREEN}Arrete.${NC}"
    disable_user_loopback
}

# ═══════════════════════════════════════════════════════════════════
# Menus
# ═══════════════════════════════════════════════════════════════════
choose_build_mode() {
    [ "${QUICK_EXPLICIT:-0}" = "1" ] && return 0
    local choice
    choice=$(wt_menu "Build de l'image" "Comment (re)construire l'image run ?" \
        "quick"  "Quick - reutilise le cache docker (rapide)" \
        "normal" "Normal - docker build --no-cache (image a neuf)") \
        || { echo "Annule."; exit 1; }
    case "$choice" in
        quick)  QUICK=1 ;;
        normal) QUICK=0 ;;
    esac
}

choose_ran_host() {
    local r
    r=$(wt_menu "RAN - Pipeline QEMU Calypso" \
        "Pipeline radio (conteneur de l'operateur) :" \
        "full-grgsm"   "gr-gsm = le DSP (defaut, valide)" \
        "start-clean"  "Pipeline historique (start-clean.sh)" \
        "full"         "full radio + vrai c54x (WIP)" \
        "shunt"        "DSP shunt canned (bissection FBSB)" \
        "bare"         "QEMU + osmocon only" \
        "faketrx-qemu" "COMBINE : fake_trx vivant + Calypso QEMU" \
        "free"         "menu complet run.sh (--menu)") || exit 1

    if [ "$r" = "faketrx-qemu" ]; then
        HANDOFF_MODE="faketrx-qemu"; HANDOFF_QEMU_CHOICE="faketrx-qemu"
    else
        HANDOFF_MODE="qemu";         HANDOFF_QEMU_CHOICE="$r"
    fi

    local e
    e=$(wt_menu "Chiffrement A5" "Chiffrement :" \
        "1" "A5/1 - chiffrement legacy (defaut)" \
        "0" "A5/0 - pas de chiffrement" \
        "2" "A5/2 - (casse, usage test)") || exit 1
    case "$e" in 0) ENCRYPTION="a5 0" ;; 2) ENCRYPTION="a5 2" ;; *) ENCRYPTION="a5 1" ;; esac
    echo -e "  ${GREEN}[RAN] mode=${CYAN}${HANDOFF_MODE}${NC}${GREEN}
     ↪  pipeline=${CYAN}${HANDOFF_QEMU_CHOICE}${NC}${GREEN} chiffrement=${CYAN}${ENCRYPTION}${NC}"
}

choose_network_mode() {
    local choice
    choice=$(wt_menu "Que lancer ?" "Premier choix :" \
        "qemu"    "PoC QEMU Calypso - qosmo-grgsm/run.sh (defaut)" \
        "virtual" "Reseau VIRTUEL multi-operateurs SS7 (config niveau 2)" \
        "hw"      "SDR physique (net-host) - hw (in dev)") || { echo "Annule."; exit 1; }
    case "$choice" in
        qemu)    NETWORK_MODE="qemu" ;;
        virtual) NETWORK_MODE="bridge" ;;
        hw)      NETWORK_MODE="host" ;;
        *) echo -e "${RED}Choix invalide.${NC}"; exit 1 ;;
    esac
```

```
}

# ═══════════════════════════════════════════════════════════════════════
# Main
# ═══════════════════════════════════════════════════════════════════════
# ── Options longues, extraites AVANT le mode positionnel ──────────────
# start-docker.sh a toujours pris son mode en $1 (qemu|virtual|hw|stop) et son build en
# quick|normal. On garde ca intact : la pre-passe ne retire que les --wan* et
# rend la main avec les positionnels d'origine.
usage_wan() {
    cat <<'USAGE'
Usage : sudo ./launch/start-docker.sh [quick|normal] [--wan ...] [qemu|virtual|hw|stop]

  --wan                   active le WAN a N noeuds (1 a 9) et pose les questions :
                            • nombre de noeuds
                            • IP publique de chaque noeud
                            • indicatif de chaque noeud (prefixe d'appel)
                            • numero du noeud construit par CE lancement
  --wan-nodes "1:IP:IND ..."  meme chose sans question (scriptable)
  --wan-id N              numero du noeud local (sinon deduit des IP locales)
  --wan-conf FICHIER      table a lire/ecrire (defaut /etc/osmo-wan.conf)

  --virtualbox[=N]        monte le WAN entre CETTE machine et des VM VirtualBox
                          (implique --wan). Cette machine devient un noeud, les
                          autres sont des VM sur un segment host-only.
  --vbox-node N           numero de noeud porte par cette machine (defaut 1)
  --hub-node N            noeud qui heberge l'inter-STP, le hub SS7 (defaut 1).
  --hub-ip ADRESSE        inter-STP a une adresse a lui, hors table WAN. Le hub
                          n'est alors PAS un noeud : aucun indicatif, aucun
                          abonne, et rien n'est lance localement.
  --node-per-op           chaque conteneur est un NOEUD (base = --wan-id) et non
                          un operateur de la machine : conteneur 1 -> noeud N,
                          conteneur 2 -> noeud N+1... Point codes 1.<n>1.<role>.
  --host ADRESSE          adresse de CETTE machine vue des VM (defaut : deduite
                          de la route par defaut). Recours quand la machine a
                          plusieurs cartes ou vient de changer d'adresse : c'est
                          par elle que les VM routent vers les conteneurs.
  --operator IP:PREFIXE   declare un operateur du WAN par son adresse et son
                          prefixe d'appel. Repetable ; l'ordre donne les numeros
                          de noeud. Il DECLARE seulement : aucun conteneur n'est
                          lance pour lui - c'est ainsi qu'une VM deja demarree
                          entre dans la table SMS et le dialplan. Les conteneurs
                          restent gouvernes par le nombre d'operateurs demande.
                            --operator 192.168.1.123:11   (la VM, deja lancee)
                            --operator 172.20.0.11:22   (un conteneur d'ici)
                          UN SEUL noeud du WAN doit le porter.
  --build-stp             construit l'image legere osmocom-stp (Dockerfile.stp)
                          pour le hub. Sans elle, le hub tourne sur l'image
                          complete, qui contient aussi osmo-stp.

  Sans --wan, rien ne change : aucun WAN n'est monte.
  Avec --wan, la radio passe en HYBRIDE par defaut (1 faketrx + 1 QEMU Calypso).

  Composer <indicatif><numero> joint ce numero sur le noeud correspondant.
    ex. noeud 1 = indicatif 11, noeud 2 = 22 → depuis 2, "1110001" appelle
    le MS 10001 de l'operateur 1 du noeud 1.
USAGE
}

_pos_args=()
while [ $# -gt 0 ]; do
    case "$1" in
        --wan)          WAN_MESH=1 ;;
        --wan=*)        WAN_MESH=1; WAN_NODES="${1#*=}" ;;
        --wan-nodes)    WAN_MESH=1; WAN_NODES="${2:-}"; shift ;;
        --wan-nodes=*)  WAN_MESH=1; WAN_NODES="${1#*=}" ;;
```

```
        --wan-id)       WAN_MESH=1; WAN_NODE_ID="${2:-}"; shift ;;
        --wan-id=*)     WAN_MESH=1; WAN_NODE_ID="${1#*=}" ;;
        --wan-conf)     WAN_CONF_FILE="${2:-}"; shift ;;
        --wan-conf=*)   WAN_CONF_FILE="${1#*=}" ;;
        --build-stp)      BUILD_STP=1 ;;
        --hub-node)       WAN_HUB_NODE="${2:-1}"; WAN_HUB_NODE_GIVEN=1; shift ;;
        # --hub-ip et --node-per-op IMPLIQUENT le plan WAN. Sans WAN_MESH=1,
        # _pc_mid retombe sur le plan LOCAL (_pc_mid="$i") : les point codes
        # deviennent 1.1.2 / 1.2.2 et les routing contexts 150 / 250, calcules
        # a l'identique sur CHAQUE machine du WAN. Deux noeuds attaches au meme
        # hub presentent alors le meme PC et le meme contexte ; le hub etant en
        # traffic-mode override, le dernier ASP attache vole le trafic du
        # premier, en silence. C'est exactement ce que le plan WAN, indexe par
        # numero de noeud, existe pour eviter.
        --hub-ip)         WAN_HUB_IP="${2:-}"; WAN_MESH=1; shift ;;
        --node-per-op)    WAN_NODE_PER_OP=1; WAN_NODE_PER_OP_GIVEN=1; WAN_MESH=1 ;;
        --host)           HOST_LAN_IP="${2:-}"; shift ;;
        --host=*)         HOST_LAN_IP="${1#*=}" ;;
        --operators)      OPERATOR_COUNT_HINT="${2:-}"; shift ;;
        --operators=*)    OPERATOR_COUNT_HINT="${1#*=}" ;;
        --operator)       OPERATOR_DECLS+=("${2:-}"); WAN_MESH=1; shift ;;
        --operator=*)     OPERATOR_DECLS+=("${1#*=}"); WAN_MESH=1 ;;
        --hub-ip=*)       WAN_HUB_IP="${1#*=}"; WAN_MESH=1 ;;
        --hub-node=*)     WAN_HUB_NODE="${1#*=}" ;;
        --virtualbox)     WAN_MESH=1; VBOX_INTERCO=1 ;;
        --virtualbox=*)   WAN_MESH=1; VBOX_INTERCO=1; VBOX_NODES="${1#*=}" ;;
        --vbox-node)      VBOX_HOST_NODE="${2:-1}"; shift ;;
        --vbox-node=*)    VBOX_HOST_NODE="${1#*=}" ;;
        -h|--help)      usage_wan; exit 0 ;;
        *)              _pos_args+=("$1") ;;
    esac
    shift
done

# ── --operator : de la declaration a la table WAN ──────────────────────
# "IP:PREFIXE" est ce qu'on a sous les yeux ; "id:ip:ind" est ce que le reste du
# code attend. La conversion se fait ici, une fois, plutot que de demander a
# l'utilisateur de tenir lui-meme la numerotation des noeuds.
if [ "${#OPERATOR_DECLS[@]}" -gt 0 ]; then
    if [ -n "${WAN_NODES:-}" ]; then
        echo -e "\033[0;31m--operator et --wan-nodes sont exclusifs : choisissez l'un des deux.\033[0m" >&2
        exit 2
    fi
    # LES CONTENEURS D'ABORD, dans l'ordre : osmo-operator-1 est le noeud 1,
    # -2 le noeud 2... Leur adresse est celle du backbone, leur indicatif le
    # defaut du rang. Les operateurs declares en CLI prennent les rangs
    # suivants. Numerotation lisible : le nom du conteneur donne son noeud.
    _n=0; _spec=""
    _ncont="${OPERATOR_COUNT_HINT:-0}"
    while [ "$_n" -lt "$_ncont" ]; do
        _n=$(( _n + 1 ))
        _spec="${_spec}${_spec:+ }${_n}:172.20.0.$(( 10 + _n )):$(( _n * 11 ))"
    done
    for _d in "${OPERATOR_DECLS[@]}"; do
        _ip="${_d%%:*}"; _ind="${_d##*:}"
        if [ -z "$_ip" ] || [ "$_ip" = "$_d" ] || [ -z "$_ind" ]; then
            echo -e "\033[0;31m--operator : attendu IP:PREFIXE (ex: 192.168.1.123:11), recu '$_d'\033[0m" >&2
            exit 2
        fi
        _n=$(( _n + 1 ))
        _spec="${_spec}${_spec:+ }${_n}:${_ip}:${_ind}"
    done
    WAN_NODES="$_spec"
    echo -e "\033[0;36m  Operateurs declares :\033[0m $WAN_NODES"
fi
```

```
set -- ${_pos_args[@]+"${_pos_args[@]}"}

banner
[ "${1:-}" = "stop" ] && { stop_all; exit 0; }
[ "$(id -u)" -ne 0 ] && { echo -e "${RED}Root requis${NC}"; exit 1; }

case "${1:-}" in
    quick)  QUICK=1; QUICK_EXPLICIT=1; shift ;;
    normal) QUICK=0; QUICK_EXPLICIT=1; shift ;;
esac

docker rm -f $(docker ps -aq --filter "name=osmo-") 2>/dev/null || true
docker rm -f $(docker ps -aq --filter "name=egprs") 2>/dev/null || true
docker network ls --filter "name=gsm-" -q | xargs -r docker network rm 2>/dev/null || true

# Selection du mode
case "${1:-}" in
    qemu)    NETWORK_MODE="qemu" ;;
    virtual) NETWORK_MODE="bridge" ;;
    hw)      NETWORK_MODE="host" ;;
    "")
        if [ "${WAN_MESH:-0}" = "1" ]; then
            # --wan sans mode : on ne redemande pas "que lancer ?". Un WAN se
            # monte sur des operateurs, donc mode virtuel, et le menu de build
            # reste pose (quick/normal) s'il n'a pas ete tranche en ligne.
            NETWORK_MODE="bridge"
            [ "${QUICK_EXPLICIT:-0}" = "1" ] || { check_whiptail; choose_build_mode; }
        elif [ "${OSMO_MULTI:-0}" = "1" ]; then
            NETWORK_MODE="bridge"
        elif [ "${OSMO_MENU:-1}" = "0" ]; then
            NETWORK_MODE="qemu"
        else
            check_whiptail
            choose_build_mode
            choose_network_mode
        fi
        ;;
    *) echo -e "${RED}Mode inconnu : '$1' (attendu : qemu|virtual|hw|stop)${NC}"; exit 1 ;;
esac

[ "$NETWORK_MODE" = "bridge" ] && check_whiptail
echo -e "${GREEN}Mode : ${CYAN}${NETWORK_MODE}${NC}  ${GREEN}Build : ${CYAN}$([ "${QUICK:-0}" = "1" ] && echo
 ↪  "quick (cache)" || echo "normal (--no-cache)")${NC}"

./helpers/prepare_host.sh
# [2026-10-24] OSMO_NO_BUILD=1 (start-multi.sh) : lancer l'image telle qu'elle
# est. QEMU_CACHE_BUST casse le cache a chaque appel : sans ce garde, chaque
# lancement du multi refaisait git pull + recompilation de qosmo/c54x_exe.
# check_image la construit quand meme si elle est absente.
[ "${OSMO_NO_BUILD:-0}" = 1 ] || build_run_image
check_image

case "$NETWORK_MODE" in
    qemu)   start_qemu_poc ;;
    host)   start_host_mode ;;
    bridge) start_bridge_mode ;;
esac
```

3.223 /opt/GSM/osmo-operator/launch/start-interstp.sh

20320 octets, 396 lignes → 396 lignes

```
#!/bin/bash
# ================================================================================
```

```
# start-interstp.sh - lance le NOEUD INTER-STP : le hub SS7 commun au WAN
# ============================================================================
#
# Dans un lab a une seule machine, l'inter-STP est un conteneur parmi d'autres,
# a 172.20.0.10, et il ne dessert que les operateurs de CETTE machine. Des qu'on
# repartit les operateurs sur plusieurs noeuds, il devient ce qu'il est vraiment :
# un equipement a part, avec sa propre adresse, auquel tous les noeuds
# s'attachent. C'est lui - et lui seul - qui fait traverser le SS7 au WAN.
#
#   noeud 1 (1.11.2) ─┐
#   noeud 2 (1.21.2) ─┼─ M3UA/SCTP 2908 ─► inter-STP 192.168.1.49    PC 0.0.0
#   noeud 3 (1.31.2) ─┘
#
# LE PLAN DE POINT CODES, ET POURQUOI IL CHANGE
# Sur une machine isolee, chaque operateur N porte 1.N.2 - et cette formule se
# recalcule a l'identique partout. Trois noeuds attaches au meme hub y
# presenteraient donc trois fois le meme point code. Un point code est une
# ADRESSE : deux equipements qui partagent la leur, ce n'est pas un conflit de
# nom, c'est du routage faux. Le plan WAN encode donc le noeud dedans :
#
#     PC   = 1.<noeud><op>.<role>     role 1=MSC 2=STP 3=BSC
#     RCTX = noeud*1000 + op*100 + 50
#
# Usage :
#   sudo ./launch/start-interstp.sh [--nodes 3] [--ops 1] [--ip 192.168.1.49]
#   sudo ./launch/start-interstp.sh --status | --stop
#
#   --nodes N     nombre de noeuds desservis (1-9, defaut : lu dans /etc/osmo-wan.conf)
#   --ops K       operateurs par noeud (1-9, defaut 1)
#   --ip ADRESSE  adresse du hub ; posee sur l'interface si elle manque
#   --iface NOM   interface ou poser l'adresse (defaut : detectee)
#   --port N      port M3UA (defaut 2908)
#   --menu        pose les questions meme si une table existe deja
#   --defaut      applique la table du banc, en ignorant celle deja posee
#   --ips "a b c" les IP des noeuds operateurs, dans l'ordre (sans question) ;
#                 « id:ip » garde le numero du noeud (table 1,3 -> noeuds 1 et 3)
#   --no-save     ne reecrit pas /etc/osmo-wan.conf (appel depuis start-native.sh)
#   --no-ip       ne touche pas a la configuration reseau
#   --foreground  reste au premier plan (journal a l'ecran)
# ============================================================================
set -uo pipefail

RED='\033[0;31m'; GREEN='\033[0;32m'; YELLOW='\033[1;33m'
CYAN='\033[0;36m'; BOLD='\033[1m'; NC='\033[0m'

# Racine du depot : ce script vit dans launch/ depuis le 2026-10-18.
HERE="$(cd "$(dirname "$(readlink -f "${BASH_SOURCE[0]}")")/.." && pwd)"
CONF_DIR="${CONF_DIR:-/etc/osmocom}"
CFG="${CONF_DIR}/osmo-stp-interop.cfg"
LOG="${LOG:-/var/log/osmocom/osmo-stp-interop.log}"
PIDFILE="/run/osmo-interstp.pid"

NODES=""
OPS=1
MENU=0
USE_DEFAULTS=0
IPS=""
SAVE_TABLE=1
# Le hub du banc : le conteneur osmo-inter-stp, sur le backbone docker
# (INTER_STP_IP de start-docker.sh). L'ancien defaut (192.168.56.1, host-only
# VirtualBox) n'existe sur aucun segment des que les VM sont pontees : le hub
# s'y liait sans que personne ne puisse l'atteindre.
HUB_IP="172.20.0.10"
# ── La table du banc, defaut de tous les prompts ──────────────────────
# [2026-09-03] Table remise a jour - la meme que build-iso.sh : une table lue
# ici doit ressembler a celle qu'il ecrit, sinon les point codes d'un cote ne
```

```
# correspondent plus aux AS declares de l'autre.
#   noeud 1  192.168.1.123  la VM              indicatif 11
#   noeud 2  172.20.0.12  osmo-operator-2    indicatif 22   (conteneur)
#   noeud 3  172.20.0.13  osmo-operator-3    indicatif 33   (conteneur)
#   hub      172.20.0.10  osmo-inter-stp
# MSISDN = <noeud>00<op><ms> (100101 = noeud 1, op 1, MS 1).
WAN_NODES_DEFAULT="1:192.168.1.123:11 2:172.20.0.12:22 3:172.20.0.13:33"

# L'adresse par defaut du noeud i, lue dans cette table.
_default_node_ip() {
    printf '%s' "$WAN_NODES_DEFAULT" | tr ' ' '\n' \
        | awk -F: -v n="$1" '$1==n{print $2; exit}'
}
IFACE=""
PORT=2908
SET_IP=1
FOREGROUND=0
ACTION=start

while [ $# -gt 0 ]; do
    case "$1" in
        --nodes)      NODES="${2:-}"; shift ;;
        --nodes=*)    NODES="${1#*=}" ;;
        --ops)        OPS="${2:-1}"; shift ;;
        --ops=*)      OPS="${1#*=}" ;;
        --ip)         HUB_IP="${2:-}"; shift ;;
        --ip=*)       HUB_IP="${1#*=}" ;;
        --iface)      IFACE="${2:-}"; shift ;;
        --iface=*)    IFACE="${1#*=}" ;;
        --port)       PORT="${2:-2908}"; shift ;;
        --port=*)     PORT="${1#*=}" ;;
        --menu)       MENU=1 ;;
        --defaut|--default) USE_DEFAULTS=1 ;;
        --ips)        IPS="${2:-}"; shift ;;
        --ips=*)      IPS="${1#*=}" ;;
        --no-ip)      SET_IP=0 ;;
        --no-save)    SAVE_TABLE=0 ;;
        --foreground) FOREGROUND=1 ;;
        --status)     ACTION=status ;;
        --stop)       ACTION=stop ;;
        -h|--help)    sed -n '2,40p' "$0" | sed 's/^# \{0,1\}//'; exit 0 ;;
        *) echo -e "${RED}option inconnue : $1${NC}" >&2; exit 2 ;;
    esac
    shift
done

banner() {
    echo -e "${CYAN}${BOLD}"
    echo "╔══════════════════════════════════════════════════════════════════════╗"
    echo "║   Inter-STP - hub SS7 du WAN osmo-operator   ·   PC 0.0.0            ║"
    echo "╚══════════════════════════════════════════════════════════════════════╝"
    echo -e "${NC}"
}

stp_pid() {
    if [ -r "$PIDFILE" ] && kill -0 "$(cat "$PIDFILE")" 2>/dev/null; then
        cat "$PIDFILE"; return 0
    fi
    pgrep -f 'osmo-stp .*osmo-stp-interop\.cfg' | head -1
}

# Port VTY du hub. Le 4239 par defaut est celui du STP local du noeud : sur une
# machine qui porte deja un osmo-stp, le hub ne peut pas le prendre. On lit le
# port ecrit dans la config (bind 127.0.0.1 N), sinon 4239.
vty_port() {
    local p
```

```
    p="$(sed -n 's/^ *bind 127\.0\.0\.1 \([0-9]*\)$/\1/p' "$CFG" 2>/dev/null | head -1)"
    echo "${p:-4239}"
}

# ── status ──────────────────────────────────────────────────────────
if [ "$ACTION" = "status" ]; then
    banner
    pid="$(stp_pid)"
    if [ -z "$pid" ]; then
        echo -e "  ${RED}✗${NC} inter-STP arrete"; exit 1
    fi
    echo -e "  ${GREEN}✓${NC} inter-STP actif (pid ${pid})"
    echo -e "  Ecoute : $(ss -lna 2>/dev/null | grep -E "[:.]${PORT}\b" | head -3 | tr -s ' ' | cut -d' ' -f1,5 |
    ↪  tr '\n' ' ')"
    echo ""
    # Le VTY est la seule source de verite sur QUI est attache : un ASP peut
    # avoir ouvert sa SCTP sans jamais passer ASP-ACTIVE, et rien d'autre ne le
    # montre. Le port ouvert, lui, ne prouve que l'ecoute.
    VTY_PORT="$(vty_port)"
    if command -v nc >/dev/null 2>&1; then
        echo -e "  ${BOLD}ASP attaches :${NC}"
        # Pauses entre les commandes : le VTY ne lit pas un bloc d'un coup.
        { sleep 1; printf 'enable\n'; sleep 0.5; printf 'show cs7 instance 0 asp\n'; sleep 0.5
          printf 'show cs7 instance 0 as all\n'; sleep 0.5; printf 'exit\n'; sleep 1; } \
          | nc -q2 127.0.0.1 "$VTY_PORT" 2>/dev/null | tr -d '\r' | sed 's/^/    /' | grep -vE '^\s*$' | tail -30 \
            || echo -e "    ${YELLOW}VTY ${VTY_PORT} muet${NC}"
    else
        echo -e "  ${YELLOW}nc absent : ${CYAN}telnet 127.0.0.1 ${VTY_PORT}${NC} puis "show cs7 instance 0 asp""
    fi
    exit 0
fi

# ── stop ────────────────────────────────────────────────────────────
if [ "$ACTION" = "stop" ]; then
    pid="$(stp_pid)"
    if [ -n "$pid" ]; then
        kill "$pid" 2>/dev/null; sleep 1
        kill -9 "$pid" 2>/dev/null || true
        rm -f "$PIDFILE"
        echo -e "${GREEN}Inter-STP arrete.${NC}"
    else
        echo -e "${YELLOW}Inter-STP deja arrete.${NC}"
    fi
    exit 0
fi

banner
[ "$(id -u)" -eq 0 ] || { echo -e "${RED}Root requis.${NC}" >&2; exit 1; }
command -v osmo-stp >/dev/null 2>&1 || {
    echo -e "${RED}osmo-stp introuvable.${NC} Ce noeud doit porter la pile Osmocom." >&2; exit 1; }

# ═══════════════════════════════════════════════════════════════════
# Qui sont les noeuds ? - le menu
# ═══════════════════════════════════════════════════════════════════
# Le hub n'a pas besoin des IP pour ROUTER : il ecoute, les ASP viennent a lui
# (accept-asp-connections dynamic-permitted) et c'est le point code qui decide
# du reste. Il en a besoin pour trois choses concretes :
#   • ouvrir son pare-feu a ces adresses et a elles seules ;
#   • ecrire /etc/osmo-wan.conf, la table que liront les noeuds et le diagnostic ;
#   • dire, dans --status, QUI devrait etre attache - sans cette liste, un noeud
#     absent est indiscernable d'un noeud qui n'existe pas.
# shellcheck source=network/wan-nodes.sh
. "$HERE/network/wan-nodes.sh"

ask_ips() {
    local n i ip def
```

```
    echo -e "${BOLD}Configuration du hub SS7${NC}"
    echo -e "  Saisissez les adresses des noeuds operateurs, dans l'ordre."
    echo ""
    n=$(_wan_ask "Inter-STP" "Nombre de noeuds operateurs (1-9) :" "${NODES:-3}") || exit 1
    [[ "$n" =~ ^[1-9]$ ]] || { echo -e "${RED}Nombre invalide : $n${NC}" >&2; exit 2; }

    WAN_IP=(); WAN_IND=(); WAN_NODE_LIST=()
    for i in $(seq 1 "$n"); do
        # Defaut : la table du banc. Le repli ".1N sur le segment du hub" ne
        # vaut plus rien depuis que les deux premiers noeuds sont des
        # conteneurs, sur le backbone docker et non sur le LAN.
        def="$(_default_node_ip "$i")"
        [ -n "$def" ] || def="${HUB_IP%.*}.$((10 + i))"
        ip=$(_wan_ask "Noeud $i/$n" "IP du noeud operateur $i :" "$def") || exit 1
        [[ "$ip" =~ ^[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+$ ]] \
            || { echo -e "${RED}'$ip' n'est pas une IPv4${NC}" >&2; exit 2; }
        WAN_IP[$i]="$ip"; WAN_IND[$i]="$(wan_default_ind "$i")"; WAN_NODE_LIST+=("$i")
    done
    WAN_NODE_COUNT="$n"; NODES="$n"
    # Le hub ne porte aucun operateur : il n'est pas un noeud de la table. On note
    # 0, que wan_nodes_validate accepte, plutot qu'un numero qui le ferait
    # ressembler a un noeud et lui volerait un point code.
    WAN_NODE_ID=0
    WAN_OPS="$OPS"
}

# --defaut : la table du banc, quoi qu'il y ait sur la machine.
# Sans cette option, une table deja posee dans /etc/osmo-wan.conf est rechargee
# telle quelle - c'est le bon comportement au quotidien, mais c'est aussi ce qui
# fait qu'un hub garde indefiniment des adresses perimees apres un changement de
# plan : le nouveau defaut du script n'est jamais consulte.
if [ "$USE_DEFAULTS" = "1" ] && [ -z "$IPS" ]; then
    IPS="$(printf '%s' "$WAN_NODES_DEFAULT" | tr ' ' '\n' | awk -F: '{print $2}' | tr '\n' ' ')"
    [ -n "$NODES" ] || NODES="$(printf '%s' "$WAN_NODES_DEFAULT" | wc -w)"
    echo -e "  ${CYAN}--defaut : table du banc${NC} ${WAN_NODES_DEFAULT}"
fi

if [ -n "$IPS" ]; then
    # Forme scriptable : --ips "10.0.0.11 10.0.0.12 10.0.0.13"
    # [2026-10-15, revue] « id:ip » garde le numero du noeud : une table 1,3
    # (permise par wan_nodes_validate) etait renumerotee 1,2 par position, et le
    # noeud 3 s'attachait a un AS absent du hub. Une IP seule : numero = rang.
    WAN_IP=(); WAN_IND=(); WAN_NODE_LIST=(); i=0
    for tok in $IPS; do
        i=$((i+1))
        case "$tok" in *:*) id="${tok%%:*}"; ip="${tok#*:}" ;; *) id="$i"; ip="$tok" ;; esac
       [[ "$id" =~ ^[1-9]$ ]] || { echo -e "${RED}--ips : numero de noeud '$id' invalide (1-9)${NC}" >&2; exit 2; }
        WAN_IP[$id]="$ip"; WAN_IND[$id]="$(wan_default_ind "$id")"; WAN_NODE_LIST+=("$id")
    done
    WAN_NODE_COUNT="$i"; NODES="$i"; WAN_NODE_ID=0; WAN_OPS="$OPS"
elif [ "$MENU" = "1" ] || { [ -z "$NODES" ] && [ ! -r "$WAN_CONF_FILE" ]; }; then
    if [ -t 0 ]; then
        ask_ips
    else
        echo -e "${YELLOW}Pas de terminal et pas de table : on part sur 3 noeuds sans adresses.${NC}"
        NODES=3
    fi
elif [ -z "$NODES" ] && [ -r "$WAN_CONF_FILE" ]; then
    wan_nodes_load "$WAN_CONF_FILE" 2>/dev/null || true
    NODES="${WAN_NODE_COUNT:-3}"
    OPS="${WAN_OPS:-$OPS}"
    echo -e "  Table lue dans ${CYAN}${WAN_CONF_FILE}${NC} : ${NODES} noeud(s)"
    echo -e "  ${CYAN}--menu${NC} pour la ressaisir."
fi

[[ "$NODES" =~ ^[1-9]$ ]] || { echo -e "${RED}--nodes : 1 a 9${NC}" >&2; exit 2; }
```

```
[[ "$OPS"   =~ ^[1-9]$ ]] || { echo -e "${RED}--ops : 1 a 9${NC}" >&2; exit 2; }

# Table connue : on la persiste et on ouvre le pare-feu pour ces adresses.
if [ "${#WAN_NODE_LIST[@]}" -gt 0 ] && [ -n "${WAN_IP[${WAN_NODE_LIST[0]}]:-}" ]; then
    echo ""
    wan_nodes_summary 2>/dev/null || true
    # [2026-10-15, revue] --no-save : appele par start-native.sh, dont la table
    # (WAN_AUTO=1, numero du noeud, indicatifs) vient d'etre ecrite - la
    # reecrire ici en WAN_AUTO=0 / noeud 0 coupait le WAN au reboot suivant.
    if [ "$SAVE_TABLE" = 1 ]; then
        WAN_AUTO=0 wan_nodes_save "$WAN_CONF_FILE" \
            && echo -e "  ${GREEN}✓${NC} table ecrite dans ${CYAN}${WAN_CONF_FILE}${NC}"
    else
        echo -e "  ${CYAN}--no-save${NC} : ${WAN_CONF_FILE} laisse tel quel"
    fi
    if [ -x "$HERE/network/firewall-wan.sh" ] || [ -r "$HERE/network/firewall-wan.sh" ]; then
        for i in "${WAN_NODE_LIST[@]}"; do
            bash "$HERE/network/firewall-wan.sh" "${WAN_IP[$i]}" "$OPS" >/dev/null 2>&1 || true
        done
        echo -e "  ${GREEN}✓${NC} pare-feu ouvert pour ${#WAN_NODE_LIST[@]} noeud(s)"
    fi
fi

# ── L'adresse du hub ──────────────────────────────────────────────────
# Elle est ecrite en dur dans la config de CHAQUE noeud : si elle manque au
# demarrage, tous les ASP echouent a s'attacher et le seul symptome est un
# "connection refused" cote noeuds, jamais cote hub.
if [ "$SET_IP" = "1" ]; then
    if ip -4 addr show 2>/dev/null | grep -q "inet ${HUB_IP}/"; then
        echo -e "  ${GREEN}✓${NC} ${CYAN}${HUB_IP}${NC} deja presente sur cette machine"
    else
        if [ -z "$IFACE" ]; then
            # L'interface qui porte deja le meme /24 : c'est le segment du WAN.
            IFACE="$(ip -4 -o addr show scope global 2>/dev/null \
                | awk -v pfx="${HUB_IP%.*}." '$4 ~ "^"pfx {print $2; exit}')"
            [ -z "$IFACE" ] && IFACE="$(ip -4 -o addr show scope global 2>/dev/null | awk 'NR==1{print $2}')"
        fi
        if [ -n "$IFACE" ] && ip addr add "${HUB_IP}/24" dev "$IFACE" 2>/dev/null; then
            echo -e "  ${GREEN}✓${NC} ${CYAN}${HUB_IP}/24${NC} posee sur ${IFACE}"
        else
            echo -e "  ${YELLOW}⚠${NC} impossible de poser ${HUB_IP} (interface ${IFACE:-?})"
            echo -e "     Les noeuds ne pourront pas s'attacher tant qu'elle manque."
        fi
    fi
fi

# ── Config du hub ─────────────────────────────────────────────────────
mkdir -p "$CONF_DIR" "$(dirname "$LOG")"
if [ ! -x "$HERE/helpers/create_interop.sh" ] && [ ! -r "$HERE/helpers/create_interop.sh" ]; then
    echo -e "${RED}helpers/create_interop.sh introuvable${NC}" >&2; exit 1
fi
# [2026-10-15, revue] Le port VTY du hub precedent, relu AVANT la regeneration :
# le nouveau hub reprend le meme (le scan d'un port libre, fait pendant que
# l'ancien tournait encore, le faisait deriver de 1 a chaque relance).
VTY_PREC="$(sed -n 's/^ *bind 127\.0\.0\.1 \([0-9]*\)$/\1/p' "$CFG" 2>/dev/null | head -1)"
# --listen-ip : le hub se lie a SON adresse, pas a 0.0.0.0. Sans elle, SCTP
# annonce toutes les adresses de la machine dans son INIT et les noeuds tentent
# des chemins morts - l'ASP monte puis retombe, en boucle.
WAN_NODE_IDS="${WAN_NODE_LIST[*]:-}" bash "$HERE/helpers/create_interop.sh" --listen-ip "$HUB_IP" \
     --wan "$NODES" "$OPS" "$CFG" || exit 1
[ "$PORT" = "2908" ] || sed -i "s/^ listen m3ua 2908/ listen m3ua ${PORT}/" "$CFG"
# VTY du hub. [2026-10-15, revue] 4239 tant qu'il est libre : c'est le port
# que lisent checks/diag-interstp.sh et vty-debug-dump.sh, et sur un hub dedie
# (pas d'osmo-stp local) rien ne le prend. Sinon le port du hub precedent,
# sinon le premier libre de 4300 a 4399 (vty_port le relit dans la config).
if ! grep -q '^ *bind 127\.0\.0\.1 [0-9]' "$CFG"; then
```

```
    VTY_PORT="${VTY_PORT:-}"
    hub_pid="$(stp_pid)"
    _port_pris() {   # pris par un autre que le hub qu'on va remplacer
        local who; who="$(ss -lntpH "( sport = :$1 )" 2>/dev/null)"
        [ -n "$who" ] && ! { [ -n "$hub_pid" ] && printf '%s' "$who" | grep -q "pid=$hub_pid,"; }
    }
    # [2026-10-16] Hub co-localise avec un STP operateur. L'operateur prend le
    # VTY 4239 (son defaut) quand run.sh le lance, APRES ce hub. Si le hub
    # prenait 4239 maintenant - libre, l'operateur pas encore demarre - run.sh
    # verrait 4239 ecouter, croirait OsmoSTP deja debout (run_modules/10-stp.sh,
    # mod_stp_status teste core_vty_listen 4239) et sauterait le STP operateur :
    # le MSC se retrouve sans STP (SCTP 2905 jamais ouvert), [FAIL] a chaque
    # --wan sur le noeud 1. On reserve donc 4239 a l'operateur des qu'un
    # osmo-stp.cfg operateur (VTY 4239 par defaut) est present ici.
    _op_stp_4239=0
    if [ -r "$CONF_DIR/osmo-stp.cfg" ] && [ "$CONF_DIR/osmo-stp.cfg" != "$CFG" ] \
       && ! grep -q '^ *bind 127\.0\.0\.1 [0-9]' "$CONF_DIR/osmo-stp.cfg"; then
        _op_stp_4239=1
    fi
    if [ -z "$VTY_PORT" ]; then
        if [ "$_op_stp_4239" = 0 ] && ! _port_pris 4239; then VTY_PORT=4239
        elif [ -n "$VTY_PREC" ] && [ "$VTY_PREC" != 4239 ] && ! _port_pris "$VTY_PREC"; then VTY_PORT="$VTY_PREC"
        else
            for p in $(seq 4300 4399); do _port_pris "$p" || { VTY_PORT=$p; break; }; done
        fi
    fi
    [ "${VTY_PORT:-4239}" = 4239 ] || sed -i "s/^line vty$/line vty\n bind 127.0.0.1 ${VTY_PORT}/" "$CFG"
fi
echo ""

# ── Lancement ──────────────────────────────────────────────────────────
if pid="$(stp_pid)" && [ -n "$pid" ]; then
    echo -e "  ${YELLOW}Un inter-STP tourne deja (pid ${pid}) - arret puis relance.${NC}"
    kill "$pid" 2>/dev/null; sleep 1; kill -9 "$pid" 2>/dev/null || true
fi

echo -e "  ${BOLD}Noeuds desservis :${NC} ${NODES} × ${OPS} operateur(s)"
for n in ${WAN_NODE_LIST[*]:-$(seq 1 "$NODES")}; do
    for o in $(seq 1 "$OPS"); do
        printf '    noeud %s op %s → STP %s  MSC %s  BSC %s  (rctx %s)\n' \
            "$n" "$o" "1.${n}${o}.2" "1.${n}${o}.1" "1.${n}${o}.3" "$(( n*1000 + o*100 + 50 ))"
    done
done
echo ""

if [ "$FOREGROUND" = "1" ]; then
    echo -e "  ${GREEN}osmo-stp au premier plan - Ctrl-C pour arreter${NC}"
    exec osmo-stp -c "$CFG"
fi

osmo-stp -c "$CFG" >> "$LOG" 2>&1 &
echo $! > "$PIDFILE"
sleep 2
if kill -0 "$(cat "$PIDFILE")" 2>/dev/null; then
    echo -e "  ${GREEN}✓ inter-STP lance${NC} - M3UA ${CYAN}${HUB_IP}:${PORT}${NC}, VTY $(vty_port)"
    echo -e "  Journal : ${CYAN}${LOG}${NC}"
    echo ""
    echo -e "  ${BOLD}Cote noeuds :${NC} leur osmo-stp doit viser ${CYAN}${HUB_IP}${NC}"
    echo -e "    ISO construite avec ${CYAN}--role operator --node N${NC} : deja fait."
    echo -e "  ${BOLD}Verifier qui est attache :${NC} ${CYAN}./launch/start-interstp.sh --status${NC}"
else
    echo -e "  ${RED}✗ osmo-stp n'a pas demarre${NC} - ${CYAN}tail -30 ${LOG}${NC}"
    tail -15 "$LOG" 2>/dev/null | sed 's/^/    /'
    exit 1
fi
```

3.224 /opt/GSM/osmo-operator/launch/start-multi.sh

45947 octets, 856 lignes → 854 lignes (1 groupes compactés)

```
#!/bin/bash
# ============================================================================
# start-multi.sh - LE BANC MULTI-OPERATEUR : natif + docker, relies en SS7
#
#   op 1  NATIF   deja en place (start-native.sh)   PC 1.1.2
#   op 2  DOCKER  conteneur                          PC 1.2.2   172.20.0.12
#   op 3  DOCKER  conteneur                          PC 1.3.2   172.20.0.13
#   hub   DOCKER  inter-STP                          PC 0.0.0   172.20.0.10
#                                                    M3UA/SCTP 2908
#
# CE SCRIPT NE REIMPLEMENTE RIEN. start-docker.sh sait deja lancer des operateurs en
# conteneur, poser le hub (--hub-ip) et DECLARER un operateur sans lui lancer de
# conteneur (--operator IP:PREFIXE) - ce dernier point est exactement le cas du
# natif, qui tourne deja et qu on veut seulement raccorder. Refaire ce travail
# ici donnerait une seconde formule a maintenir, et le depot montre deja ou cela
# mene : start-docker.sh et lib/gabarits.sh portent les MEMES fonctions en double.
# start-multi.sh se contente donc de trois choses : verifier, traduire la
# topologie en options, et deleguer.
#
# LA TOPOLOGIE VIENT DE addition.sh, pas d ici : /etc/osmocom/osmo-multi.conf.
# Un seul endroit la calcule.
#
# Usage :
#   sudo ./launch/start-multi.sh              monte le banc multi-operateur
#   sudo ./launch/start-multi.sh --status     etat des conteneurs et du hub
#   sudo ./launch/start-multi.sh --stop       arrete les conteneurs (le natif reste)
#   sudo ./launch/start-multi.sh --dry-run    affiche la commande sans la lancer
#   sudo ./launch/start-multi.sh --restart-natif
#                                     arrete et relance aussi la 2G native
#                                     (par defaut on la GARDE si elle tourne,
#                                     on la LANCE si elle est a l arret)
# ============================================================================
set -uo pipefail

RED='\033[0;31m'; GREEN='\033[0;32m'; YELLOW='\033[1;33m'
CYAN='\033[0;36m'; BOLD='\033[1m'; NC='\033[0m'

# Racine du depot : ce script vit dans launch/ depuis le 2026-10-18.
DIR="$(cd "$(dirname "$(readlink -f "$0")")/.." && pwd)"
MULTI_CONF="${MULTI_CONF:-/etc/osmocom/osmo-multi.conf}"

# ── LES ABONNES DU NATIF VIENNENT DE coeur.env, PAS DE ~/.bashrc ────────────
# [2026-09-04] N_MS et OPERATOR_ID (/etc/osmocom/coeur.env) n'etaient charges
# que par /root/.bashrc : depuis l'icone ou une unite systemd, le natif lance
# d'ici partait avec N_MS=1 et le HLR n'avait qu'un abonne (le second etait
# cree a la volee par le HLR avec un MSISDN aleatoire et sans Ki). On le
# charge ici comme start-native.sh le fait desormais lui-meme ; « := » laisse
# gagner une valeur deja posee dans l'environnement.
if [ -f "${OSMOCOM_CFG:-/etc/osmocom}/coeur.env" ]; then
    set -a; . "${OSMOCOM_CFG:-/etc/osmocom}/coeur.env"; set +a
fi
ACTION="start"
# [2026-10-12] La 2G native n est plus relancee a chaque clic : on la garde
# si elle tourne, on la lance si elle manque. --restart-natif (ou
# OSMO_MULTI_RESTART_NATIF=1) retablit l arret + relance.
RESTART_NATIF="${OSMO_MULTI_RESTART_NATIF:-0}"
for a in "$@"; do
    case "$a" in
        --status)  ACTION="status" ;;
        --stop)    ACTION="stop" ;;
        --dry-run) ACTION="dry" ;;
        --restart-natif) RESTART_NATIF=1 ;;
```

```
        -h|--help) sed -n '2,33p' "$0"; exit 0 ;;
    esac
done

# ── LANCE PAR UNE ICONE : TERMINAL ET PRIVILEGES ──────────────────────
# [2026-08-31] Sans ce preambule, l antenne bleue du bureau ne faisait RIEN de
# visible. Le .desktop est en Terminal=false ; le script exigeait root et
# ecrivait son refus sur stdout, c est-a-dire nulle part. Un double-clic
# lancait donc bien le script, qui sortait aussitot sur "Root requis" sans une
# fenetre ni un message : l icone paraissait morte, alors qu elle marchait.
# launch.sh avait deja ce traitement ; celui-ci ne l avait pas.
#
# On teste la presence d un TTY, pas $DISPLAY : c est la difference reelle
# entre "lance a la main" et "lance par une icone".
# ⚠ SEULEMENT pour le LANCEMENT. --status, --dry-run et --help sont des
# lectures : les rejouer dans un terminal graphique ouvrirait une fenetre a
# chaque appel depuis un script ou un pipe - et leur sortie n irait pas a
# l appelant, qui n obtiendrait RIEN. Constate en les testant d ici.
_multi_readonly=0
case " $* " in
    *" --status "*|*" --dry-run "*|*" -h "*|*" --help "*) _multi_readonly=1 ;;
esac

# [2026-08-31] stdin ET stdout : `[ ! -t 0 ]` seul prenait un
# `./launch/start-multi.sh < fichier` lance depuis un vrai terminal pour un
# double-clic, et ouvrait une fenetre dont la sortie n allait pas a l appelant.
_multi_besoin=0
if [ "$_multi_readonly" = "0" ]; then
    { [ ! -t 0 ] || [ ! -t 1 ]; } && _multi_besoin=1
fi
# OSMO_NO_TERM=1 : lance par systemd (osmo-multi.service), il n'y a ni ecran ni
# personne devant - on n'ouvre rien, la sortie va au journal.
if [ "$_multi_besoin" = "1" ] && [ "${OSMO_MULTI_TERM:-0}" != "1" ] && [ "${OSMO_NO_TERM:-0}" != "1" ]; then
    export OSMO_MULTI_TERM=1
    for _t in gnome-terminal xfce4-terminal konsole xterm; do
        command -v "$_t" >/dev/null 2>&1 || continue
        case "$_t" in
            gnome-terminal) exec "$_t" --title="osmo-operator multi-operateur" -- "$0" "$@" ;;
            *)              exec "$_t" -T "osmo-operator multi-operateur" -e "$0" "$@" ;;
        esac
    done
fi

# Le drapeau ne sert qu a ne pas boucler sur la relance. On le RETIRE de
# l environnement : exporte, il descendait dans start-docker.sh, dans le raccourci du
# natif et jusqu a launch.sh, ou il n a aucun sens. La valeur reste ici, dans
# une variable de shell : c est elle qui decide de retenir la fenetre a la fin.
_multi_own_window=0
[ "${OSMO_MULTI_TERM:-0}" = "1" ] && _multi_own_window=1
unset OSMO_MULTI_TERM

# Meme raison pour root : une lecture n a pas a demander de mot de passe.
if [ "$_multi_readonly" = "0" ]; then
        if [ "$(id -u)" -ne 0 ]; then
            if command -v pkexec >/dev/null 2>&1 && [ -n "${DISPLAY:-}" ]; then
                exec pkexec env DISPLAY="${DISPLAY:-}" XAUTHORITY="${XAUTHORITY:-}" \
                     OSMO_MULTI_TERM="$_multi_own_window" "$0" "$@"
            fi
            command -v sudo >/dev/null 2>&1 && exec sudo -E "$0" "$@"
        fi
fi

echo -e "${CYAN}${BOLD}"
echo "╔═══════════════════════════════════════════════════════════╗"
echo "║   osmo-operator - banc MULTI-OPERATEUR (SS7)              ║"
echo "╚═══════════════════════════════════════════════════════════╝"
```

```
echo -e "${NC}"

# ── PREFLIGHT ──────────────────────────────────────────────────────────
# Docker N EST PAS dans l ISO (voir addition.sh) : son absence est le cas
# NORMAL au premier lancement, pas une panne. On le dit comme tel, et on
# renvoie vers le supplement plutot que de laisser docker repondre
# "command not found" - message qui n indique nulle part quoi faire.
manque() {
    echo -e "  ${RED}✗ $1${NC}"
    echo -e "    ${CYAN}→ sudo ${DIR}/addition.sh${NC}  (ou l icone ${BOLD}Supplements${NC} du bureau)"
    exit 1
}
command -v docker >/dev/null 2>&1 || manque "docker n est pas installe"
docker info >/dev/null 2>&1       || manque "le demon docker ne repond pas"

if [ ! -f "$MULTI_CONF" ]; then
    manque "topologie absente : $MULTI_CONF"
fi
# shellcheck disable=SC1090
. "$MULTI_CONF"

: "${MULTI_HUB_IP:=172.20.0.10}"
: "${MULTI_M3UA_PORT:=2908}"
: "${MULTI_OPS:=}"
: "${MULTI_IMAGE:=osmocom-run}"

# ── LIRE LE NATIF PLUTOT QUE LE SUPPOSER ──────────────────────────────
# [2026-08-31] La topologie ecrite par addition.sh dit ce qu on ATTEND du
# natif ; ces deux fonctions disent ce qu il EST. Les confondre a coute cher :
# on affichait "op natif actif" sur la seule presence d un processus, sans voir
# qu il portait le point code d un conteneur.
#
# Le point code depend du plan choisi, et les deux existent :
#   plan LOCAL (set-node-id.sh --local) : 1.<op>.<role>        - une machine
#   plan WAN   (defaut)                 : 1.<noeud><op>.<role> - plusieurs
# Sur un banc a une machine c est le plan local qui vaut, mais rien n empeche
# la config d avoir ete posee autrement : on LIT, on ne recalcule pas.
natif_pc() {
    awk '/^[[:space:]]*point-code[[:space:]]/{print $2; exit}' \
        /etc/osmocom/osmo-stp.cfg 2>/dev/null
}

# L adresse par laquelle le natif sort vers le hub : le local-ip de son ASP
# asp-to-inter. C est elle qui doit exister sur l hote - un local-ip inbindable
# est un ASP qui ne monte jamais, sans message clair (cf. le 172.20.0.11 des
# gabarits, corrige dans generate_configs.sh).
natif_asp_ip() {
    local ip
    ip="$(awk '/^[[:space:]]*asp[[:space:]]+asp-to-inter[[:space:]]/{a=1; next}
               a && /^[[:space:]]*local-ip[[:space:]]/{print $2; exit}' \
          /etc/osmocom/osmo-stp.cfg 2>/dev/null)"
    [ -n "$ip" ] || return 0
    # [2026-08-31] ON VERIFIE QUE L ADRESSE EXISTE ICI.
    # Cette colonne affichait bêtement ce que disait la config, et la config
    # portait 172.20.0.11 - l adresse du PREMIER CONTENEUR, remise par les
    # gabarits a chaque demarrage du natif. On lisait donc "op 1 native
    # 172.20.0.11", ce qui est impossible : le natif tourne sur l hote.
    # Une adresse absente de la machine est un ASP qui ne montera jamais ; on le
    # DIT dans la colonne plutot que de laisser croire a une topologie saine.
    if ip -4 -o addr show 2>/dev/null | awk '{print $4}' | cut -d/ -f1 \
         | grep -qx "$ip"; then
        printf '%s\n' "$ip"
    else
        printf '%s(absente!)\n' "$ip"
    fi
}
```

```
# ── UN PROCESSUS DE L HOTE, PAS CELUI D UN CONTENEUR ──────────────────────
# [2026-08-31] `pgrep -x osmo-bsc` et `pgrep -f 'osmo-bsc|asterisk'` repondaient
# OUI des qu un CONTENEUR tournait : l hote voit TOUS les processus, y compris
# les leurs (mesure : osmo-bsc pid 140689 et asterisk pid 140878, enfants du
# systemd de osmo-operator-2, listes par le `ps` de l hote). Le natif etait donc
# confondu avec n importe quelle pile du banc. Deux consequences, vues toutes
# les deux :
#   - etat() affichait « op 1 native actif » sans rien de natif en marche ;
#   - lancer_natif_si_absent() sortait aussitot et NE LANCAIT JAMAIS le natif
#     des qu un conteneur tournait deja - precisement le cas d un start-multi
#     relance, celui ou on en a le plus besoin.
# Le discriminant est le NAMESPACE PID : un conteneur a le sien, l hote garde
# celui de son init. Plus sur que le cgroup, qui depend de --cgroupns.
#
# `pgrep -x` ET NON `pgrep -f` : on compare le NOM du processus, pas sa ligne de
# commande. `-f 'osmo-bsc|asterisk'` repondait OUI sur n importe quelle ligne
# CITANT ces mots - le `tail -F /var/log/osmocom/osmo-bsc.log` que le natif
# ouvre lui-meme, un editeur, ou le shell appelant. Un nom exact ne peut pas
# mentir de cette facon, et rend inutile l exclusion de notre propre PID.
_ns_hote="$(readlink /proc/1/ns/pid 2>/dev/null)"
_est_natif() {          # $@ = noms EXACTS ; vrai si l un tourne hors conteneur
    local n p
    for n in "$@"; do
        for p in $(pgrep -x "$n" 2>/dev/null); do
            [ "$(readlink "/proc/$p/ns/pid" 2>/dev/null)" = "$_ns_hote" ] && return 0
        done
    done
    return 1
}

# ── ETAT ──────────────────────────────────────────────────────────────────
etat() {
    local spec idx mode ip pc _r
    echo -e "  ${BOLD}Topologie${NC} ($MULTI_CONF)"
    for spec in $MULTI_OPS; do
        IFS=: read -r idx mode ip pc _rctx <<< "$spec"
        if [ "$mode" = "native" ]; then ip="$(natif_asp_ip)"; fi
        printf '    op %-2s %-7s %-13s PC %-8s ' "$idx" "$mode" "${ip:-(hote)}" "$pc"
        if [ "$mode" = "native" ]; then
            if _est_natif osmo-bsc asterisk \
               || systemctl is-active --quiet asterisk 2>/dev/null; then
                _r="$(natif_pc)"
                if [ -n "$_r" ] && [ "$_r" != "$pc" ]; then
                    echo -e "${GREEN}actif${NC} ${YELLOW}(PC reel ${_r})${NC}"
                else
                    echo -e "${GREEN}actif${NC}"
                fi
            else
                echo -e "${YELLOW}arrete${NC}"
            fi
        else
            if docker ps --format '{{.Names}}' 2>/dev/null | grep -qx "osmo-operator-${idx}"; then
                echo -e "${GREEN}actif${NC}"
            else
                echo -e "${YELLOW}arrete${NC}"
            fi
        fi
    done
    printf '    hub  %-7s %-13s PC %-8s ' "docker" "$MULTI_HUB_IP" "${MULTI_HUB_PC:-0.0.0}"
    if docker ps --format '{{.Names}}' 2>/dev/null | grep -qx "${MULTI_HUB_NAME:-osmo-inter-stp}"; then
        echo -e "${GREEN}actif${NC}"
    else
        echo -e "${YELLOW}arrete${NC}"
    fi
}
```

```
case "$ACTION" in
  status) etat; exit 0 ;;
  stop)
        for spec in $MULTI_OPS; do
            IFS=: read -r idx mode _ <<< "$spec"
            [ "$mode" = "docker" ] || continue
            docker rm -f "osmo-operator-${idx}" >/dev/null 2>&1 \
                && echo -e "  ${GREEN}✓${NC} osmo-operator-${idx} arrete"
        done
        docker rm -f "${MULTI_HUB_NAME:-osmo-inter-stp}" >/dev/null 2>&1 \
            && echo -e "  ${GREEN}✓${NC} hub arrete"
        echo -e "  ${CYAN}i${NC} l operateur NATIF n est pas touche - ${BOLD}sudo ${DIR}/start-native.sh stop${NC}
         ↪  pour lui."
        exit 0 ;;
esac

# ── L IMAGE ──────────────────────────────────────────────────────────────
# NOM DE L IMAGE : osmocom-run, PAS "osmo-operator".
# build.sh produit osmocom-nitb (Dockerfile, ~11 Go) puis Dockerfile.run en
# derive osmocom-run, et c est CELLE-LA que start-docker.sh lance - verifie sur le
# banc : `docker inspect osmo-operator-1 --format {{.Config.Image}}` rend
# osmocom-run. Chercher "osmo-operator" (le nom du DEPOT, pas de l image)
# rendait la sonde toujours negative : "image absente" en permanence, et une
# recompilation Osmocom de 40 minutes relancee pour rien a chaque passage.
# [2026-09-02] SI SEULE LA BASE EST LA, ON DERIVE ICI. addition.sh peut
# desormais TELECHARGER osmocom-nitb (docker pull bastienbaranoff/norf_gsm)
# au lieu de la compiler ; osmocom-run en decoule par Dockerfile.run en
# quelques secondes. Refuser de demarrer pour ca et renvoyer vers addition.sh
# - qui aurait repondu "image deja presente" - etait une boucle sans issue.
# [2026-10-24] ABSENTE : ON LA TIRE DE norf_gsm D ABORD. Le Hub publie
# osmocom-run telle quelle (Dockerfile.run deja joue) ; la derivation depuis
# osmocom-nitb ne reste qu en repli (pas de reseau, Hub injoignable).
: "${MULTI_HUB_RUN_IMAGE:=bastienbaranoff/norf_gsm:latest}"
if ! docker image inspect "$MULTI_IMAGE" >/dev/null 2>&1; then
    echo -e "  ${CYAN}→${NC} image '$MULTI_IMAGE' absente : docker pull ${MULTI_HUB_RUN_IMAGE}"
    docker pull "$MULTI_HUB_RUN_IMAGE" && docker tag "$MULTI_HUB_RUN_IMAGE" "$MULTI_IMAGE" \
        || echo -e "  ${YELLOW}!${NC} pull ${MULTI_HUB_RUN_IMAGE} impossible - repli sur la derivation locale"
fi
if ! docker image inspect "$MULTI_IMAGE" >/dev/null 2>&1; then
    if docker image inspect osmocom-nitb >/dev/null 2>&1; then
        echo -e "  ${CYAN}→${NC} image '$MULTI_IMAGE' absente, base osmocom-nitb presente : derivation
         ↪  (Dockerfile.run)"
        ( cd "$DIR" && docker build --build-arg QEMU_CACHE_BUST=$(date +%s) \
                        -f Dockerfile.run -t "$MULTI_IMAGE" . ) \
            || manque "derivation de l image '$MULTI_IMAGE' echouee"
    else
        manque "image '$MULTI_IMAGE' absente"
    fi
fi

# ── LE RACCORD DU NATIF AU HUB ──────────────────────────────────────────
# [2026-08-31] RESOLU, dans start-docker.sh, au lancement du hub.
# /etc/osmocom/osmo-stp.cfg (natif) declare
#       asp asp-to-inter 2908 2910 m3ua
#        remote-ip 127.0.0.1
# et rien n ecoutait sur 127.0.0.1:2908 : le hub vit dans le conteneur et ne
# publiait aucun port. Le hub publie desormais 2908 en SCTP sur la boucle
# locale (docker run -p 127.0.0.1:2908:2908/sctp), ce qui rend le remote-ip du
# natif exact sans toucher a une seule config regeneree.
# Verifie sur le banc : docker port -> "2908/sctp -> 127.0.0.1:2908",
# ss -an -> "sctp LISTEN 127.0.0.1:2908", docker-proxy -proto sctp actif.
#
# L autre voie - repointer l ASP natif sur 172.20.0.10 / local-ip 172.20.0.1 -
# a ete ecartee : ces deux adresses n existent que tant que le bridge docker
# est monte, et disparaissent avec les conteneurs.
```

```
#
# Le second defaut du meme bloc, lui, etait dans la config : local-ip valait
# 172.20.0.11, l adresse du PREMIER CONTENEUR, inbindable sur l hote. Corrige
# dans generate_configs.sh (apply_native_post_patches), qui ne couvrait pas
# osmo-stp.cfg.

# ── COMBIEN DE CONTENEURS ──────────────────────────────────────────
# Ceux marques "docker" dans la table, pas un compte fige : la topologie est
# la seule source.
N_DOCKER=0
for spec in $MULTI_OPS; do
    IFS=: read -r idx mode _ <<< "$spec"
    [ "$mode" = "docker" ] && N_DOCKER=$((N_DOCKER + 1))
done
[ "$N_DOCKER" -ge 1 ] || { echo -e "  ${RED}✗ aucun operateur docker dans la topologie${NC}"; exit 1; }

# OPERATOR_COUNT_HINT : start-docker.sh l utilise pour NE PAS reposer la question du
# nombre de conteneurs (voir start_bridge_mode). Sans elle, le script s arrete
# sur une boite de dialogue - fatal pour un lancement par icone.
# [2026-08-31] PAS DE --wan ICI, ET C EST LE POINT IMPORTANT.
# La premiere version passait --wan --wan-id --hub-ip --operator. Resultat
# mesure : le check de fin cherchait le hub en 192.168.1.49 et l operateur
# portait le point code 1.23.2, alors que la topologie dit 172.20.0.10 et
# 1.2.2. --wan est la machinerie MULTI-MACHINE : il lit/ecrit
# /etc/osmo-wan.conf (qui n existe meme pas ici), renumerote les point codes en
# 1.<noeud><op>.<role> et attend un hub a une adresse routable entre machines.
# Sur UNE machine, c est l inverse qu on veut - et c est deja le defaut :
# checks/wan_ss7_check.sh le dit noir sur blanc, "sans WAN l ASP de chaque
# operateur pointe sur 172.20.0.10", le hub local, avec un catch-all
# "route 0.0.0 0.0.0 → as-inter". Il n y a donc rien a demander a start-docker.sh
# au-dela du nombre de conteneurs.
#
# OPERATOR_COUNT_HINT : start-docker.sh l utilise pour NE PAS reposer la question du
# nombre de conteneurs (voir start_bridge_mode). Sans elle, le script s arrete
# sur une boite de dialogue - fatal pour un lancement par icone.
# OSMO_QUICK=1 : reutiliser le cache docker. Sans elle start-docker.sh partait en
# "Build : normal (--no-cache)", soit une recompilation Osmocom complete de ~40
# minutes A CHAQUE lancement du multi-operateur - alors que l image est deja la
# (addition.sh s en charge, et le prealable plus haut le verifie).
# OSMO_NONINTERACTIVE=1 : les boites whiptail de start-docker.sh rendent leur valeur
# par defaut au lieu d attendre. Sans elle, le lancement s arretait sur
# « Nombre d'operateurs (1-36) » - un dialogue que personne ne voit quand on a
# clique sur une icone, et le banc paraissait mort.
# --operators : le drapeau EXPLICITE de start-docker.sh (l.2552), et non la seule
# variable d environnement. start-docker.sh remettait OPERATOR_COUNT_HINT a vide des
# sa ligne 70 - corrige depuis, mais l argument reste la voie sure : il est
# analyse apres, il prime, et il documente l intention dans la ligne de
# commande qu on affiche.
# HANDOFF_MODE=faketrx-qemu : la radio MIXTE, un faketrx + un QEMU Calypso par
# operateur - le meme montage que le natif. start-docker.sh ne la choisissait seul
# qu en mode WAN ; impose ici, elle vaut aussi pour ce banc a une machine.
# OSMO_SKIP_CHECKS=1 : start-docker.sh joue desormais ss7_check et operator_summary en
# fin de lancement. Ici on enchaine verifier(), qui rejoue ss7_check avec en
# plus la lecture de la topologie - inutile de le passer deux fois de suite.
# OP_ID_BASE=2 : les conteneurs prennent les rangs 2 et 3, le 1 reste au natif.
# OSMO_NO_ATTACH=1 : ne pas finir bloque dans le tmux d un conteneur - le script
# doit rendre la main pour enchainer verifier().
# OSMO_NO_BUILD=1 : l image osmocom-run est verifiee (et derivee si absente) plus
# haut ; start-docker.sh ne la rebatit pas a chaque lancement.
# WAN_NODE_ID : le numero de noeud (MCC). Sans lui start-docker.sh le trouvait a 0 en
# mode non interactif et affichait « numero de noeud (MCC) vaut 0 - ramene a 1 »
# a chaque lancement ; la topologie le connait (MULTI_NODE, 1 sur une machine).
CMD=(env "WAN_NODE_ID=${MULTI_NODE:-1}" "OSMO_QUICK=1" "OSMO_NO_BUILD=1" "OSMO_NONINTERACTIVE=1"
 ↪  "HANDOFF_MODE=faketrx-qemu" "OSMO_SKIP_CHECKS=1"
     "OP_ID_BASE=2" "OSMO_NO_ATTACH=1"
     "$DIR/launch/start-docker.sh" virtual --operators "$N_DOCKER")
```

```
echo -e "  ${BOLD}Topologie${NC} : ${N_DOCKER} conteneur(s) + 1 natif + hub ${MULTI_HUB_IP}"
echo -e "  ${CYAN}→${NC} ${CMD[*]}"
echo

if [ "$ACTION" = "dry" ]; then
    echo -e "  ${YELLOW}--dry-run : rien n a ete lance.${NC}"
    exit 0
fi

[ "$(id -u)" -eq 0 ] || { echo -e "${RED}Root requis : sudo $0${NC}"; exit 1; }

# ── SE PLACER DANS LE DEPOT AVANT D APPELER start-docker.sh ──────────────────
# [2026-08-31] start-docker.sh appelle ses aides en chemin RELATIF - ligne 2595 :
#     ./helpers/prepare_host.sh
# Lance depuis ailleurs (une icone du bureau demarre dans $HOME), il sortait
# donc sur
#     start-docker.sh: line 2595: ./helpers/prepare_host.sh: No such file or directory
# et abandonnait AVANT de creer le moindre conteneur. Le bilan affichait
# ensuite "conteneur absent" trois fois, ce qui faisait chercher un probleme
# de docker alors que le script n avait jamais atteint le docker run.
cd "$DIR" || { echo -e "${RED}Impossible d entrer dans $DIR${NC}"; exit 1; }

# ── ALIGNER L IDENTITE SS7 DU NATIF, AVANT DE LANCER ─────────────────────────
# start-docker.sh numerote SES conteneurs 1..N sans decalage possible : avec 2
# conteneurs il produit les point codes 1.1.2 et 1.2.2. Un natif laisse a son
# defaut porte 1.1.2 lui aussi - deux equipements a la meme adresse SS7, ce que
# ce depot decrit comme "pas un conflit de nom, du routage faux".
#
# On repointe donc le natif sur le rang que la topologie lui donne, AVANT de
# lancer les conteneurs. Deux drapeaux comptent :
#   --local   plan d une seule machine, MID=<op> -> 1.<op>.<role>. SANS lui,
#             set-node-id.sh prend le plan WAN (MID=<noeud><op>) et rend
#             1.13.2 au lieu de 1.3.2 - verifie en dry-run.
#   --hub-ip  sinon l ASP est repointe sur 192.168.1.49, le hub WAN par defaut,
#             qui n existe pas ici : c est de la que venait le "Aucune
#             association SCTP vers 192.168.1.49:2908" des premiers essais.
# ── L IDENTITE DU NATIF NE TIENT PAS QU AU POINT CODE ────────────────────────
# [2026-08-31] set-node-id.sh repointe les point codes, et rien d autre. Or
# start-native.sh derive TOUT le reste de son operateur depuis OPERATOR_ID
# (/etc/osmocom/coeur.env) :
#     MS_OP_ID  = OPERATOR_ID
#     ms_imsi() = MCC MNC <op sur 4> <ms sur 6>      -> l IMSI
#     MSISDN    = 100<op><ms>
#     MS_ARFCN1 = 512 + MS_OP_ID * 2                 -> la FREQUENCE
# Un natif realigne en operateur 3 mais laisse a OPERATOR_ID=1 gardait donc les
# abonnes de l operateur 1 - MSISDN 100101, IMSI 001010001000001 - exactement
# ceux que start-docker.sh donne a son conteneur osmo-operator-1. Deux abonnes avec le
# meme MSISDN dans deux HLR, et deux cellules sur le MEME ARFCN : rien ne
# proteste au demarrage, et c est a l appel que ca part de travers.
_aligner_coeur_env() {
    local idx="$1" f="/etc/osmocom/coeur.env"
    [ -f "$f" ] || return 0
    if grep -qE '^[[:space:]]*:[[:space:]]*"\$\{OPERATOR_ID:=' "$f"; then
        sed -i -E "s|^([[:space:]]*:[[:space:]]*\"\\\$\\{OPERATOR_ID:=)[0-9]+(\\}\")|\\1${idx}\\2|" "$f"
        grep -qE "OPERATOR_ID:=${idx}\\}" "$f" \
            || echo -e "  ${GREEN}✓${NC} coeur.env : OPERATOR_ID=${idx} (IMSI, MSISDN et ARFCN du natif suivent)"
    else
        echo ": \"\${OPERATOR_ID:=${idx}}\"" >> "$f"
        echo -e "  ${GREEN}✓${NC} coeur.env : OPERATOR_ID=${idx} ajoute"
    fi
}

# Mis a 1 quand le point code du natif vient d etre change : ses demons
# gardent l ancien en memoire et DOIVENT repartir (voir tout_arreter).
NATIF_REALIGNE=0
```

```
aligner_natif() {
    local idx pc spec reel
    for spec in $MULTI_OPS; do
        IFS=: read -r idx mode _ip pc _rctx <<< "$spec"
        [ "$mode" = "native" ] || continue
        # [2026-08-31] coeur.env est aligne D ABORD, et INCONDITIONNELLEMENT.
        # Il etait traite dans la branche "point code different" plus bas : une
        # fois les point codes deja bons, aligner_natif sortait par le
        # raccourci "deja en PC 1.3.2" et OPERATOR_ID restait a 1 pour
        # toujours. Le natif gardait alors les IMSI/MSISDN de l operateur 1 -
        # 001010001000001, 100101 - exactement ceux du conteneur op1, et
        # l ARFCN avec. Les deux reglages sont INDEPENDANTS : point codes
        # (set-node-id.sh) et plan d abonnes (OPERATOR_ID) ; les lier faisait
        # que le second ne se rattrapait jamais.
        _aligner_coeur_env "$idx"
        reel="$(natif_pc)"
        [ -n "$reel" ] || return 0
        [ "$reel" = "$pc" ] && { echo -e "  ${GREEN}✓${NC} natif deja en PC ${pc} (op ${idx})"; return 0; }
        echo -e "  ${CYAN}→${NC} natif en PC ${reel}, attendu ${pc} : realignement sur l operateur ${idx}"
        if [ -x "$DIR/network/set-node-id.sh" ] || [ -r "$DIR/network/set-node-id.sh" ]; then
            if bash "$DIR/network/set-node-id.sh" --node "${MULTI_NODE:-1}" --op "$idx" \
                    --local --hub-ip 127.0.0.1 >/dev/null 2>&1; then
                echo -e "  ${GREEN}✓${NC} natif realigne : PC $(natif_pc)"
                echo -e "  ${YELLOW}!${NC} les demons natifs gardent l ancienne identite en memoire :"
                echo -e "    ils seront arretes et relances pour qu ils la relisent."
                NATIF_REALIGNE=1
            else
                echo -e "  ${RED}✗${NC} realignement du natif echoue - voir network/set-node-id.sh"
            fi
        fi
        return 0
    done
}
aligner_natif

# ── LA 2G NATIVE RESTE DEBOUT ; LES CONTENEURS REPARTENT A NEUF ──────────────
# [2026-08-31] Un clic = un banc NEUF, parce que les demons Osmocom ne
# relisent PAS leur configuration a chaud : un osmo-stp realigne sur le bon
# point code gardait l ancien en memoire, et `kill -HUP` n y change rien
# (verifie : config a jour sur le disque, ASP toujours ASP_DOWN sur l ancienne
# adresse). Le seul geste qui compte est l arret complet.
#
# [2026-10-12] MAIS PAS LE NATIF, SAUF SI SA CONFIG A CHANGE. On arretait la
# 2G native a chaque lancement du multi, pour la relancer aussitot : deux
# minutes de demontage et de remontage, les abonnes detaches, le mobile a
# refaire son attach - pour une pile qui n avait pas bouge. Desormais :
#   - natif en marche et deja aligne  -> on le GARDE tel quel ;
#   - natif a l arret                 -> lancer_natif_si_absent le LANCE ;
#   - natif realigne (aligner_natif)  -> arret + relance, il n y a pas le choix ;
#   - --restart-natif                 -> arret + relance, a la demande.
#
# Quand on l arrete, il delegue a run.sh --stop, qui demonte proprement radio
# et coeur. Rien ici ne doit faire echouer le lancement : ce qui est deja
# arrete l est tres bien, d ou les || true.
# ── LE NATIF EST UNE UNITE SYSTEMD QUAND ELLE EXISTE ─────────────────────────
# [2026-09-04] osmo-banc.service tient le natif (services/osmo-banc.service).
# L'arreter par start-native.sh --stop sous les pieds de systemd laisserait
# l'unite « active (exited) » sur une pile morte ; le relancer par le
# raccourci du bureau exige une session graphique, que le service n'a pas.
# Quand l'unite est installee, c'est elle qu'on arrete et qu'on (re)lance.
BANC_UNIT="${OSMO_BANC_SERVICE:-osmo-banc.service}"
_banc_unit_present() {
    command -v systemctl >/dev/null 2>&1 || return 1
    [ -d /run/systemd/system ] || return 1
    systemctl cat "$BANC_UNIT" >/dev/null 2>&1
}
```

```
arreter_natif() {
    if _banc_unit_present; then
        # Le natif est relance juste apres par lancer_natif_si_absent : ici on
        # se contente d'arreter proprement (ExecStop = start-native.sh --stop).
        timeout 200 systemctl stop "$BANC_UNIT" >/dev/null 2>&1 || true
        echo -e "    ${GREEN}✓${NC} pile native arretee (${BANC_UNIT})"
    elif [ -x "$DIR/start-native.sh" ]; then
        timeout 120 "$DIR/start-native.sh" --stop >/dev/null 2>&1 || true
        echo -e "    ${GREEN}✓${NC} pile native arretee"
    fi
}

tout_arreter() {
    if _est_natif osmo-bsc asterisk; then
        if [ "$NATIF_REALIGNE" = 1 ]; then
            echo -e "  ${CYAN}→${NC} natif realigne : arret de la pile native pour qu elle relise sa config"
            arreter_natif
        elif [ "$RESTART_NATIF" = 1 ]; then
            echo -e "  ${CYAN}→${NC} --restart-natif : arret de la pile native"
            arreter_natif
        else
            echo -e "  ${GREEN}✓${NC} operateur natif (2G) deja en marche : on le garde tel quel"
        fi
    else
        echo -e "  ${CYAN}i${NC} operateur natif (2G) a l arret : il sera lance avant les conteneurs"
    fi
    # Les conteneurs ne sont PAS touches ici : start-docker.sh fait deja
    # `docker rm -f $(docker ps -aq --filter name=osmo-)` en tete de course.
    # Le refaire serait redondant, et surtout destructeur depuis launch.sh -
    # un clic sur le telephone rouge, qui ne monte que le natif, balayerait
    # les conteneurs multi-operateur d a cote.
    # Les ponts audio survivent aux conteneurs (setsid) : sans ca on empile un
    # relais de plus a chaque relance, et le son se dedouble.
    pkill -f 'paplay --server=tcp:' 2>/dev/null || true
}

tout_arreter

# ── SI LE NATIF EST A TERRE, ON LANCE SON RACCOURCI ──────────────────────
# Le natif n est PAS lance par ce script : il a le sien (le telephone rouge du
# bureau -> launch.sh -> start-native.sh). Mais monter les conteneurs sans lui,
# c est monter un banc a trois operateurs dont le premier manque : le hub
# n aura qu un ASP sur les trois attendus, et le bilan de fin sortira en
# DEGRADE pour une raison qui n a rien a voir avec ce qu on vient de lancer.
#
# On declenche donc son raccourci, et AVANT les conteneurs : l ordre compte,
# start-native.sh a besoin de poser ses configs et son coeur pendant que le
# plan d adressage est encore libre.
#
# Pourquoi le .desktop et pas launch.sh en direct : c est le meme chemin que le
# double-clic, donc le meme terminal, la meme elevation pkexec et le meme
# comportement. Passer a cote reviendrait a tester une autre facon de demarrer
# que celle des utilisateurs.
NATIF_DESKTOP="${NATIF_DESKTOP:-/root/Desktop/osmo-launch.desktop}"

lancer_natif_si_absent() {
    _est_natif osmo-bsc asterisk && return 0

    if _banc_unit_present; then
        # systemctl start bloque jusqu'a la fin de start-native.sh (oneshot) :
        # quand il rend la main, le coeur est deja la. Le suivi ci-dessous ne
        # sert que si l'unite a rendu 0 sans coeur, ce qui ne devrait pas arriver.
        echo -e "  ${YELLOW}!${NC} operateur natif a l arret - ${CYAN}systemctl start ${BANC_UNIT}${NC}"
        if ! systemctl start "$BANC_UNIT"; then
            echo -e "  ${RED}✗${NC} ${BANC_UNIT} n a pas demarre - journalctl -u ${BANC_UNIT%.service} -n 50"
```

```
            return 1
        fi
        _est_natif osmo-bsc asterisk && { echo -e "  ${GREEN}✓${NC} natif lance (${BANC_UNIT})"; return 0; }
        echo -n "  attente du natif "
        for i in $(seq 1 60); do
            _est_natif osmo-bsc asterisk && { echo -e " ${GREEN}✓${NC}"; return 0; }
            sleep 1; echo -n "."
        done
        echo -e " ${YELLOW}toujours absent apres 60 s - on continue quand meme${NC}"
        return 0
    fi

    echo -e "  ${YELLOW}!${NC} operateur natif a l arret - lancement de son raccourci"
    local d="$NATIF_DESKTOP"
    # Repli sur les autres emplacements : Bureau (locale fr) puis le menu.
    for _c in "$d" "${d%/Desktop/*}/Bureau/${d##*/}" "/usr/share/applications/${d##*/}"; do
        [ -f "$_c" ] && { d="$_c"; break; }
    done
    if [ ! -f "$d" ]; then
        echo -e "  ${RED}✗${NC} raccourci introuvable ($NATIF_DESKTOP)"
        echo -e "    ${CYAN}sudo ${DIR}/start-native.sh${NC}"
        return 1
    fi

    # gio launch est LA facon d executer un .desktop : il lit Exec, applique
    # Terminal= et passe par le bus de session. gtk-launch en repli (il veut un
    # ID d application, pas un chemin - d ou le basename sans .desktop).
    # En dernier recours on extrait Exec a la main : mieux vaut demarrer le banc
    # que d echouer sur l outillage du bureau.
    # Le natif nous sert de PILE, pas de poste de travail : wireshark, linphone
    # et firefox sont deja ouverts par ailleurs (ou le seront par nos soins pour
    # les dashboards). Sans ce drapeau, launch.sh les rouvrait une seconde fois.
    export OSMO_LAUNCH_APPS=0
    # (a) LE NATIF PREND SA PROPRE FENETRE. Sans ce marqueur, launch.sh voyait
    # un tty - LE NOTRE - le prenait pour le sien, et start-native.sh le tenait
    # jusqu au bout : notre sortie ecrasee, et la main rendue seulement a la fin
    # du natif. Voir launch.sh, garde « 1. Un terminal ».
    export OSMO_TERM_TAKEN=1
    export XDG_RUNTIME_DIR="${XDG_RUNTIME_DIR:-/run/user/$(id -u)}"
    export DBUS_SESSION_BUS_ADDRESS="${DBUS_SESSION_BUS_ADDRESS:-unix:path=${XDG_RUNTIME_DIR}/bus}"
    if command -v gio >/dev/null 2>&1 && gio launch "$d" >/dev/null 2>&1; then
        echo -e "  ${GREEN}✓${NC} raccourci lance : ${CYAN}${d}${NC}"
    elif command -v gtk-launch >/dev/null 2>&1 \
         && gtk-launch "$(basename "$d" .desktop)" >/dev/null 2>&1; then
        echo -e "  ${GREEN}✓${NC} raccourci lance (gtk-launch) : ${CYAN}${d}${NC}"
    else
        local _exec
        _exec="$(sed -n 's/^Exec=//p' "$d" | head -1 | sed 's/ *%[fFuUdDnNickvm]//g')"
        [ -n "$_exec" ] || { echo -e "  ${RED}✗${NC} pas de ligne Exec dans $d"; return 1; }
        echo -e "  ${YELLOW}○${NC} gio/gtk-launch indisponibles - execution directe : ${CYAN}${_exec}${NC}"
        setsid bash -c "$_exec" </dev/null >/dev/null 2>&1 &
        disown 2>/dev/null || true
    fi

    # On attend qu il soit reellement debout : enchainer les conteneurs pendant
    # que le natif se monte, c est se disputer les memes ressources.
    echo -ne "  ${CYAN}→${NC} attente du natif"
    local i
    for i in $(seq 1 120); do
        _est_natif osmo-bsc asterisk && { echo -e " ${GREEN}✓${NC}"; return 0; }
        sleep 1; echo -n "."
    done
    echo -e " ${YELLOW}toujours absent apres 120 s - on continue quand meme${NC}"
    return 0
}
lancer_natif_si_absent
```

```
  ×3
"${CMD[@]}"
rc=$?

# ── LE CHECK DE FIN ──────────────────────────────────────────────────────
# start-docker.sh peut rendre 0 alors que le SS7 n est pas monte : son code de retour
# dit que les conteneurs ont DEMARRE, pas que les ASP se sont associes. C est
# la distinction que le depot documente ailleurs sous le nom de sonde
# mensongere - un superviseur vivant au-dessus d un pont qui n a jamais
# transporte un echantillon. On verifie donc l etat REEL, apres coup.
verifier() {
    local ko=0 spec idx mode nom _ip pc _rctx _pc_reel
    echo
    echo -e "  ${BOLD}── Verification ──${NC}"

    # 1. Les quatre elements sont-ils la ?
    for spec in $MULTI_OPS; do
        IFS=: read -r idx mode _ip pc _rctx <<< "$spec"
        if [ "$mode" = "docker" ]; then
            nom="osmo-operator-${idx}"
            if docker ps --format '{{.Names}}' 2>/dev/null | grep -qx "$nom"; then
                echo -e "    ${GREEN}✓${NC} conteneur ${nom}"
            else
                echo -e "    ${RED}✗${NC} conteneur ${nom} absent"; ko=1
            fi
        else
            if ! _est_natif osmo-bsc asterisk; then
                echo -e "    ${RED}✗${NC} operateur natif arrete - ${CYAN}sudo ${DIR}/start-native.sh --op
                 ↪  ${idx}${NC}"; ko=1
            else
                # ── LE POINT CODE DU NATIF DOIT CORRESPONDRE A SON RANG ──
                # Un natif lance sans --op tourne en operateur 1, donc en
                # PC 1.1.2 - le meme que le PREMIER CONTENEUR, que start-docker.sh
                # numerote toujours a partir de 1. La pile demarre des deux
                # cotes, tout parait vert, et c est le ROUTAGE SS7 qui est faux :
                # deux equipements a la meme adresse. On le verifie donc au lieu
                # de se contenter de "le processus tourne".
                _pc_reel="$(natif_pc)"
                if [ -z "$_pc_reel" ]; then
                    echo -e "    ${YELLOW}?${NC} operateur natif actif, point code illisible
                     ↪  (/etc/osmocom/osmo-stp.cfg)"
                elif [ "$_pc_reel" = "$pc" ]; then
                    echo -e "    ${GREEN}✓${NC} operateur natif (op ${idx}, PC ${_pc_reel})"
                else
                    echo -e "    ${RED}✗${NC} operateur natif en PC ${_pc_reel}, attendu ${pc}"
                    echo -e "      il tourne en operateur $(echo "$_pc_reel" | cut -d. -f2), rang deja pris par un
                     ↪  CONTENEUR"
                    echo -e "      ${CYAN}sudo ${DIR}/start-native.sh --op ${idx}${NC}  (puis relancer ce script)"
                    ko=1
                fi
            fi
        fi
    done
    nom="${MULTI_HUB_NAME:-osmo-inter-stp}"
    if docker ps --format '{{.Names}}' 2>/dev/null | grep -qx "$nom"; then
        echo -e "    ${GREEN}✓${NC} inter-STP ${nom}"
    else
        echo -e "    ${RED}✗${NC} inter-STP ${nom} absent"; ko=1
    fi

    # 2. Le point d entree M3UA du natif. On interroge `ss`, PAS /dev/tcp :
    #    M3UA roule sur SCTP, et une sonde TCP y est muette - la version
    #    precedente de ce check l avouait ("non concluant"), ce qui en faisait
    #    une ligne inutile. `ss -an` voit les sockets SCTP nommement.
    # awk par CHAMPS et pas un regex : `ss -an` aligne ses colonnes avec un
    # nombre d espaces VARIABLE selon la longueur des adresses, et un
```

```
    # "^sctp[[:space:]]+LISTEN.*" s y casse sans prevenir - teste, il ne
    # trouvait pas une ligne pourtant bien presente. $1/$2/$5 sont stables.
    if ss -an 2>/dev/null | awk -v p=":${MULTI_M3UA_PORT}" \
         '$1=="sctp" && $2=="LISTEN" && index($5,p)==length($5)-length(p)+1 {f=1} END{exit !f}'; then
        echo -e "    ${GREEN}✓${NC} M3UA en SCTP sur 127.0.0.1:${MULTI_M3UA_PORT} (l operateur natif peut s
         ↪  attacher)"
    else
        echo -e "    ${RED}✗${NC} rien n ecoute en SCTP sur le port ${MULTI_M3UA_PORT}"
        echo -e "      l ASP du natif (osmo-stp.cfg, remote-ip 127.0.0.1) ne trouvera personne"
        ko=1
    fi

    # 3. Le verificateur SS7 du depot, qui sait lire les ASP et les routes.
    if [ -x "$DIR/checks/ss7_check.sh" ]; then
        echo -e "    ${CYAN}→${NC} checks/ss7_check.sh"
        "$DIR/checks/ss7_check.sh" || ko=1
    else
        echo -e "    ${YELLOW}○${NC} checks/ss7_check.sh absent - verification SS7 sautee"
    fi

    echo
    if [ "$ko" = "0" ]; then
        echo -e "  ${GREEN}${BOLD}Banc multi-operateur OPERATIONNEL.${NC}"
    else
        echo -e "  ${RED}${BOLD}Le banc n est PAS complet${NC} - voir les lignes ✗ ci-dessus."
        echo -e "  ${CYAN}→${NC} etat detaille : ${BOLD}sudo $0 --status${NC}"
    fi
    return "$ko"
}

# ── LES DASHBOARDS DES CONTENEURS, EN ONGLETS SUPPLEMENTAIRES ──────────────
# launch.sh a deja ouvert Firefox sur le dashboard du NATIF et le tutoriel, et
# lance Linphone et Wireshark avec lui : on ne redouble rien de tout ca. Il
# manque seulement les consoles des conteneurs, une par operateur docker.
#
# Les adresses viennent de la topologie, pas d une liste en dur : ajouter un
# operateur a osmo-multi.conf suffit a lui ouvrir son onglet.
#
# On attend que chaque port reponde avant d ouvrir : un onglet lance trop tot
# affiche une page d erreur, et il faut alors recharger a la main - le
# dashboard d un conteneur met plusieurs secondes a monter apres le demarrage.
ouvrir_dashboards() {
    # ── OSMO_NO_TERM=1 : AUCUNE FENETRE, PAS MEME UN ONGLET ────────────────
    # [2026-09-04] Le drapeau ne couvrait que le terminal ; cette fonction
    # ouvrait quand meme Firefox sur les dashboards des conteneurs. Lance par
    # osmo-multi.service - donc par l icone du bureau, desormais - le banc
    # faisait surgir un navigateur au milieu de ce que l operateur etait en
    # train de faire, plusieurs minutes apres le clic, sans que rien ne relie
    # les deux. Un service ne s invite pas a l ecran.
    # Les adresses restent AFFICHEES : elles sont dans le journal
    # (`journalctl -u osmo-multi`), a copier quand on en veut une.
    if [ "${OSMO_NO_TERM:-0}" = "1" ]; then
        local _spec _idx _mode _ip _liste=""
        for _spec in $MULTI_OPS; do
            IFS=: read -r _idx _mode _ip _ <<< "$_spec"
            [ "$_mode" = "docker" ] && [ -n "$_ip" ] && _liste="$_liste http://${_ip}:8080"
        done
        echo -e "  ${CYAN}i${NC} OSMO_NO_TERM=1 : aucun onglet ouvert. Dashboards :${CYAN}${_liste}${NC}"
        return 0
    fi
    command -v firefox >/dev/null 2>&1 || return 0

    local spec idx mode ip urls="" i
    for spec in $MULTI_OPS; do
        IFS=: read -r idx mode ip _pc _rctx <<< "$spec"
        [ "$mode" = "docker" ] && [ -n "$ip" ] || continue
```

```
        for i in $(seq 1 60); do
            (exec 3<>"/dev/tcp/${ip}/8080") 2>/dev/null && { exec 3>&- 2>/dev/null; break; }
            sleep 1
        done
        if (exec 3<>"/dev/tcp/${ip}/8080") 2>/dev/null; then
            exec 3>&- 2>/dev/null
            urls="$urls http://${ip}:8080"
        else
            echo -e "  ${YELLOW}○${NC} dashboard op ${idx} (${ip}:8080) ne repond pas - onglet non ouvert"
        fi
    done
    [ -n "$urls" ] || return 0

    # Sous le compte de la session : Firefox y tourne deja, il ajoutera les URL
    # comme ONGLETS a la fenetre existante plutot que d en ouvrir une seconde.
    local u uid home
    u="$(natif_session_user)"; uid="$(id -u "$u" 2>/dev/null || echo 0)"
    home="$(getent passwd "$u" 2>/dev/null | cut -d: -f6)"
    # shellcheck disable=SC2086
    setsid sudo -u "$u" \
        env DISPLAY="${DISPLAY:-:0}" \
            XAUTHORITY="${XAUTHORITY:-${home:-/root}/.Xauthority}" \
            DBUS_SESSION_BUS_ADDRESS="unix:path=/run/user/${uid}/bus" \
        firefox $urls </dev/null >/dev/null 2>&1 &
    disown 2>/dev/null || true
    echo -e "  ${GREEN}✓${NC} onglets dashboards :${CYAN}${urls}${NC}"
}

# Le compte de la session graphique, meme critere qu ailleurs : la presence de
# son socket pulse, et non son nom (l ISO ouvre la session sous root).
natif_session_user() {
    local u uid sock
    for u in "${HOST_PULSE_USER:-}" "${SUDO_USER:-}" "$(logname 2>/dev/null || true)"; do
        [ -n "$u" ] || continue
        uid="$(id -u "$u" 2>/dev/null)" || continue
        [ -S "/run/user/${uid}/pulse/native" ] && { echo "$u"; return 0; }
    done
    for sock in /run/user/*/pulse/native; do
        [ -S "$sock" ] || continue
        uid="${sock#/run/user/}"; uid="${uid%%/*}"
        u="$(getent passwd "$uid" 2>/dev/null | cut -d: -f1)"
        [ -n "$u" ] && { echo "$u"; return 0; }
    done
    echo "${SUDO_USER:-$(id -un)}"
}

# EN TACHE DE FOND, et c est essentiel : cette fonction attend jusqu a 60 s par
# conteneur que le port 8080 reponde. En avant-plan, ca fige le script jusqu a
# deux minutes AVANT les verifications - et le silence ressemble a un blocage.
# Pire quand le dashboard ne monte pas du tout : on attendait le maximum pour
# n ouvrir aucun onglet. Les onglets s ouvriront quand ils pourront ; les tests,
# eux, partent tout de suite.
ouvrir_dashboards &
disown 2>/dev/null || true

echo
etat
verifier || rc=1

# ── LA FICHE DU NOEUD SUIT LE MULTI ──────────────────────────────────────
# [2026-09-04] Le natif l'a ecrite avec un seul operateur ; les conteneurs
# sont la maintenant : on la reecrit (network/node-conf.sh compte les
# osmo-operator-N en marche). C'est cette fiche qu'un autre noeud importe.
if [ "$ACTION" = "start" ] && [ -x "$DIR/start-native.sh" ]; then
    NO_MENU=1 "$DIR/start-native.sh" --gen-conf 2>/dev/null | grep -E "Fiche|mynode" || true
fi
```

```
# Ouverte par nos soins : on la retient, sinon le bilan disparait avec elle.
if [ "$_multi_own_window" = "1" ]; then
    echo
    echo "Fenetre maintenue ouverte - Entree pour fermer."
    read -r _ || true
fi
exit "$rc"
```

3.225 /opt/GSM/osmo-operator/launch/start-nitb.sh

4318 octets, 74 lignes → 74 lignes

```
#!/bin/bash
# start-nitb.sh - meme lanceur que start-docker.sh, mais sur l'image osmocom-nitb.
#
# POURQUOI UN WRAPPER ET PAS UNE COPIE. start-docker.sh fait 1 300 lignes (reseaux,
# generation de configs, SS7 inter-operateurs, ports, audio). Le dupliquer
# creerait une 2e copie qui divergerait - le depot en a deja paye le prix
# ailleurs (4 copies de si_bridge.py, 3 de gapk-start.sh, toutes divergentes).
# Ici il n'y a qu'UNE variable qui change : l'image.
#
# CE QUI A ETE VERIFIE AVANT D'ECRIRE CE SCRIPT (12/08/2026) :
#   - les deux images n'ont PAS le meme entrypoint :
#       osmocom-run  -> /scripts/entrypoint.sh
#       osmocom-nitb -> /etc/osmocom/entrypoint.sh
#   - start-docker.sh monte le dossier de config genere SUR /etc/osmocom, donc il
#     MASQUE l'entrypoint de l'image nitb. Ce n'est pas un probleme : le
#     dossier genere contient bien un entrypoint.sh (copie de scripts/
#     entrypoint.sh, le meme fichier que /scripts/entrypoint.sh dans l'image
#     run - 1988 o, identique). C'est la condition qui rend ce wrapper viable :
#     SI scripts/entrypoint.sh disparaissait du dossier genere, l'image nitb
#     demarrerait sans entrypoint alors que l'image run continuerait de
#     marcher. Le controle ci-dessous le verifie a chaque lancement.
#
# USAGE : identique a start-docker.sh, tous les arguments sont transmis tels quels.
#   ./launch/start-nitb.sh [args de start-docker.sh...]
#
# REVENIR A L'IMAGE RUN : lancer ./launch/start-docker.sh directement.

set -eu

# Racine du depot : ce script vit dans launch/ depuis le 2026-10-18.
HERE="$(cd "$(dirname "$(readlink -f "$0")")/.." && pwd)"
IMAGE_NITB="${IMAGE_NITB:-osmocom-nitb}"

RED='\033[0;31m'; GREEN='\033[0;32m'; YELLOW='\033[1;33m'; CYAN='\033[0;36m'; NC='\033[0m'

# -- 1. L'image doit EXISTER : on ne la construit pas ici ----------------------
# start-docker.sh a un check_image() qui, si l'image nommee par IMAGE_RUN est absente,
# lance build_run_image() -> `docker build -f Dockerfile.run -t "$IMAGE_RUN" .`.
# Avec IMAGE_RUN=osmocom-nitb, cela ECRASERAIT osmocom-nitb par une image
# construite depuis Dockerfile.run. On barre la route en verifiant d'abord.
if ! docker image inspect "$IMAGE_NITB" >/dev/null 2>&1; then
    echo -e "${RED}Image '${IMAGE_NITB}' introuvable.${NC}"
    echo -e "Ce script ne la construit pas volontairement : le build integre de"
    echo -e "start-docker.sh utilise ${YELLOW}Dockerfile.run${NC} et ecraserait l'image nitb."
    echo -e "Construire d'abord, depuis ${CYAN}${HERE}${NC} :"
    echo -e "  ${CYAN}docker build -f Dockerfile --target osmocom-nitb -t ${IMAGE_NITB} .${NC}"
    exit 1
fi

# -- 2. L'entrypoint doit survivre au montage de /etc/osmocom ------------------
# On teste la PRESENCE, pas le bit d'execution : dans le depot le fichier est en
# 644, et c'est normal. Le bit +x est pose EN AVAL, aux deux endroits qui en ont
```

```
# besoin - generate_configs.sh l.204-205 (`cp scripts/*` puis `chmod +x` dans le
# dossier genere, celui qui est monte sur /etc/osmocom) et Dockerfile.run l.75
# (`chmod +x /scripts/*.sh` dans l'image). Tester `-x` ici refusait donc de
# demarrer sur un depot parfaitement sain.
if [ ! -s "${HERE}/scripts/entrypoint.sh" ]; then
    echo -e "${RED}scripts/entrypoint.sh absent ou vide.${NC}"
    echo -e "L'image ${IMAGE_NITB} a son entrypoint dans /etc/osmocom, que start-docker.sh"
    echo -e "recouvre par le dossier de config genere. Sans ce fichier, le"
    echo -e "conteneur demarrerait sans entrypoint - alors que l'image"
    echo -e "osmocom-run, elle, continuerait de marcher (son entrypoint est dans"
    echo -e "/scripts). Panne asymetrique et illisible : on refuse avant."
    exit 1
fi

echo -e "${GREEN}Lanceur nitb${NC} : image = ${CYAN}${IMAGE_NITB}${NC} (au lieu d'osmocom-run)"
echo -e "  entrypoint attendu dans le conteneur : /etc/osmocom/entrypoint.sh (monte)"

# ── 3. Delegation ──────────────────────────────────────────────────────────
# start-docker.sh lit IMAGE_RUN avec l'idiome `:=` : la valeur posee ici gagne.
export IMAGE_RUN="$IMAGE_NITB"
cd "$HERE"
exec ./launch/start-docker.sh "$@"
```

3.226 /opt/GSM/osmo-operator/launch/start-oqc.sh

2185 octets, 54 lignes → 54 lignes

```
#!/bin/bash
# ============================================================================
#  launch/start-oqc.sh - lance qosmo-grgsm depuis osmo-operator
# ============================================================================
#
#  "oqc" = qosmo-grgsm, l'emulation Calypso (ARM7 + DSP TMS320C54x).
#  Ce fichier est le raccourci depuis osmo-operator : il trouve le depot et lui
#  passe la main. Toute la logique de lancement vit la-bas, dans run_modules/.
#
#      ./launch/start-oqc.sh                 profil calypso (defaut)
#      ./launch/start-oqc.sh --list          le plan, sans rien lancer
#      ./launch/start-oqc.sh --dry-run       deroule sans effet de bord
#      ./launch/start-oqc.sh --status        ou en est chaque etape
#      ./launch/start-oqc.sh --stop          arrete, en ordre inverse
#      ./launch/start-oqc.sh --check-paths   verifie les dependances de la machine
#
#  Les variables CALYPSO_* passees en prefixe traversent jusqu'a QEMU :
#      CALYPSO_MODE=native ./launch/start-oqc.sh
#
#  Ou est le depot : OQC_ROOT, sinon les emplacements habituels sont essayes.
#      OQC_ROOT=~/src/qosmo-grgsm ./launch/start-oqc.sh
#
#  A ne pas confondre avec start-native.sh, qui orchestre TOUTE la pile
#  (coeur Osmocom + radio) et dont le mode "qemu" aboutit egalement ici.
# ----------------------------------------------------------------------------
set -euo pipefail

_find_oqc() {
    [ -n "${OQC_ROOT:-}" ] && { printf '%s\n' "$OQC_ROOT"; return 0; }
    local here; here="$(cd "$(dirname "${BASH_SOURCE[0]}")/.." && pwd)"   # launch/ -> racine du depot
    local c
    for c in "$here/../qosmo-grgsm" \
             "${GSM_ROOT:-/opt/GSM}/qosmo-grgsm" \
             "$HOME/qosmo-grgsm"; do
        [ -x "$c/run.sh" ] && { (cd "$c" && pwd); return 0; }
    done
    return 1
```

```
}

if ! OQC="$(_find_oqc)"; then
    cat >&2 <<'ERR'
qosmo-grgsm introuvable.

Cherche dans : ../qosmo-grgsm, $GSM_ROOT/qosmo-grgsm, ~/qosmo-grgsm

Indiquez son emplacement :
    OQC_ROOT=/chemin/vers/qosmo-grgsm ./launch/start-oqc.sh
ou recuperez-le :
    git clone https://github.com/bbaranoff/qosmo-grgsm qosmo-grgsm
ERR
    exit 2
fi

exec "$OQC/run.sh" "$@"
```

3.227 /opt/GSM/osmo-operator/lib/audio.sh

46884 octets, 787 lignes → 787 lignes

```
# ============================================================================
#  lib/audio.sh - la chaine audio d'osmo-operator (PulseAudio + pont GAPK)
# ============================================================================
#
#  CE FICHIER EST UNE EXTRACTION, PAS UNE REECRITURE.
#  ensure_pulse / ensure_host_audio / ensure_gapk viennent telles quelles de
#  start-direct.sh.legacy (L566-722). Aucune ligne n'a ete retouchee.
#
#  POURQUOI LES PRESERVER.
#  Le mode `qemu` de start-native.sh appelait ensure_gapk JUSTE AVANT de passer
#  la main a qosmo-grgsm/start-clean.sh (legacy L1077). C'est ce qui branche le RTP
#  du MGW sur le sink `gsm_audio` : sans lui, la pile monte, l'appel s'etablit,
#  et personne n'entend rien. Le chemin `qemu` devant continuer a marcher a
#  l'identique, cette precondition devait survivre au decoupage - on la deplace,
#  on ne la reinvente pas.
#
#  CE FICHIER NE FAIT RIEN AU SOURCE : il ne definit que des fonctions.
#  Appelant attendu : run_modules/25-audio.sh.
#
#  AUDIO=0 desactive toute la mise en place, exactement comme dans l'original.
# ----------------------------------------------------------------------------

: "${HERE:=/opt/GSM/osmo-operator}"
: "${LOG_DIR:=/root}"
: "${AUDIO:=1}"
: "${PULSE_SOCK:=/var/run/pulse/native}"
# L'original colorait ses messages ; sous `set -u` une couleur non definie
# ferait echouer la fonction avant meme d'avoir agi. On les neutralise.
RED='' GREEN='' YELLOW='' CYAN='' NC='' BOLD=''

# Les DEUX null-sinks de la chaine GSM, declares au meme endroit : gsm_audio
# (= alsa gsm_out, la voix qui SORT) et gsm_mic (= alsa gsm_in via
# gsm_mic.monitor, le micro silencieux qui ENTRE).
#
# [2026-08-12] gsm_mic n'etait cree QUE par scripts/pulse-gsm-setup.sh, que
# run.sh appelle ligne 422 - soit APRES osmo-start.sh (ligne 307). Un HLR qui ne
# demarre pas fait `exit 1` dans osmo-start.sh, le `set -euo pipefail` de run.sh
# tue tout, et le sink n'est jamais cree. Or gapk_io ABANDONNE LES DEUX SENS
# quand la capture echoue :
#     pq_alsa.c:168  Couldn't init ALSA device 'gsm_in': Input/output error
#     gapk_io.c:468  Failed to initialize GAPK I/O
# → appel parfaitement etabli mais TOTALEMENT muet, et l'erreur est enterree
```

```
# dans mobile.log. C'est le "ca marche sur un PC, pas sur l'autre" : le tirage
# au sort, c'est de savoir si le HLR est monte avant.
#
# Les deux sinks sont desormais SOLIDAIRES - ici, dans system.pa et dans le
# Dockerfile. Aucun ordre de script ne peut plus en perdre un.
# [2026-09-09] osmo_tts_off : un sink poubelle pour speech-dispatcher. Sur le
# banc, une appli du bureau declenche parfois la synthese vocale ; sans voix
# installee, le module « dummy » de speech-dispatcher joue son message factice
# (dummy-message.wav : 29 s, voix grave, 83 % de l energie sous 300 Hz, niveau
# 100 %) dans les haut-parleurs - par-dessus l appel en cours. Mesure : « voix
# saturee, grave », par intermittence, sur le 600. Une fois le flux deplace ici,
# module-stream-restore le renvoie ici a chaque apparition.
GSM_SINKS="gsm_audio:GSM_Audio gsm_mic:GSM_Mic osmo_tts_off:TTS_off"

load_gsm_sinks() {
    local entry name desc
    for entry in $GSM_SINKS; do
        name="${entry%%:*}"; desc="${entry##*:}"
        pactl list short sinks 2>/dev/null | grep -qw "$name" || \
            pactl load-module module-null-sink sink_name="$name" \
                format=s16le rate=8000 channels=1 \
                sink_properties=device.description="$desc" >/dev/null 2>&1 || true
    done
}

# ── Loopback local : gsm_audio.monitor → la carte son de la machine ──────────
# [2026-08-14] CE MAILLON N'EXISTAIT NULLE PART. Le commentaire de ensure_pulse
# ("en local l'hote entend via le module-loopback vers ses enceintes") le
# SUPPOSE deja present, mais aucun chemin du depot ne le chargeait :
#   - enable_user_loopback() de start-docker.sh n'est appelee par personne ;
#   - network/loopback.sh n'est lance que sciemment, a la main.
# Resultat mesure dans la VM osmo-egprs : gsm_audio RUNNING (mobile ecrit
# dedans), gsm_audio.monitor IDLE (personne ne lit), sortie ALSA SUSPENDED,
# /proc/asound/card0/pcm0p/sub0/status = "closed". Appel etabli, zero son.
# gsm_audio est un module-null-sink : sans consommateur, la voix descendante est
# jetee PAR CONSTRUCTION. Ce n'est pas une panne, c'est un maillon manquant.
#
# ⚠ Le choix du sink n'est pas cosmetique : reboucler sur gsm_audio ou gsm_mic
# recree la boucle fermee du 08/08 (cf. ensure_gapk : "les mobiles ecrivent
# dans gsm_out, on les rejouerait dans leur propre uplink"). On exclut donc
# explicitement les deux null-sinks et on ne garde qu'une sortie materielle.
# [2026-09-09] 40 ms et pas 20. PulseAudio repondait « Configured latency of
# 20.00 ms is smaller than minimum latency, using minimum instead » puis
# « Doing resync » a repetition : la latence minimale de ces sinks est de
# 26 ms, un bouclage qui en demande 20 se resynchronise sans arret et jette
# des echantillons (journal : « drop sink », « drop source »). 40 ms passe
# au-dessus du plancher, et 20 ms de plus sont inaudibles sur un appel.
LOOPBACK_LATENCY_MSEC="${LOOPBACK_LATENCY_MSEC:-40}"

# ── L AFFECTATION DES HAUT-PARLEURS ET DU MICRO : UN SEUL ENDROIT ────────────
# [2026-09-09] TROIS SCRIPTS SE DISPUTAIENT LES MEMES ROUTES. Le son du banc a
# deux modes : SANS telephone postmarketOS, l hote ecoute le descendant par un
# module-loopback direct (gsm_audio.monitor -> haut-parleurs) ; AVEC la VM, la
# voix doit passer PAR ELLE (carte « pont » 0x12 <-> carte « combine » 0x13,
# tools/osmo-pmos-setup.sh) et ce bouclage direct doit disparaitre, sinon on
# entend deux fois - le direct, puis la VM 500 ms plus tard - « voix pourrie ».
# Or scripts/audio-chain.sh (au boot, ExecStartPost de pulseaudio) et
# ensure_pulse (start-native) reposaient le bouclage direct sans regarder si
# la VM etait la, pendant qu osmo-pmos-setup le retirait : le dernier passe
# gagnait. Et chacun choisissait « les haut-parleurs » a sa facon (sink par
# defaut, premiere carte alsa, osmo_hp_ec...), alors que le sink par defaut
# MEMORISE (osmo_hp_ec) n existe pas au boot - l annuleur d echo n arrivait
# qu avec la VM - et que les flux QEMU « combine » se posaient donc sur la
# carte brute, sans annuleur, le temps que le setup les deplace.
#
# Desormais :
```

```
#   audio_hw_sink / audio_hw_source   la carte son materielle (defaut si c en
#                                     est une, sinon la premiere alsa_*, HDMI
#                                     exclu) ;
#   audio_hp_sink / audio_mic_source  CE QUE L OPERATEUR ENTEND / DIT : l
#                                     annuleur d echo (osmo_hp_ec / osmo_mic_ec)
#                                     s il est charge, sinon le materiel ;
#   ensure_echo_cancel                charge cet annuleur (webrtc, sans AGC
#                                     analogique, micro a 25 %) et en fait le
#                                     defaut - au boot comme avec la VM ;
#   pmos_vm_audio_present             la VM postmarketOS a ses cartes son ici
#                                     (flux QEMU media.name=combine) ;
#   remove_direct_loopbacks           retire les bouclages directs ;
#   ensure_local_loopback / _mic      ne posent le direct QUE sans VM, vers
#                                     audio_hp_sink / depuis audio_mic_source,
#                                     et remplacent un bouclage pose vers une
#                                     autre sortie (la carte brute d avant l
#                                     annuleur, par exemple).
# osmo-pmos-qemu charge l annuleur AVANT de lancer QEMU et nomme ses sorties
# a la carte « combine » (OSMO_PMOS_HP / OSMO_PMOS_MIC, patch pmbootstrap) ;
# osmo-pmos-setup et l arret de la VM (pmos_stop) passent par ces fonctions.
# AUDIO_ECHO_CANCEL=0 pour se passer de l annuleur (les HP = la carte brute).
EC_AEC_ARGS='aec_args="analog_gain_control=0 digital_gain_control=1 noise_suppression=1 high_pass_filter=1"'
: "${AUDIO_AEC_METHOD:=speex}"
audio_aec_method() { echo "$AUDIO_AEC_METHOD"; }
# Les aec_args ne valent que pour webrtc (voir ensure_echo_cancel, aec_opts).
: "${EC_MIC_VOLUME:=25%}"
# [2026-10-18] Le micro PDM de l Acer ne sature pas a 100 % (bruit de piece
# ~70 rms) et n est plus qu un murmure a 25-50 % (~6 rms) : son volume a lui.
: "${EC_MIC_VOLUME_PDM:=100%}"

audio_hw_sink() {
    local d; d="$(pactl get-default-sink 2>/dev/null || true)"
    case "$d" in alsa_output.*) echo "$d"; return 0 ;; esac
    pactl list short sinks 2>/dev/null \
        | awk '$2 ~ /^alsa_output/ && $2 !~ /hdmi/ { print $2; exit }'
}
# [2026-10-18] LE MICRO NUMERIQUE INTERNE (PDM/DMIC) AVANT LA PRISE JACK.
# Sur l Acer (acer, 10.99.0.2), la source par defaut etait l entree jack
# analogique (alsa_input.pci-0000_04_00.6.analog-stereo) : rien de branche,
# zero absolu (pas meme du bruit), et l annuleur d echo s y accrochait. Le
# micro interne est la source acp_pdm_mach, laissee SUSPENDED. Resultat :
# gsm_mic muet, la ROM codait la trame de silence et l echo test (600) ne
# renvoyait rien. Le port jack y est en « availability unknown » : pas de
# detection de prise sur laquelle s appuyer, donc on prefere par nom le micro
# numerique interne. AUDIO_MIC_SOURCE=<source> force un autre micro (casque).
audio_hw_source() {
    if [ -n "${AUDIO_MIC_SOURCE:-}" ] && pactl list short sources 2>/dev/null \
            | awk -v s="$AUDIO_MIC_SOURCE" '$2 == s { f=1 } END { exit !f }'; then
        echo "$AUDIO_MIC_SOURCE"; return 0
    fi
    local p; p="$(pactl list short sources 2>/dev/null \
        | awk '$2 ~ /^alsa_input/ && $2 ~ /pdm|dmic/ && $2 !~ /\.monitor$/ { print $2; exit }')"
    [ -n "$p" ] && { echo "$p"; return 0; }
    local d; d="$(pactl get-default-source 2>/dev/null || true)"
    case "$d" in alsa_input.*) echo "$d"; return 0 ;; esac
    pactl list short sources 2>/dev/null \
        | awk '$2 ~ /^alsa_input/ && $2 !~ /\.monitor$/ { print $2; exit }'
}
audio_hp_sink() {
    pactl list short sinks 2>/dev/null | awk '$2 == "osmo_hp_ec" { f=1 } END { exit !f }' \
        && { echo osmo_hp_ec; return 0; }
    audio_hw_sink
}
audio_mic_source() {
    pactl list short sources 2>/dev/null | awk '$2 == "osmo_mic_ec" { f=1 } END { exit !f }' \
        && { echo osmo_mic_ec; return 0; }
```

```
    audio_hw_source
}

# ── QUI PORTE LA VOIX : L HOTE, OU LA VM ? ──────────────────────────────
# [2026-09-09] PAR DEFAUT, L HOTE. Le telephone postmarketOS est l INTERFACE du
# banc (composer, decrocher, les SMS) ; il n a pas besoin de porter les
# echantillons. Le faire coutait cher et s entendait : micro -> QEMU -> Pulse
# de la VM -> module-loopback 400 ms -> QEMU -> gsm_mic, et autant au retour,
# soit ~800 ms d aller-retour, deux rythmes d horloge emules, et le
# rechantillonnage de deux cartes HDA a 44,1 kHz pour un signal telephonique a
# 8 kHz. Mesure et verdict de l operateur : « le son est bon sans pmOS ».
#
# Donc : la voix reste sur l hote (gsm_audio -> haut-parleurs, micro ->
# gsm_mic, 40 ms), et la VM garde ses cartes pour ce qui la regarde - sonnerie,
# notifications, l application Appels. Elle reste dans le trajet de la
# SIGNALISATION, qui est son vrai role.
#
# Pour remettre la voix DANS la VM (l ancien montage croise, si l on veut
# vraiment l entendre traverser le telephone) : OSMO_PMOS_VOIX_VM=1 au
# lancement d osmo-pmos-setup. Il pose alors les deux bouclages dans la VM et
# depose ce marqueur ; les fonctions ci-dessous retirent alors les bouclages
# directs de l hote, sans quoi la voix arriverait DEUX fois - en direct, puis
# par la VM 400 ms plus tard, ce qui etait le defaut du 09/09.
PMOS_VOIX_VM_MARQUEUR="${PMOS_VOIX_VM_MARQUEUR:-/run/osmo-pmos-voix-vm}"

# Vrai seulement si la VM porte VRAIMENT la voix : le marqueur ET le processus.
# Un marqueur oublie par une VM tuee sans menagement ne doit pas priver l hote
# de ses haut-parleurs.
pmos_vm_audio_present() {
    [ -f "$PMOS_VOIX_VM_MARQUEUR" ] || return 1
    pgrep -f 'qemu-system.*hdac[o]mbine' >/dev/null 2>&1 && return 0
    pactl list sink-inputs 2>/dev/null | grep -q 'media.name = "combine"'
}


remove_direct_loopbacks() {
    local m n=0
    for m in $(pactl list short modules 2>/dev/null \
               | awk '/module-loopback/ && (/source=gsm_audio\.monitor/ || /sink=gsm_mic([ \t]|$)/) { print $1
                ↪  }'); do
        pactl unload-module "$m" >/dev/null 2>&1 && n=$(( n + 1 ))
    done
    [ "$n" -gt 0 ] && echo -e "  ${YELLOW}[audio] ${n} bouclage(s) direct(s) retire(s)${NC}"
    return 0
}

# [2026-09-08/09] L ANNULEUR D ECHO ENTRE LE MICRO ET LES HAUT-PARLEURS.
# Micro interne + haut-parleurs : l echo du 600 se reinjectait dans le micro
# (2 400 de crete HP coupe, 32 768 sature HP ouvert) : Larsen. Et l AGC
# analogique de webrtc, laisse par defaut, promenait le gain du micro de 100 %
# a 0 % d une minute a l autre (voix ecrasee, 93 % de l energie sous 300 Hz a
# l entree du GSM). Donc : webrtc sans AGC analogique, micro fixe a 25 % (il
# sature des 45 %), et un demutage explicite - module-device-restore rend
# parfois le module muet. Le volume d osmo_mic_ec est PARTAGE avec le micro
# materiel : meme valeur pour les deux.
ensure_echo_cancel() {
    [ "${AUDIO:-1}" = "1" ] || return 0
    [ "${AUDIO_ECHO_CANCEL:-1}" = "1" ] || {
        echo -e "  ${YELLOW}[audio] annuleur d echo desactive (AUDIO_ECHO_CANCEL=0)${NC}"; return 0; }
    pactl info >/dev/null 2>&1 || return 0
    local mic hp aec_opts=()
    [ "$AUDIO_AEC_METHOD" = "webrtc" ] && aec_opts=("$EC_AEC_ARGS")
    mic="$(audio_hw_source)"; hp="$(audio_hw_sink)"
    if [ -z "$mic" ] || [ -z "$hp" ]; then
        echo -e "  ${YELLOW}[audio] pas de micro ou de haut-parleur materiel - annuleur d echo ignore${NC}";
         ↪  return 0
```

```
    fi
    # Le defaut (osmo_hp_ec / osmo_mic_ec) n est pose qu au CHARGEMENT : le
    # rejouer a chaque passage renvoyait vers l annuleur les flux que l
    # operateur venait de mettre ailleurs. Tout ce qui compte pour la chaine
    # GSM est epingle par nom - le defaut n est plus qu une affaire de bureau.
    if pactl list short modules 2>/dev/null | grep -q 'module-echo-cancel'; then
        echo -e "  ${GREEN}[audio] annuleur d echo deja en place (osmo_hp_ec / osmo_mic_ec)${NC}"
        return 0
    # [2026-09-09] 8000 Hz MONO, COMME TOUT LE RESTE DU BANC. Sans « rate », le
    # module se charge a 32000 Hz : c est ce que montrait « pactl list short
    # sinks » (osmo_hp_ec en float32le 1ch 32000Hz) alors que gsm_audio, gsm_mic
    # et osmo_tts_off sont tous en s16le 1ch 8000Hz. Chaque traversee de
    # l annuleur coutait donc deux reechantillonnages - 8000 -> 32000 a l entree,
    # 32000 -> 8000 a la sortie - visibles dans les flux :
    #     Loopback from Micro_sans_echo ... 1ch 32005Hz  -> sink gsm_mic (8000)
    # Le signal est de toute facon borne a 4 kHz par le GSM : les 32 kHz ne
    # portaient rien de plus. L annuleur webrtc accepte 8000/16000/32000/48000 ;
    # 8000 verifie a la main avant d etre pose ici (module charge, sinks lus en
    # « float32le 1ch 8000Hz », module decharge).
    # [2026-09-23] SPEEX PAR DEFAUT. webrtc (annulation + reduction de bruit +
    # passe-haut) coutait a PulseAudio 7 a 17 % d un coeur en continu, appel ou
    # pas. speex annule l echo pour une fraction de ce prix ; il n a pas l AGC
    # analogique qui faisait pomper le micro (voir plus haut), rien a couper.
    # AUDIO_AEC_METHOD=webrtc retablit l ancien annuleur et ses aec_args.
    # aec_args est UN argument (guillemets compris) : un $(...) non cite le
    # coupait en morceaux, pactl ignorait analog_gain_control=0 et l AGC de
    # webrtc descendait le micro a 0 % (micro muet en appel, 2026-09-23).
    elif pactl load-module module-echo-cancel aec_method="$(audio_aec_method)" \
            rate=8000 channels=1 \
            source_master="$mic" sink_master="$hp" \
            source_name=osmo_mic_ec sink_name=osmo_hp_ec "${aec_opts[@]}" \
            source_properties=device.description=Micro_sans_echo \
            sink_properties=device.description=HP_sans_echo >/dev/null 2>&1; then
        echo -e "  ${GREEN}[audio] annuleur d echo ${AUDIO_AEC_METHOD} pose entre ${mic} et ${hp}${NC}"
    else
        echo -e "  ${YELLOW}[audio] echec de l annuleur d echo - les HP restent ${hp}${NC}"; return 0
    fi
    pactl set-default-sink osmo_hp_ec >/dev/null 2>&1
    pactl set-default-source osmo_mic_ec >/dev/null 2>&1
    pactl set-sink-mute "$hp" 0 2>/dev/null;       pactl set-sink-mute osmo_hp_ec 0 2>/dev/null
    local vol="$EC_MIC_VOLUME"
    case "$mic" in *pdm*|*dmic*) vol="$EC_MIC_VOLUME_PDM" ;; esac
    pactl set-source-mute "$mic" 0 2>/dev/null;    pactl set-source-volume "$mic" "$vol" 2>/dev/null
    pactl set-source-mute osmo_mic_ec 0 2>/dev/null; pactl set-source-volume osmo_mic_ec "$vol" 2>/dev/null
    return 0
}

# ── La source d ENREGISTREMENT : ce qu on entend + ce qu on dit ──────────────
# [2026-09-09] « pourquoi j ai pas l enregistrement ». L enregistreur d ecran
# (extension EasyScreenCast, menu « audio source ») ne sait capter qu une
# SOURCE PulseAudio. Or un appel ne vit pas sur une source : le descendant va
# dans des sinks (gsm_audio, puis les haut-parleurs), et seul le micro est une
# source. Resultat : « No audio source » enregistre le silence, et choisir le
# micro n enregistre que la voix de l operateur, jamais le correspondant.
#
# On fabrique donc UN sink poubelle, osmo_rec, alimente par les deux cotes :
#   - le moniteur des haut-parleurs de l operateur (tout ce qu il ENTEND, que
#     la voix vienne du bouclage direct ou du telephone postmarketOS) ;
#   - son micro (tout ce qu il DIT).
# Son moniteur, osmo_rec.monitor, apparait alors dans le menu de l enregistreur
# sous « Enregistrement_appel » : un seul choix, et l appel entier est dedans.
# 48 kHz stereo : c est une piste video, pas du GSM ; PulseAudio reechantillonne
# les 8 kHz une fois pour toutes, ici, hors du chemin de la voix.
# AUDIO_RECORD_MIX=0 pour ne pas la creer.
# ── ALLEGER PULSEAUDIO ──────────────────────────────────────────────────────
# [2026-09-23] Mesure : PulseAudio a 7-17 % d un coeur en permanence, pour
```

```
#   - speech-dispatcher (sd_dummy) : un flux 44100 Hz jamais suspendu vers la
#     carte son, donc un reechantillonnage 44,1 -> 48 kHz continu pour rien ;
#     il est lance a la demande par le bureau (« -s -t 0 » : ne s arrete jamais) ;
#   - l annuleur webrtc (voir ensure_echo_cancel) ;
#   - osmo_rec en 48 kHz stereo, alimente par deux boucles 8 kHz mono.
# Ici on arrete le premier et on MIGRE un banc deja monte vers speex / 8 kHz :
# decharger l annuleur fait tomber les boucles epinglees dessus, on les repose.
# AUDIO_ALLEGER=0 pour ne rien toucher.
alleger_audio() {
    [ "${AUDIO:-1}" = "1" ] || return 0
    [ "${AUDIO_ALLEGER:-1}" = "1" ] || return 0
    if pgrep -x speech-dispatch >/dev/null 2>&1 || pgrep -f '[/]usr/bin/speech-dispatcher' >/dev/null 2>&1; then
        pkill -f '[/]usr/bin/speech-dispatcher' 2>/dev/null
        pkill -f '[/]usr/lib/speech-dispatcher-modules/' 2>/dev/null
        echo -e "  ${GREEN}[audio] speech-dispatcher arrete (flux 44,1 kHz permanent vers les HP)${NC}"
    fi
    pactl info >/dev/null 2>&1 || return 0
    # [2026-09-23] module-stream-restore SANS restore_device : il renvoyait le
    # `mobile` (ALSA « default ») sur la carte brute ou il avait joue une fois,
    # hors de l annuleur -- echo test : le HP repart dans le micro, Larsen qui
    # monte au fil de l appel. Les volumes restent memorises, pas le peripherique.
    local sr
    sr="$(pactl list short modules 2>/dev/null | awk '$2 == "module-stream-restore" && $0 !~
     ↪  /restore_device=false/ {print $1; exit}')"
    if [ -n "$sr" ]; then
        pactl unload-module "$sr" >/dev/null 2>&1 \
            && pactl load-module module-stream-restore restore_device=false >/dev/null 2>&1 \
            && echo -e "  ${GREEN}[audio] module-stream-restore sans restauration de peripherique${NC}"
    fi
    local m refaire=0
    m="$(pactl list short modules 2>/dev/null | awk '/module-echo-cancel/ && /aec_method=webrtc/ {print $1;
     ↪  exit}')"
    if [ -n "$m" ] && [ "$AUDIO_AEC_METHOD" != "webrtc" ]; then
        pactl unload-module "$m" >/dev/null 2>&1 \
            && echo -e "  ${GREEN}[audio] annuleur webrtc decharge -> ${AUDIO_AEC_METHOD}${NC}"
        refaire=1
    fi
    if pactl list short sinks 2>/dev/null | awk '$2 == "osmo_rec"' | grep -q 48000Hz; then
        for m in $(pactl list short modules 2>/dev/null | awk '/module-loopback/ && /sink=osmo_rec([ \t]|$)/
         ↪  {print $1}') \
                 $(pactl list short modules 2>/dev/null | awk '/module-null-sink/ && /sink_name=osmo_rec([ \t]|$)/
                  ↪  {print $1}'); do
            pactl unload-module "$m" >/dev/null 2>&1
        done
        echo -e "  ${GREEN}[audio] osmo_rec 48 kHz stereo retire -> 8 kHz mono${NC}"
        refaire=1
    fi
    if [ "$refaire" = 1 ]; then
        ensure_echo_cancel
        ensure_local_loopback
        ensure_local_mic
        ensure_record_mix
    fi
    return 0
}

# ── REINIT : repartir d une chaine propre sans relancer PulseAudio ──────────
# [2026-09-23] scripts/audio-chain.sh --reinit. Decharge ce que la chaine a
# pose PAR-DESSUS les sinks GSM -- annuleur d echo, toutes les boucles,
# osmo_rec -- et arrete speech-dispatcher ; audio-chain.sh repose ensuite le
# tout comme au boot. gsm_audio, gsm_mic et osmo_tts_off restent : le
# `mobile` et gapk y sont branches (PCM ALSA gsm_out/gsm_in), les decharger
# couperait un appel pour de bon au lieu d une seconde.
reinit_audio() {
    [ "${AUDIO:-1}" = "1" ] || return 0
    pactl info >/dev/null 2>&1 || return 0
```

```
    pkill -f '[/]usr/bin/speech-dispatcher' 2>/dev/null
    pkill -f '[/]usr/lib/speech-dispatcher-modules/' 2>/dev/null
    local m n=0
    for m in $(pactl list short modules 2>/dev/null | awk '/module-loopback/ {print $1}') \
             $(pactl list short modules 2>/dev/null | awk '/module-echo-cancel/ {print $1}') \
             $(pactl list short modules 2>/dev/null | awk '/module-null-sink/ && /sink_name=osmo_rec([ \t]|$)/
              ↪  {print $1}'); do
        pactl unload-module "$m" >/dev/null 2>&1 && n=$((n + 1))
    done
    echo -e "  ${GREEN}[audio] ${n} module(s) de la chaine decharge(s)${NC}"
}

ensure_record_mix() {
    [ "${AUDIO:-1}" = "1" ] || return 0
    [ "${AUDIO_RECORD_MIX:-1}" = "1" ] || return 0
    pactl info >/dev/null 2>&1 || return 0

    pactl list short sinks 2>/dev/null | grep -qw osmo_rec \
        || pactl load-module module-null-sink sink_name=osmo_rec \
               format=s16le rate=8000 channels=1 \
               sink_properties=device.description=Enregistrement_appel >/dev/null 2>&1 \
        || { echo -e "  ${YELLOW}[audio] sink osmo_rec impossible - enregistrement non prepare${NC}"; return 0; }

    local hp mic src
    hp="$(audio_hp_sink)"; mic="$(audio_mic_source)"
    for src in ${hp:+${hp}.monitor} ${mic:-}; do
        # Idempotent, et epingle comme les autres (cf. le bloc dont_move).
        pactl list short modules 2>/dev/null | grep -F 'module-loopback' \
            | grep -F "source=${src}" | grep -qF 'sink=osmo_rec' && continue
        pactl load-module module-loopback source="$src" sink=osmo_rec \
            sink_dont_move=true source_dont_move=true \
            latency_msec="$LOOPBACK_LATENCY_MSEC" >/dev/null 2>&1
    done
    echo -e "  ${GREEN}[audio] source d enregistrement prete : osmo_rec.monitor (Enregistrement_appel)${NC}"
    # L enregistreur d ecran s en souvient par son nom d application
    # (module-stream-restore) : on le deplace une fois, il y revient ensuite.
    local so
    for so in $(pactl list source-outputs 2>/dev/null \
                | awk '/Source Output #/{id=$3} /application.name = ".*[Ss]creen[Cc]ast|.*Shell.Screencast/{print
                 ↪  id}' | tr -d '#'); do
        pactl move-source-output "$so" osmo_rec.monitor 2>/dev/null \
            && echo -e "  ${GREEN}[audio] enregistreur d ecran bascule sur osmo_rec.monitor${NC}"
    done
    return 0
}

ensure_local_loopback() {
    [ "${AUDIO:-1}" = "1" ] || return 0
    [ "${AUDIO_LOCAL_LOOPBACK:-1}" = "1" ] || {
        echo -e "  ${YELLOW}[audio] loopback local desactive (AUDIO_LOCAL_LOOPBACK=0)${NC}"; return 0; }
    pactl info >/dev/null 2>&1 || return 0

    pactl list short sources 2>/dev/null | grep -qw 'gsm_audio.monitor' || {
        echo -e "  ${YELLOW}[audio] gsm_audio.monitor absent - loopback local ignore${NC}"; return 0; }

    # Avec le telephone postmarketOS, la voix passe par la VM : pas de direct.
    if pmos_vm_audio_present; then
        echo -e "  ${GREEN}[audio] telephone postmarketOS en place - l ecoute passe par la VM, pas de bouclage
         ↪  direct${NC}"
        remove_direct_loopbacks
        return 0
    fi

    local sink
    sink="$(audio_hp_sink)"
    [ -n "$sink" ] || {
```

```
        echo -e "  ${YELLOW}[audio] aucune sortie materielle - loopback local ignore${NC}"; return 0; }

    # Idempotent : ne pas empiler un 2e loopback (voix doublee + echo)...
    if pactl list short modules 2>/dev/null | grep -F 'module-loopback' \
         | grep -F 'source=gsm_audio.monitor' | grep -F "sink=${sink}" | grep -qF 'sink_dont_move=true'; then
        echo -e "  ${GREEN}[audio] loopback local deja en place → ${sink}${NC}"
        return 0
    fi
    # ...et ne pas en garder un vers une AUTRE sortie (la carte brute posee
    # avant l annuleur d echo, par exemple).
    local m
    for m in $(pactl list short modules 2>/dev/null \
                | awk '/module-loopback/ && /source=gsm_audio\.monitor/ { print $1 }'); do
        pactl unload-module "$m" >/dev/null 2>&1
    done

    # [2026-09-09] EPINGLE. PulseAudio 16 deplace tout flux non epingle vers le
    # nouveau peripherique par defaut (set-default-sink / GNOME « Sortie ») :
    # mesure - un clic dans les reglages son et ce bouclage lisait le MICRO
    # (source par defaut) pour le jouer dans la carte brute (sink par defaut).
    # Larsen, plus rien sur gsm_audio, « son pourri ». dont_move le rend sourd
    # aux changements de defaut : il ne bouge que si on le decharge.
    if pactl load-module module-loopback \
            source=gsm_audio.monitor sink="$sink" \
            sink_dont_move=true source_dont_move=true \
            latency_msec="$LOOPBACK_LATENCY_MSEC" >/dev/null 2>&1; then
        echo -e "  ${GREEN}[audio] loopback local charge : gsm_audio.monitor → ${sink} (${LOOPBACK_LATENCY_MSEC}
        ↪  ms)${NC}"
    else
        echo -e "  ${YELLOW}[audio] echec du loopback local vers ${sink}${NC}"
    fi
}

# ── Micro local : la carte son de la machine → gsm_mic ──────────────────
# [2026-08-31] SYMETRIQUE de ensure_local_loopback, et il manquait pareil. Le
# montant (ce que le mobile EMET vers le reseau) part de gsm_in, c'est-a-dire de
# gsm_mic.monitor - un null-sink que PERSONNE n'alimentait depuis la machine.
# Le seul producteur etait la console web : le navigateur capturait le micro,
# poussait les echantillons a /opt/GSM/osmo-egprs-web/server.js, qui les
# reinjectait via `pacat --playback -d gsm_mic`. Chemin mesure le 31/08 :
# 3 a 4 secondes de retard sur la voix montante, alors que TOUT le trajet
# PulseAudio tient sous 100 ms (pacat 45 ms, capture du mobile 0 us, loopback
# descendant 20 ms). Le tampon n'est donc pas dans PulseAudio mais dans le
# transport navigateur -> node -> socket, qu'aucun reglage pulse n'atteint.
# Un module-loopback depuis la carte son supprime ce transport au lieu de
# l'accelerer : le micro alimente gsm_mic directement, en 20 ms.
#
# ⚠ DEUX SOURCES A NE JAMAIS PRENDRE ICI :
#   - gsm_audio.monitor : c'est le DESCENDANT. Le rejouer dans le montant, c'est
#     exactement la boucle fermee du 08/08 (RMS x11 en 10 s) que le commentaire
#     d'/etc/asound.conf documente. On ne garde qu'une entree MATERIELLE.
#   - gsm_mic.monitor : le sink qu'on alimente - il se recopierait sur lui-meme.
#
# ⚠ Boucle ACOUSTIQUE : le descendant sort sur les enceintes (loopback
# ci-dessus) et le micro les reentend. Au casque c'est sans objet ; sur
# haut-parleurs, couper l'un des deux ou baisser le volume.
AUDIO_LOCAL_MIC="${AUDIO_LOCAL_MIC:-1}"

ensure_local_mic() {
    [ "${AUDIO:-1}" = "1" ] || return 0
    [ "$AUDIO_LOCAL_MIC" = "1" ] || {
        echo -e "  ${YELLOW}[audio] micro local desactive (AUDIO_LOCAL_MIC=0)${NC}"; return 0; }
    pactl info >/dev/null 2>&1 || return 0

    pactl list short sinks 2>/dev/null | grep -qw 'gsm_mic' || {
        echo -e "  ${YELLOW}[audio] sink gsm_mic absent - micro local ignore${NC}"; return 0; }
```

```
    # Avec le telephone postmarketOS, c est la VM qui alimente gsm_mic.
    if pmos_vm_audio_present; then
        echo -e "  ${GREEN}[audio] telephone postmarketOS en place - le micro passe par la VM, pas de bouclage
         ↪  direct${NC}"
        remove_direct_loopbacks
        return 0
    fi

    # Entree = le micro de l operateur (annuleur d echo s il est la, sinon le
    # materiel) - jamais un .monitor (voir l avertissement ci-dessus).
    local src
    src="$(audio_mic_source)"
    [ -n "$src" ] || {
        echo -e "  ${YELLOW}[audio] aucune entree materielle - micro local ignore${NC}"; return 0; }

    # Idempotent : un 2e loopback doublerait la voix montante...
    if pactl list short modules 2>/dev/null | grep -F 'module-loopback' \
         | grep -F "source=${src}" | grep -F 'sink=gsm_mic' | grep -qF 'sink_dont_move=true'; then
        echo -e "  ${GREEN}[audio] micro local deja en place : ${src} → gsm_mic${NC}"
        return 0
    fi
    # ...et pas un depuis une AUTRE entree.
    local m
    for m in $(pactl list short modules 2>/dev/null \
               | awk '/module-loopback/ && /sink=gsm_mic([ \t]|$)/ { print $1 }'); do
        pactl unload-module "$m" >/dev/null 2>&1
    done

    if pactl load-module module-loopback \
            source="$src" sink=gsm_mic \
            sink_dont_move=true source_dont_move=true \
            latency_msec="$LOOPBACK_LATENCY_MSEC" >/dev/null 2>&1; then
        echo -e "  ${GREEN}[audio] micro local charge : ${src} → gsm_mic (${LOOPBACK_LATENCY_MSEC} ms)${NC}"
        echo -e "  ${CYAN}       → la console web n'est plus necessaire pour parler.${NC}"
    else
        echo -e "  ${YELLOW}[audio] echec du micro local depuis ${src}${NC}"
    fi
}

# Post-condition : les PCM que `mobile` va reellement ouvrir s'ouvrent-ils ?
# Tester le sink avec `pactl list sinks` ne suffit pas - c'est le mapping ALSA
# de /etc/asound.conf qui casse (gsm_in → gsm_mic.monitor). On ouvre pour de
# vrai, 1 s de chaque cote, et on gueule si ca rate. Sans ca la panne est
# silencieuse jusqu'au premier appel muet.
assert_audio_devices() {
    command -v aplay >/dev/null 2>&1 || return 0
    local ko=0
    timeout 5 aplay  -D gsm_out -f S16_LE -r 8000 -c 1 -d 1 /dev/zero >/dev/null 2>&1 || {
        echo -e "  ${RED}[audio] KO : lecture 'gsm_out' impossible (sink gsm_audio ?)${NC}"; ko=1; }
    timeout 5 arecord -D gsm_in  -f S16_LE -r 8000 -c 1 -d 1 /dev/null  >/dev/null 2>&1 || {
        echo -e "  ${RED}[audio] KO : capture 'gsm_in' impossible (sink gsm_mic ?)${NC}"; ko=1; }
    if [ "$ko" = "1" ]; then
        echo -e "  ${RED}       → gapk_io va echouer et l'appel sera MUET DANS LES DEUX SENS.${NC}"
        echo -e "  ${RED}       → pactl list short sinks  (attendu : gsm_audio ET gsm_mic)${NC}"
        return 1
    fi
    echo -e "  ${GREEN}[audio] gsm_out (lecture) + gsm_in (capture) ouverts - chaine OK${NC}"
    return 0
}

ensure_pulse() {
    [ "${AUDIO:-1}" = "1" ] || { echo -e "  ${YELLOW}[audio] desactive (AUDIO=0)${NC}"; return 0; }

    # 0. Mapping ALSA gsm_out/gsm_in → sink PulseAudio gsm_audio (REQUIS cote hote
    #    en mode natif). Sans /etc/asound.conf, le `mobile` (io-handler gapk,
```

```
#    alsa-output-dev gsm_out) ouvre un PCM ALSA inexistant → "Unknown PCM
#    gsm_out" → l'audio TCH n'est jamais decode vers gsm_audio → silence
#    navigateur. Le sink seul ne suffit pas : ce mapping doit exister.
if [ -f "$HERE/configs/asound.conf" ] && ! cmp -s "$HERE/configs/asound.conf" /etc/asound.conf 2>/dev/null;
 ↪  then
    cp -f "$HERE/configs/asound.conf" /etc/asound.conf \
        && echo -e "  ${GREEN}[audio] /etc/asound.conf deploye (ALSA gsm_out/gsm_in → pulse gsm_audio)${NC}"
fi

export PULSE_SERVER="unix:${PULSE_SOCK}"
if pactl info >/dev/null 2>&1; then
    # PulseAudio deja actif (service osmo-pulse au boot, ou un 'fake' solo
    # precedent). Le sink gsm_audio n'est PAS forcement charge dans CE demon
    # -> on le (re)charge a la volee s'il manque. SANS ca : l'audio ne marchait
    # qu'apres un 'fake' solo (qui avait pose le sink) - c'est le maillon qui
    # manquait a fake+qemu pour avoir l'audio de lui-meme.
    load_gsm_sinks
    # Dedoublonnage : sur une PipeWire/Pulse PARTAGEE entre containers, chaque
    # ensure_pulse charge son propre module-null-sink gsm_audio → doublons.
    # parec (flux /audio du dashboard) lit alors le monitor d'un sink que gapk
    # n'alimente PAS → audio muet pour les clients distants (navigateur/Windows ;
    # en local l'hote entend via le module-loopback vers ses enceintes, d'ou
    # "ca marche sur Linux mais pas sur Windows"). On garde UN seul gsm_audio
    # (le 1er module) et on decharge les suivants.
    # (le dedoublonnage vaut pour gsm_mic autant que pour gsm_audio : un
    #  gsm_mic en double et gapk capture le monitor du mauvais sink)
    local _entry _name
    for _entry in $GSM_SINKS; do
        _name="${_entry%%:*}"
        pactl list short modules 2>/dev/null \
            | awk -v s="$_name" '/module-null-sink/ && $0 ~ ("sink_name=" s "([ \t]|$)") {print $1}' \
            | tail -n +2 \
            | while read -r _m; do [ -n "$_m" ] && pactl unload-module "$_m" >/dev/null 2>&1 \
                && echo -e "  ${YELLOW}[audio] sink ${_name} en double decharge (module $_m)${NC}"; done
    done
    echo -e "  ${GREEN}[audio] PulseAudio deja actif (sinks gsm_audio + gsm_mic uniques assures)${NC}"
    assert_audio_devices || true
    alleger_audio
    ensure_local_loopback
    ensure_local_mic
    ensure_record_mix
    return 0
fi

# 1. Installer pulseaudio si absent (le binaire demon, pas que les clients)
if ! command -v pulseaudio >/dev/null 2>&1; then
    if command -v apt-get >/dev/null 2>&1; then
        echo -e "  ${YELLOW}[audio] installation pulseaudio...${NC}"
        DEBIAN_FRONTEND=noninteractive apt-get install -y -q \
            pulseaudio pulseaudio-utils alsa-utils >/dev/null 2>&1 || true
    fi
fi
command -v pulseaudio >/dev/null 2>&1 || {
    echo -e "  ${YELLOW}[audio] pulseaudio indisponible - audio ignore${NC}"; return 0; }

# 2. Config system.pa : acces anonyme + les DEUX sinks null (idempotent).
#    Les declarer ici plutot que de compter sur un load-module au runtime
#    est ce qui rend la chaine robuste : ils existent des le demarrage du
#    demon, y compris si le demon redemarre tout seul plus tard.
local sp=/etc/pulse/system.pa
if [ -f "$sp" ]; then
    grep -q 'auth-anonymous=1' "$sp" || sed -i \
       's|^load-module module-native-protocol-unix.*|load-module module-native-protocol-unix auth-anonymous=1
        ↪  socket=/var/run/pulse/native|' "$sp"
    local _entry _name _desc
    for _entry in $GSM_SINKS; do
```

```
            _name="${_entry%%:*}"; _desc="${_entry##*:}"
            grep -q "sink_name=${_name}\b" "$sp" || \
                echo "load-module module-null-sink sink_name=${_name} format=s16le rate=8000 channels=1
                 ↪  sink_properties=device.description=${_desc}" >> "$sp"
        done
    fi

    # 3. (Re)demarrer le demon systeme - SERIALISE
    mkdir -p /var/run/pulse "$LOG_DIR"
    # ensure_pulse est appele en parallele (flux principal + ensure_gapk via la
    # session tmux 'gapk') → deux 'pulseaudio --system' se lancaient en meme
    # temps et se disputaient /var/run/pulse/native → "bind(): Address already
    # in use" → le module socket echoue → demon mort → injoignable.
    # flock garantit UN SEUL (re)demarrage a la fois ; un demon residuel detenant
    # un socket perime (parfois sous un autre uid) est tue et le socket efface
    # par root avant le bind.
    (
        flock 9
        if ! pactl info >/dev/null 2>&1; then
            pkill -x pulseaudio 2>/dev/null || true
            local _k=10
            while pgrep -x pulseaudio >/dev/null 2>&1 && [ $_k -gt 0 ]; do sleep 0.3; ((_k--)) || true; done
            pkill -9 -x pulseaudio 2>/dev/null || true
            rm -f /var/run/pulse/pid /var/run/pulse/native 2>/dev/null || true
            chown -R pulse:pulse /var/run/pulse 2>/dev/null || true
            pulseaudio --system --daemonize=yes --disallow-exit --exit-idle-time=-1 \
                --log-target="file:${LOG_DIR}/pulse-system.log" >/dev/null 2>&1 || true
            local r=10
            while [ $r -gt 0 ]; do pactl info >/dev/null 2>&1 && break; sleep 1; ((r--)) || true; done
        fi
    ) 9>/run/osmo-pulse.lock

    if pactl info >/dev/null 2>&1; then
        load_gsm_sinks
        echo -e "  ${GREEN}[audio] PulseAudio pret (sinks gsm_audio + gsm_mic) @ ${PULSE_SOCK}${NC}"
        assert_audio_devices || true
        ensure_local_loopback
        ensure_local_mic
        ensure_record_mix
    else
        echo -e "  ${YELLOW}[audio] PulseAudio injoignable - audio degrade${NC}"
        echo -e "  ${YELLOW}        → voir ${LOG_DIR}/pulse-system.log${NC}"
    fi
}

# ── Bridge audio : osmo-gapk auto (chemin reseau MGW RTP → sink gsm_audio) ────
# gapk auto poll le VTY OsmoMGW (4243) et bridge le RTP de CHAQUE appel vers
# alsa://gsm_out (= sink null PulseAudio gsm_audio). C'est ce maillon qui rend
# l'audio des appels "reseau" (ex: 600 = echo-test Asterisk via MGW) audible
# dans le dashboard web : server.js capte gsm_audio.monitor → /audio (MP3).
# Lance en session tmux DETACHEE 'gapk' : survit aux 'exec' (start-clean.sh /
# tmux attach) des modes qemu/hybride et poll jusqu'a ce que le MGW soit up.
# Idempotent (relance la session), non-fatal. AUDIO=0 → desactive.
# Pont voix conteneur → hote : parec(gsm_audio.monitor) | paplay(--server=relai).
# run.sh (no-process) tente ce pont AVANT que PulseAudio soit pret (course →
# "PulseAudio injoignable apres 30s" → pont jamais lance → VOIX MUETTE dans docker).
# On le (re)lance ICI, apres ensure_pulse (sink gsm_audio garanti up). Idempotent,
# gated sur HOST_AUDIO_RELAY (pose par start-docker.sh = tcp:<gw>:4713).
ensure_host_audio() {
    local relay="${HOST_AUDIO_RELAY:-}"
    [ -n "$relay" ] || return 0
    if ! pactl --server="$relay" info >/dev/null 2>&1; then
        echo -e "  ${YELLOW}[host-audio] relai ${relay} injoignable - pont voix non lance${NC}"; return 0
    fi
    # ── UN SEUL PONT POUR TOUT LE BANC ──────────────────────────────────────
    # [2026-08-31] LE SON ETAIT ENTENDU EN DOUBLE A PARTIR DE DEUX OPERATEURS.
```

```
    # Le `pkill` plus bas s execute DANS le conteneur, qui a son propre espace de
    # PID : il ne voit JAMAIS le pont du voisin. Chaque operateur demarrait donc
    # le sien, et comme ils poussent tous vers le MEME PulseAudio d hote, on
    # entendait l appel deux fois a deux operateurs, trois fois a trois.
    # On interroge donc le SERVEUR PARTAGE, pas la table des processus locale :
    # un client deja connecte sous ce nom veut dire qu un pont tourne quelque
    # part, peu importe dans quel conteneur.
    #
    # ⚠ scripts/run.sh (audio_bridge) porte le MEME garde-fou. Les deux copies
    # doivent garder LE MEME nom de client : c est par ce nom qu elles se voient.
    # C est precisement ce qui manquait ici - le correctif n avait ete pose que
    # sur run.sh, et cette copie-ci lancait un pont ANONYME, donc invisible au
    # garde-fou de l autre, qui en ajoutait un second.
    local _bridge_name="osmo-gsm-bridge"
    if pactl --server="$relay" list clients 2>/dev/null \
         | grep -q "application.name = \"${_bridge_name}\""; then
        echo -e "  ${GREEN}[host-audio] pont deja actif sur l hote (${relay}) - on n en ajoute pas un second${NC}"
        return 0
    fi
    # ── NI DOUBLON AVEC LE LOOPBACK DE L HOTE ──────────────────────────
    # L hote charge deja son module-loopback gsm_audio.monitor -> carte son
    # (ensure_local_loopback, plus haut dans ce fichier). C est le MEME travail
    # que ce pont : meme source, meme sortie. Les deux ensemble, tout est entendu
    # DEUX FOIS - et le symptome survit a l arret d un conteneur, ce qui fait
    # chercher le doublon du mauvais cote. On garde le loopback : il est cote
    # hote et bien plus court (20 ms contre 250 ms pour le pont TCP).
    if pactl --server="$relay" list short modules 2>/dev/null \
         | grep -q 'source=gsm_audio.monitor'; then
        echo -e "  ${GREEN}[host-audio] loopback gsm_audio deja en place sur l hote - pont TCP inutile${NC}"
        return 0
    fi
    pkill -f "paplay --server=${relay}" 2>/dev/null || true          # idempotent (local)
    [ -f /run/host-audio.pid ] && kill -- "-$(cat /run/host-audio.pid)" 2>/dev/null || true
    # parec lit le pulse LOCAL (gsm_audio.monitor) ; paplay pousse vers l'hote.
    # --latency-msec : capture courte (30ms) + lecture TAMPONNEE (250ms) pour
    # absorber la gigue (ordonnancement/TCP/2 horloges pulse) - sinon voix HACHEE.
    # --client-name : c est LUI qui rend ce pont visible au garde-fou ci-dessus,
    # depuis n importe quel conteneur. Sans lui, chacun se croit seul.
    setsid env PULSE_SERVER="unix:${PULSE_SOCK}" sh -c '
      while true; do
        parec -d gsm_audio.monitor --latency-msec=30 --format=s16le --rate=8000 --channels=1 \
          | paplay --server='"${relay}"' --client-name='"${_bridge_name}"' --latency-msec=250 --raw --format=s16le
↪   --rate=8000 --channels=1
        sleep 1
      done' >"${LOG_DIR}/host-audio.log" 2>&1 &
    echo $! > /run/host-audio.pid
    echo -e "  ${GREEN}[host-audio] pont voix parec|paplay → hote (${relay})${NC}"
}

ensure_gapk() {
    [ "${AUDIO:-1}" = "1" ] || return 0
    command -v tmux      >/dev/null 2>&1 || { echo -e "  ${YELLOW}[gapk] tmux absent - bridge audio non
     ↪  lance${NC}"; return 0; }
    command -v osmo-gapk >/dev/null 2>&1 || { echo -e "  ${YELLOW}[gapk] osmo-gapk absent - bridge audio non
     ↪  lance${NC}"; return 0; }
    ensure_pulse   # gapk ecrit dans alsa://gsm_out = sink gsm_audio (idempotent)
    local gapk_sh="/etc/osmocom/gapk-start.sh"; [ -x "$gapk_sh" ] || gapk_sh="$HERE/scripts/gapk-start.sh"
    [ -x "$gapk_sh" ] || { echo -e "  ${YELLOW}[gapk] gapk-start.sh introuvable - bridge audio non lance${NC}";
     ↪  return 0; }
    tmux kill-session -t gapk 2>/dev/null || true
    # [2026-08-10] Le format etait code en dur a "gsmfr", un nom qui n'existe
    # pas dans osmo-gapk : les deux process mouraient a l'analyse des arguments
    # ("Unsupported format: gsmfr") et le pont n'a jamais rien transporte.
    # Le nom du GSM FR est "gsm". Le 3e argument est le peripherique de
    # CAPTURE : il doit rester gsm_in (moniteur du null-sink gsm_mic). Le
    # pointer sur gsm_out rebranche la boucle fermee du 08/08 - les mobiles
```

```
    # ecrivent dans gsm_out, on les rejouerait dans leur propre uplink.
    tmux new-session -d -s gapk \
        "GAPK_ALSA_DEV=gsm_out GAPK_ALSA_DEV_IN=gsm_in PULSE_SERVER=unix:${PULSE_SOCK} bash '$gapk_sh' auto gsm
         ↪  gsm_out gsm_in 2>&1 | tee ${LOG_DIR}/gapk-auto.log"
    # [2026-08-26] LA FENETRE "exited".
    # `tmux new-session -d` rend 0 des que le serveur a pris la commande, pas
    # quand celle-ci tourne. Si gapk-start.sh rend la main tout de suite - format
    # refuse, sink pas encore la, binaire absent - la fenetre reste affichee avec
    # "[exited]" en travers, et le message vert ci-dessous annonce quand meme un
    # pont audio en place. On attend donc une seconde, et on regarde.
    # La session morte est retiree : une fenetre "exited" au milieu des autres
    # fait chercher une panne dans la pile alors que c'est le pont audio, seul,
    # qui n'a pas demarre - et le journal, lui, dit pourquoi.
    sleep 1
    if tmux has-session -t gapk 2>/dev/null && \
       [ "$(tmux list-panes -t gapk -F '#{pane_dead}' 2>/dev/null | head -1)" != "1" ]; then
        echo -e "  ${GREEN}[gapk] auto lance (RTP MGW → sink gsm_audio) - tmux 'gapk', log
         ↪  ${LOG_DIR}/gapk-auto.log${NC}"
    else
        tmux kill-session -t gapk 2>/dev/null || true
        echo -e "  ${YELLOW}[gapk] mort au demarrage - pas de pont audio (fenetre retiree)${NC}"
        echo -e "          ${CYAN}tail -20 ${LOG_DIR}/gapk-auto.log${NC}"
    fi
    ensure_host_audio   # (re)lance le pont voix → hote maintenant que pulse+sink sont up
}

# ============================================================================
# Modes 1 operateur (host loopback + enp0s3) : noproc/faketrx/virtphy/qemu/combine
# ============================================================================
# ============================================================================
# Mode HYBRIDE faketrx-qemu (Approche C) : 2 osmo-bts-trx, 1 coeur
#   BTS#0 = pipeline QEMU INTOUCHE (run.sh : osmo-trx-ipc 5700, ARFCN 514, Calypso)
#   BTS#1 = side-car : osmo-bts-trx (unit-id 6002, base-port 5820/5720) + fake_trx
#           (-P 5720 -p 6720) + trxcon + mobile osmocom-bb (ARFCN 516, IMSI ...0002)
#   Les 2 MS s'enregistrent sur le meme osmo-bsc/MSC/HLR → appel intra-MSC.
#   1 BTS = 1 horloge (clk_s par process osmo-bts-trx) → pas de conflit d'horloge :
#   une seule osmo-bts-trx 2-PHY est IMPOSSIBLE (clk_s partage, reset ping-pong),
#   d'ou 2 process distincts. On NE TOUCHE NI qosmo-grgsm/run.sh NI osmo-bts-trx.cfg.
# ============================================================================
```

3.228 /opt/GSM/osmo-operator/lib/gabarits.sh

29733 octets, 516 lignes → 516 lignes

```
# ============================================================================
#  lib/gabarits.sh - le moteur de gabarits de configuration d'osmo-operator
# ============================================================================
#
#  CE FICHIER EST UNE EXTRACTION, PAS UNE REECRITURE.
#  Les fonctions ci-dessous viennent telles quelles de start-direct.sh.legacy
#  (L65-76, L106-258, L499-511). Elles n'ont pas ete retouchees : le jour ou
#  un gabarit change, on veut pouvoir comparer ligne a ligne avec l'ancien.
#
#  POURQUOI LES SORTIR DU POINT D'ENTREE.
#  Elles sont la SEULE chose que start-native.sh faisait et que personne
#  d'autre ne fait : substituer __ENCRYPTION__, __MCC__, __KI__, __ARFCN__ ...
#  dans configs/*.cfg puis les deposer dans /etc/osmocom, /etc/asterisk et
#  ~/.osmocom/bb. Sans elles, les demons du coeur demarrent sur les fichiers
#  qu'un run precedent a laisses - et `ENCRYPTION=a5 0` n'a aucun effet, en
#  silence. Le reste de start-native.sh (lancer vingt processus) est repris
#  par le moteur ; ceci ne l'est pas, donc on le preserve a l'identique et on
#  le rend appelable depuis un module.
#
#  CE FICHIER NE FAIT RIEN AU SOURCE : il ne definit que des fonctions.
```

```
#  Le seul appelant attendu est run_modules/08-gabarits.sh.
#
#  DEPENDANCES D'ENVIRONNEMENT (posees par l'appelant, valeurs de l'ancien
#  script en defaut) : ENCRYPTION, HOST_IP, ALSA_OUTPUT, ALSA_INPUT.
#  Le repertoire courant doit etre celui d'osmo-operator : les fonctions lisent
#  `configs/*.cfg` et `scripts/*` en chemin RELATIF, comme dans l'original.
# ------------------------------------------------------------------------------

: "${ENCRYPTION:=a5 0}"
: "${HOST_IP:=127.0.0.1}"
: "${ALSA_OUTPUT:=default}"
: "${ALSA_INPUT:=default}"

# Ces fonctions ont une JUMELLE dans start-docker.sh : elles doivent dire la meme
# chose. C'est CETTE copie que lit une machine qui reclone le depot a chaque
# demarrage - donc celle qui compte sur une ISO. Tant qu'elle a garde l'ancien
# plan prive (172.20.<op>.x), les configs regenerees au boot liaient
# 172.20.1.10, une adresse que plus rien ne pose : osmo-ggsn et osmo-sgsn
# refusaient de demarrer ("adresse introuvable localement") et osmo-pcu tombait
# avec eux.
#
#   backbone  172.20.0.<10+op>   le segment partage avec l'inter-STP (.10)
#   prive     192.168.<op+1>.x   un segment par operateur ; le +1 laisse
#                                192.168.1.0/24 au LAN du banc.
op_backbone_ip()  { echo "172.20.0.$((10 + $1))"; }
# ── L'INDEX DU SEGMENT PRIVE : LE NOEUD OU L'OPERATEUR, SELON L'HOTE ────────
# 192.168.<index+1>.x, et tout le desaccord tenait a « index ».
#
#   DOCKER   un hote porte N conteneurs operateurs, chacun dans son netns. Ce
#            qui les distingue est le NUMERO D'OPERATEUR ; le noeud, lui, est
#            commun a tous les conteneurs de la machine. index = operateur.
#   VM/NATIF la machine EST le noeud et ne porte qu'un operateur. Ce qui la
#            distingue de ses voisines est le NUMERO DE NOEUD. index = noeud.
#
# Sans cette distinction, qosmo-grgsm/run_modules/08-gabarits.sh appelait
# op_private_ip($OPERATOR_ID) et ecrivait 192.168.2.10 dans osmo-ggsn.cfg sur
# TOUTES les VM - operateur 1 partout - pendant que le plan de l'ISO reservait
# 192.168.<noeud+1>.x. Sur le noeud 1 les deux coincidaient et rien ne se
# voyait ; sur le noeud 2, le GGSN se liait a l'adresse du noeud 1.
# La bascule est ici, une fois, et les deux jumelles (start-docker.sh, lib/gabarits.sh)
# en heritent.
_osmo_priv_index() {
    local op="${1:-1}"
    # [2026-08-31] L ORCHESTRATEUR N EST PAS DANS LE CONTENEUR QU IL CREE.
    # La detection ci-dessous demande "suis-je DANS un conteneur ?" et c est la
    # bonne question pour du code qui tourne a l interieur. Mais start-docker.sh, lui,
    # provisionne les conteneurs DEPUIS L HOTE : il prenait donc la branche
    # VM/natif et rendait l index du NOEUD - 1 - pour tous les operateurs.
    # Les deux reseaux prives se retrouvaient sur 192.168.2.0/24, docker
    # refusait le second pour chevauchement, et la boucle s arretait net :
    #     ── Operateur 2 : ... Prive : 192.168.2.10
    #     ✗ Echec creation reseau gsm-net-op2 (192.168.2.0/24)
    # osmo-operator-2 n etait jamais cree, sans qu aucun message ne parle de
    # collision d adresses.
    # OSMO_PRIV_BY_OP=1 dit "je provisionne par operateur" ; start_bridge_mode
    # la pose. C est la seule chose que la detection ne pouvait pas deviner.
    [ "${OSMO_PRIV_BY_OP:-0}" = "1" ] && { printf '%s' "$op"; return; }
    # Meme detection que start-native.sh : le couple /.dockerenv +
    # /etc/docker-entrypoint-cmd identifie un conteneur DE CE DEPOT ; le cgroup
    # sert de repli pour un conteneur quelconque.
    if [ -f /.dockerenv ] || grep -qa 'docker\|containerd' /proc/1/cgroup 2>/dev/null; then
        printf '%s' "$op"; return
    fi
    local n="${OSMO_WAN_NODE:-${WAN_NODE_ID:-}}"
    [ -n "$n" ] || n="$(awk -F= '/^OSMO_WAN_NODE=/{gsub(/[ \r\t]/,"",$2);v=$2} END{print v}' \
                        "${ROLE_FILE:-/etc/osmo-role}" 2>/dev/null)"
```

```
    [ -n "$n" ] || n="$(sed -n 's/^PLAN_NODE=//p' "${OSMOCOM_CFG:-/etc/osmocom}/radio-plan.env" 2>/dev/null | tail
    ↪  -1)"
    case "$n" in [1-9]) ;; *) n=1 ;; esac
    printf '%s' "$n"
}
op_private_ip()   { echo "192.168.$(($(_osmo_priv_index "$1") + 1)).10"; }
op_private_gw()   { echo "192.168.$(($(_osmo_priv_index "$1") + 1)).1"; }
op_private_net()  { echo "192.168.$(($(_osmo_priv_index "$1") + 1)).0/24"; }
op_netns()        { echo "osmo-op$1"; }
op_rctx_msc()     { echo $(( $1 * 100 + 10 )); }
op_rctx_stp()     { echo $(( $1 * 100 + 20 )); }
op_rctx_bsc()     { echo $(( $1 * 100 + 30 )); }
op_rctx_inter()   { echo $(( $1 * 100 + 50 )); }
# ── PLUS AUCUN DECALAGE DE PORTS ──────────────────────────────────────
# [2026-08-31] Il y avait ici une detection : "5060 est-il pris ? alors decale
# les publications de +100". Elle ne servait qu a une chose - eviter que le
# 5060 publie d un conteneur n entre en conflit avec l Asterisk du natif.
#
# On ne publie PLUS aucun port SIP/RTP pour les conteneurs : ils sont sur un
# bridge routable et se joignent en direct, a leur propre adresse, sur le port
# nominal (verifie : 172.20.0.12:5060 et 172.20.0.13:5060 repondent 401 depuis
# l hote, sans la moindre publication). Le conflit ayant disparu avec sa cause,
# le decalage n a plus d objet - et il tirait avec lui une sonde instantanee,
# donc une course : lancee juste apres l arret du banc, elle voyait 5060 libre,
# ne decalait rien, et le conteneur mourait quand le natif remontait.
#
# Les fonctions ci-dessous gardent leur forme (generate_configs.sh s en sert
# encore pour ecrire les configs) mais rendent desormais les valeurs NOMINALES.
# OSMO_PORT_OFFSET=N reste disponible pour qui voudrait retablir un decalage.
PORT_OFFSET="${OSMO_PORT_OFFSET:-0}"
# Le RTP suit la meme regle : plage nominale, 200 ports par operateur.
linphone_sip_port()  { echo $(( 5060  + ${PORT_OFFSET:-0}      + ($1 - 1) )); }
linphone_rtp_start() { echo $(( 30000 + ${PORT_OFFSET:-0} * 10 + ($1 - 1) * 200 )); }
linphone_rtp_end()   { echo $(( 30000 + ${PORT_OFFSET:-0} * 10 + $1 * 200 - 1 )); }
# [2026-08-31] Les boucles de ce fichier enumeraient 1..N en dur, comme celles
# de start-docker.sh avant leur decalage. Avec OP_ID_BASE=2 - les conteneurs prenant
# les rangs 2 et 3, le 1 restant au natif - l operateur 2 se fabriquait donc un
# trunk vers l "operateur 1" a 172.20.0.11, une adresse de dorsale que PERSONNE
# ne porte (le natif est sur l hote), et AUCUN trunk vers l operateur 3.
# Les appels inter-operateurs entre conteneurs ne pouvaient pas aboutir, et
# rien ne le disait : le trunk existe, il pointe simplement dans le vide.
# Les deux jumelles doivent donc partager la meme base.
generate_pjsip_interop_trunks() {
    local op_id=$1 n_operators=$2 remote_op remote_ip
    # ── LES OPERATEURS SOUS LA BASE SONT LES NATIFS ──────────────────
    # [2026-08-31] La boucle partait DE la base. Avec OP_ID_BASE=2 - les
    # conteneurs prenant les rangs 2..N, le 1 restant au natif sur l hote -
    # op2 ne fabriquait qu un trunk vers op3 : RIEN vers le natif. Un appel
    # 100201 -> 100101 tombait dans le fourre-tout _X. de [interop_out] et
    # sortait en Congestion, sans qu une ligne dise que la route n existe pas.
    # Meme trou que sms-routing.conf, sur le chemin VOIX.
    for remote_op in $(seq 1 $(( ${OP_ID_BASE:-1} - 1 ))) \
                     $(seq "${OP_ID_BASE:-1}" "$(( ${OP_ID_BASE:-1} + n_operators - 1 ))"); do
        [ "$remote_op" -eq "$op_id" ] && continue
        # Un operateur sous la base est NATIF : il tourne sur l hote, pas dans
        # un conteneur. Vu d un conteneur, l hote est la passerelle du bridge
        # docker - la meme adresse que le pont audio et que sms-routing.conf.
        # NB : match=172.20.0.1 n avale pas les REGISTER Linphone, qui arrivent
        # par la meme passerelle : pjsip.conf pose endpoint_identifier_order=
        # username,ip,anonymous - le username passe avant l IP.
        if [ "$remote_op" -lt "${OP_ID_BASE:-1}" ]; then
            remote_ip="${INTER_NET_GATEWAY:-172.20.0.1}"
        else
            remote_ip=$(op_backbone_ip "$remote_op")
        fi
        cat <<EOF
```

```
[interop-identify-op${remote_op}]
type=identify
endpoint=interop_trunk_op${remote_op}
match=${remote_ip}

[interop_trunk_op${remote_op}]
type=endpoint
transport=transport-udp
context=interop_in
disallow=all
allow=gsm
allow=ulaw
aors=interop_trunk_op${remote_op}
direct_media=no
rtp_symmetric=yes
force_rport=yes
media_encryption=no

[interop_trunk_op${remote_op}]
type=aor
contact=sip:${remote_ip}:5060
qualify_frequency=15
qualify_timeout=5.0
EOF
    done
}
generate_extensions_interop_out() {
    local op_id=$1 n_operators=$2 remote_op
    # Le prefixe porte le NOEUD : MSISDN = <noeud>00<operateur><rang>. Fige a
    # "600", le motif ne matchait plus rien des que le numero commencait par le
    # numero de noeud - les appels inter-operateurs sortaient en Congestion
    # sans qu'aucune ligne ne dise que c'est le motif qui n'accroche pas.
    local _pfx; _pfx="$(osmo_msisdn_pfx "$(osmo_node_id)")"
    printf '[interop_out]\n\n'
    # Un seul motif : les MSISDN font six chiffres, <noeud>00<operateur><rang>. Les
    # deux motifs _<op>XXXX / _<op>XXXXX visaient l'ancien plan a cinq chiffres,
    # ou le premier chiffre du numero ETAIT l'operateur - plus rien ne matchait.
    # ── LES OPERATEURS SOUS LA BASE SONT LES NATIFS ──────────────────────
    # [2026-08-31] La boucle partait DE la base. Avec OP_ID_BASE=2 - les
    # conteneurs prenant les rangs 2..N, le 1 restant au natif sur l hote -
    # op2 ne fabriquait qu un trunk vers op3 : RIEN vers le natif. Un appel
    # 100201 -> 100101 tombait dans le fourre-tout _X. de [interop_out] et
    # sortait en Congestion, sans qu une ligne dise que la route n existe pas.
    # Meme trou que sms-routing.conf, sur le chemin VOIX.
    for remote_op in $(seq 1 $(( ${OP_ID_BASE:-1} - 1 ))) \
                     $(seq "${OP_ID_BASE:-1}" "$(( ${OP_ID_BASE:-1} + n_operators - 1 ))"); do
        [ "$remote_op" -eq "$op_id" ] && continue
        cat <<EOF
exten => _${_pfx}${remote_op}XX,1,NoOp(=== INTEROP OUT Op${remote_op}: \${EXTEN} ===)
 same => n,Dial(PJSIP/\${EXTEN}@interop_trunk_op${remote_op},,rT)
 same => n,Congestion()
 same => n,Hangup()

EOF
    done

    # ── LES SOFTPHONES AUSSI, ET SANS PREFIXE ────────────────────────────
    # [2026-08-31] Depuis [internal], un Linphone ne pouvait joindre un autre
    # operateur qu'en composant 9 + numero (_9. du gabarit - dont l'exemple
    # « 920001 pour joindre 20001 sur Op2 » date d'ailleurs du plan a CINQ
    # chiffres, abandonne). Compose tel quel, 100201 tombait sur le fourre-tout
    # _X. de [internal] et partait au MSC LOCAL, qui le rendait en
    #     rx MNCC_SETUP_REQ for unknown subscriber number '100201'
    # avant de le ressortir par MNCC vers Asterisk. Le detour aboutissait, mais
    # il consommait une transaction MNCC pour rien et faisait dependre un appel
```

```
    # SIP->SIP du bon vouloir du MSC.
    # On declare donc la route explicitement. _<pfx><op>XX est PLUS SPECIFIQUE
    # que _X. : Asterisk la choisit d'office, sans qu'il faille toucher au
    # fourre-tout du gabarit. Gosub(sub-record) comme pour _9., sinon ces
    # appels-la seraient les seuls a ne pas etre enregistres.
    printf '[internal]\n\n'
    for remote_op in $(seq 1 $(( ${OP_ID_BASE:-1} - 1 ))) \
                     $(seq "${OP_ID_BASE:-1}" "$(( ${OP_ID_BASE:-1} + n_operators - 1 ))"); do
        [ "$remote_op" -eq "$op_id" ] && continue
        cat <<EOF
exten => _${_pfx}${remote_op}XX,1,NoOp(=== SIP -> INTER-OP Op${remote_op}: \${EXTEN} - CallerID:
 ↪  \${CALLERID(all)} ===)
 same => n,Gosub(sub-record,s,1(\${EXTEN}))
 same => n,Goto(interop_out,\${EXTEN},1)

EOF
    done
    # PAS de fourre-tout _X. ici : le gabarit configs/extensions.conf en declare
    # deja un dans ce meme contexte. Deux extensions identiques dans un contexte
    # font refuser la seconde, avec six avertissements a chaque chargement :
    #     add_priority: Unable to register extension '_X.' priority 1
    #                   in 'interop_out', already in use
    # Rien ne cassait - le fourre-tout du gabarit restait en place - mais ces
    # lignes noyaient celles qui comptent.
}
_generate_sms_routing_conf_fallback() {
    # 'local i' n'est PAS cosmetique. Cette fonction est appelee, via
    # apply_config_templates, DEPUIS la boucle "for i in ..." qui demarre les
    # operateurs. Sans local, sa propre boucle ecrase la variable de l'appelant
    # et la laisse a n_operators : tous les conteneurs suivants calculaient
    # alors leurs ports avec le MEME indice, d'ou
    #     Bind for 0.0.0.0:5082 failed: port is already allocated
    # au deuxieme conteneur - et, avant l'erreur, deux operateurs annonces sur
    # les memes SIP/RTP/SMS.
    local i
    local op_id=$1 n_operators=$2 i j
    printf '# sms-routing.conf - Fallback\n\n[local]\noperator_id = %s\nsc_address  =
     ↪  1999001%s444\n\n[operators]\n' "$op_id" "$op_id"
    # ── LES OPERATEURS SOUS LA BASE SONT LES NATIFS ──────────────────────
    # [2026-08-31] start-multi.sh pose OP_ID_BASE=2 : les conteneurs prennent
    # les rangs 2..N et l'operateur 1 reste NATIF, sur l'hote. Comme la boucle
    # part de OP_ID_BASE, il n'apparaissait ni dans [operators] ni dans
    # [routes] - releve dans osmo-operator-2 : « 2 = ... / 3 = ... » et rien
    # pour le 1. Un SMS vers 100101 sortait donc en « No route for
    # destination », et le natif etait injoignable depuis tout le banc.
    # Il existe pourtant : il n'est simplement pas dans un conteneur. On le
    # pointe sur la passerelle du bridge docker, qui EST l'hote vu d'un
    # conteneur - la meme adresse que le pont audio (172.20.0.1).
    local _natif_gw="${INTER_NET_GATEWAY:-172.20.0.1}"
    for i in $(seq 1 $(( ${OP_ID_BASE:-1} - 1 ))); do printf '%s = %s\n' "$i" "$_natif_gw"; done
    for i in $(seq "${OP_ID_BASE:-1}" "$(( ${OP_ID_BASE:-1} + n_operators - 1 ))"); do printf '%s = %s\n' "$i"
     ↪  "$(op_backbone_ip "$i")"; done
    printf '\n[routes]\n'
    for i in $(seq 1 $(( ${OP_ID_BASE:-1} - 1 ))); do
        for ms in 1 2; do printf '%s = %s\n' "$(osmo_msisdn "$(osmo_node_id)" "$i" "$ms")" "$i"; done
    done
    for i in $(seq "${OP_ID_BASE:-1}" "$(( ${OP_ID_BASE:-1} + n_operators - 1 ))"); do
        # Les MSISDN EXACTS, pas un prefixe : un prefixe trop court avalait
        # les numeros voisins, un prefixe absent laissait le relais rejeter
        # tout SMS local avec "No route for destination" (2026-07-29).
        for ms in 1 2; do printf '%s = %s\n' "$(osmo_msisdn "$(osmo_node_id)" "$i" "$ms")" "$i"; done   # MSISDN
         ↪  exacts <noeud>00<op><ms> (100101, 100102, 200101...)
    done
    printf '\n[relay]\nport = 7890\nconnect_timeout = 10\nretry_count = 3\nretry_delay = 5\n'
}
# [2026-08-03] apply_config_templates() a demenage dans generate_configs.sh :
```

```
# ce fichier en portait une copie, start-docker.sh une autre. Une seule desormais, et
# ses valeurs (ARFCN, BSIC, KI, IMSI...) sont exposees dans globals.conf au lieu
# d'etre des formules enfouies dans la substitution.
_GAB_ROOT="$(cd "$(dirname "${BASH_SOURCE[0]}")/.." && pwd)"
[ -r "$_GAB_ROOT/generate_configs.sh" ] && . "$_GAB_ROOT/generate_configs.sh"

# ── Install configs natif. $1=src $2=prefix racine (/ ou /etc/netns/<ns>) ──
# ── LES ADRESSES DE CONTENEUR N ONT RIEN A FAIRE SUR L HOTE ─────────────────
# [2026-08-31] Les gabarits sont ecrits pour un CONTENEUR : ils posent
# l adresse de dorsale de l operateur (172.20.0.<10+N>) et la passerelle du
# bridge docker (172.20.0.1). Sur l hote, aucune des deux n existe.
#
# Ces corrections vivaient dans apply_native_post_patches (generate_configs.sh),
# que start-native.sh appelle. Mais qosmo-grgsm/run_modules/08-gabarits.sh, lui,
# n appelle QUE apply_config_templates + install_configs_native, et ne source
# meme pas generate_configs.sh : il reappliquait donc les gabarits bruts a
# chaque demarrage du natif et REMETTAIT les adresses de conteneur. Le symptome
# etait un natif affiche en 172.20.0.11 et un ASP qui ne montait jamais, une
# seconde apres qu on ait corrige le fichier a la main.
# (Ce module ne s executait plus depuis un renommage de depot, ce qui cachait
# le probleme derriere un [SKIP] : il est reparu en le reparant.)
#
# On accroche donc ici, dans install_configs_native - le passage OBLIGE du
# natif, quel que soit l appelant. Idempotent : rejouer ne change rien.
_gab_fixups_natifs() {
    local src="$1" f

    # 1. osmo-stp.cfg : le local-ip de l ASP vers l inter-STP. 127.0.0.1 est la
    #    seule adresse dont on soit certain sur un hote. On ne touche QUE le
    #    local-ip qui suit « asp asp-to-inter » : ceux du bloc d ecoute
    #    au-dessus (127.0.0.1 / 127.0.0.2) sont corrects.
    #    [2026-10-15] SAUF si le hub est sur une adresse reseau, meme la notre :
    #    une SCTP liee a 127.0.0.1 vers 192.168.1.31 est refusee (« Network is
    #    unreachable »), as-inter reste AS_DOWN. On prend l adresse source que
    #    le noyau choisit vers le hub (meme regle que generate_configs.sh).
    f="$src/osmocom/osmo-stp.cfg"
    if [ -f "$f" ]; then
        local rip lip=127.0.0.1
        rip="$(awk '/^[[:space:]]*asp[[:space:]]+asp-to-inter[[:space:]]/{x=1;next}
         ↪  x&&/^[[:space:]]*remote-ip[[:space:]]/{print $2; exit} x&&/^ (asp|as) /{exit}' "$f")"
        case "$rip" in
            ""|127.*|__*) ;;   # __* : gabarit pas encore substitue
            *) lip="$(ip -4 route get "$rip" 2>/dev/null | awk '{for(i=1;i<NF;i++) if($i=="src") {print $(i+1);
             ↪  exit}}')"
               [ -n "$lip" ] || lip=127.0.0.1 ;;
        esac
        awk -v lip="$lip" '
            /^[[:space:]]*asp[[:space:]]+asp-to-inter[[:space:]]/ { a = 1; print; next }
            a && /^[[:space:]]*local-ip[[:space:]]/ {
                sub(/local-ip[[:space:]]+.*/, "local-ip " lip); a = 0; print; next
            }
            /^[[:space:]]*(as|asp|cs7|listen)[[:space:]]/ && !/asp-to-inter/ { a = 0 }
            { print }
        ' "$f" > "$f.tmp" && mv -f "$f.tmp" "$f"
    fi

    # 2. mobile.cfg : la cible GSMTAP. 172.20.0.1 disparait avec le bridge, et
    #    les paquets partent alors dans le vide - Wireshark reste vide sur
    #    udp/4729 sans qu aucun message ne le dise.
    for f in "$src/osmocom/mobile.cfg" "$src/bb/mobile.cfg" "$src/bb/mobile_group1.cfg"; do
        [ -f "$f" ] || continue
        sed -i "/^gsmtap\$/,/^!\$/ s/^\([[:space:]]*remote-host[[:space:]]\+\).*/\1127.0.0.1/" "$f"
    done

    # 3. pjsip.conf : sur un hote plat il n y a pas de frontiere NAT a declarer.
    #    external_* fait REECRIRE Contact et SDP pour tout pair hors local_net -
```

```
#    un Linphone du LAN recevait une adresse injoignable. Sans eux, pjsip
#    repond avec l adresse de la socket, la bonne, par destination.
f="$src/asterisk/pjsip.conf"
if [ -f "$f" ]; then
    awk '
        /^\[transport-udp\]/ { t = 1; print; next }
        t && /^\[/           { t = 0 }
        t && /^[[:space:]]*external_(media|signaling)_address[[:space:]]*=/ { next }
        t && /^[[:space:]]*local_net[[:space:]]*=/ {
            if (!d) {
                print "local_net=127.0.0.0/8"; print "local_net=10.0.0.0/8"
                print "local_net=172.16.0.0/12"; print "local_net=192.168.0.0/16"
                print "local_net=176.16.32.0/24"; d = 1
            }
            next
        }
        { print }
    ' "$f" > "$f.tmp" && mv -f "$f.tmp" "$f"
fi

# 4. sms-routing.conf : LE NATIF NE CONNAISSAIT QUE LUI-MEME.
#    [2026-08-31] Releve sur l hote :
#        [operators]   1 = 172.20.0.11
#        [routes]      100101 = 1   100102 = 1
#    Deux erreurs dans quatre lignes. 172.20.0.11 est l adresse de DORSALE
#    DOCKER de l operateur 1 - elle n existe pas sur l hote, ou le natif
#    tourne. Et aucun des autres operateurs n y figure : un SMS du natif
#    vers 100201 sortait en « No route for destination ».
#
#    POURQUOI CE FICHIER ETAIT FAUX. apply_native_post_patches
#    (generate_configs.sh) sait deja ecrire cette table - son propre
#    commentaire decrit exactement le defaut. Mais elle n est appelee que
#    par start-native.sh --regen et build-iso.sh ; le demarrage ordinaire du
#    natif passe par 08-gabarits.sh, qui ne fait qu appliquer les gabarits
#    bruts. Le fichier repartait donc a chaque fois sur le fallback ecrit
#    POUR UN CONTENEUR - le meme piege que les points 1 a 3 ci-dessus, d ou
#    la correction au meme endroit : install_configs_native est le passage
#    OBLIGE du natif, quel que soit l appelant.
#
#    LA SOURCE DE VERITE est osmo-multi.conf (ecrit par addition.sh) :
#        MULTI_OPS="1:native::1.1.2:150 2:docker:172.20.0.12:... 3:docker:..."
#    On garde le keying par OPERATEUR, celui des conteneurs : sur un banc a
#    une machine tout le monde est sur le noeud 1, un keying par noeud les
#    confondrait. Sans osmo-multi.conf (mono-operateur), on ne touche a rien.
f="$src/osmocom/sms-routing.conf"
local mc="${OSMO_MULTI_CONF:-/etc/osmocom/osmo-multi.conf}"
if [ -f "$f" ] && [ -r "$mc" ]; then
    local _ops _node _self _sc _id _mode _ip _e
    _ops="$(sed -n 's/^MULTI_OPS=//p' "$mc" | tr -d '"' | tail -1)"
    _node="$(sed -n 's/^MULTI_NODE=//p' "$mc" | tail -1)"; case "$_node" in [1-9]) ;; *) _node=1 ;; esac
    # Identite : on la RELIT, on ne la recalcule pas - c est elle qui dit
    # quel operateur ce natif est, et elle a pu etre realignee par
    # start-multi.sh (aligner_natif).
    _self="$(sed -n 's/^operator_id[[:space:]]*=[[:space:]]*//p' "$f" | head -1)"
    _sc="$(sed -n 's/^sc_address[[:space:]]*=[[:space:]]*//p'  "$f" | head -1)"
    case "$_self" in [1-9]*) ;; *) _self=1 ;; esac
    [ -n "$_sc" ] || _sc="1999001${_self}444"
    if [ -n "$_ops" ]; then
        {
            printf '# sms-routing.conf - natif, reecrit par _gab_fixups_natifs\n'
            printf '# Table issue de %s (MULTI_OPS). Cle = numero d operateur.\n' "$mc"
            printf '# MSISDN = <noeud>00<operateur><rang du MS sur 2 chiffres>\n\n'
            printf '[local]\noperator_id = %s\nsc_address  = %s\n\n[operators]\n' "$_self" "$_sc"
            for _e in $_ops; do
                IFS=: read -r _id _mode _ip _ <<< "$_e"
                [ -n "$_id" ] || continue
```

```
                # Le natif, c est NOUS : 127.0.0.1 est la seule adresse dont
                # on soit certain sur un hote (meme choix qu au point 1).
                if [ "$_mode" = native ] || [ "$_id" = "$_self" ]; then _ip=127.0.0.1; fi
                [ -n "$_ip" ] || continue
                printf '%s = %s\n' "$_id" "$_ip"
            done
            printf '\n[routes]\n'
            for _e in $_ops; do
                IFS=: read -r _id _mode _ip _ <<< "$_e"
                [ -n "$_id" ] || continue
                for _m in 1 2; do
                    printf '%s = %s\n' "$(( _node * 100000 + _id * 100 + _m ))" "$_id"
                done
            done
            printf '\n[relay]\nport = 7890\nconnect_timeout = 10\nretry_count = 3\nretry_delay = 5\n'
        } > "$f.tmp" && mv -f "$f.tmp" "$f"
    fi
fi

# 5. LE CHEMIN VOIX INTER-OPERATEUR : trunks SIP + [interop_out].
#    [2026-08-31] Meme trou que le point 4, sur la VOIX. Releve sur l hote :
#        [interop_out] du natif :  _X.  ->  Congestion()   (et rien d autre)
#        pjsip.conf             :  aucun interop_trunk_opN
#    Un appel 100101 -> 100201 sortait donc du MSC ...
#        rx MNCC_SETUP_REQ for unknown subscriber number '100201'
#    ... passait a osmo-sip-connector, tombait sur le fourre-tout, et
#    revenait en
#        INVITE got status(503), releasing leg
#    Les SMS passaient (point 4), pas les appels : deux chemins, deux tables,
#    une seule des deux corrigee.
#
#    On NE REECRIT PAS la logique : generate_extensions_interop_out et
#    generate_pjsip_interop_trunks (plus haut dans ce fichier) savent deja
#    la produire. Elles etaient seulement appelees avec le nombre
#    d operateurs du CONTENEUR courant - le natif se croyait donc seul.
#    On les rejoue ici avec la topologie reelle, comme au point 4.
#    OP_ID_BASE=1 : vu du natif, tous les distants sont des conteneurs et
#    portent bien une adresse de dorsale ; c est l inverse du cas conteneur,
#    ou le natif est celui qui n en a pas.
local mc5="${OSMO_MULTI_CONF:-/etc/osmocom/osmo-multi.conf}"
if [ -r "$mc5" ]; then
    local _ops5 _self5 _n5=0 _e5 _i5
    _ops5="$(sed -n 's/^MULTI_OPS=//p' "$mc5" | tr -d '"' | tail -1)"
    _self5="$(sed -n 's/^operator_id[[:space:]]*=[[:space:]]*//p' "$src/osmocom/sms-routing.conf" 2>/dev/null
    ↪  | head -1)"
    case "$_self5" in [1-9]*) ;; *) _self5=1 ;; esac
    for _e5 in $_ops5; do
        IFS=: read -r _i5 _ <<< "$_e5"
        case "$_i5" in [1-9]*) [ "$_i5" -gt "$_n5" ] && _n5="$_i5" ;; esac
    done
    # Marqueur d idempotence : rejouer install_configs_native ne doit pas
    # empiler les memes extensions (Asterisk refuserait les doublons avec
    # « already in use » et noierait le journal).
    local _mk5='; --- interop natif : ajoute par _gab_fixups_natifs ---'
    if [ "$_n5" -gt 1 ]; then
        f="$src/asterisk/extensions.conf"
        if [ -f "$f" ] && ! grep -qF "$_mk5" "$f"; then
            { printf '\n%s\n' "$_mk5"
              OP_ID_BASE=1 generate_extensions_interop_out "$_self5" "$_n5"; } >> "$f"
        fi
        f="$src/asterisk/pjsip.conf"
        if [ -f "$f" ] && ! grep -qF "$_mk5" "$f"; then
            { printf '\n%s\n' "$_mk5"
              OP_ID_BASE=1 generate_pjsip_interop_trunks "$_self5" "$_n5"; } >> "$f"
        fi
    fi
```

```
    fi
}

install_configs_native() {
    local src=$1 root="${2:-}"
    _gab_fixups_natifs "$src"
    mkdir -p "${root}/etc/osmocom" "${root}/etc/asterisk" "$HOME/.osmocom/bb"
    # [2026-10-22] /etc/osmocom/.proteges : un nom de fichier par ligne (# commentaire), jamais ecrase par la
    # regeneration. La page http://127.0.0.1:8090/ (onglet Configs osmo) y inscrit chaque fichier qu'elle
    # enregistre ; retirer la ligne rend le fichier au gabarit.
    local _prot="${root}/etc/osmocom/.proteges" _f _b
    for _f in "$src/osmocom"/*; do
        [ -e "$_f" ] || continue
        _b="${_f##*/}"
        if [ -f "$_prot" ] && grep -qxF "$_b" <(sed 's/#.*//; s/[[:space:]]*$//' "$_prot"); then
            printf '  gabarits   %s protege (%s) : conserve\n' "$_b" "$_prot" >&2
            continue
        fi
        cp -f "$_f" "${root}/etc/osmocom/" 2>/dev/null || true
    done
    cp -f "$src/asterisk"/*.conf "${root}/etc/asterisk/" 2>/dev/null || true
    [ -f "$src/bb/mobile.cfg" ]        && cp -f "$src/bb/mobile.cfg"        "$HOME/.osmocom/bb/mobile.cfg"
    [ -f "$src/bb/mobile_group1.cfg" ] && cp -f "$src/bb/mobile_group1.cfg" "$HOME/.osmocom/bb/mobile_group1.cfg"
    if [ -f configs/asound.conf ]; then
        cp -f configs/asound.conf "${root}/etc/asound.conf"
        ALSA_OUTPUT="gsm_out"; ALSA_INPUT="gsm_in"
    fi
}

detect_host_ip() {
    HOST_IP=$(ip route get 1.1.1.1 2>/dev/null \
        | awk '{for(i=1;i<=NF;i++) if($i=="src") print $(i+1)}' | head -1) || true
    [ -z "$HOST_IP" ] && HOST_IP=$(hostname -I 2>/dev/null | awk '{print $1}')
    HOST_IP="${HOST_IP:-127.0.0.1}"
}

# ── Auto-attach tmux ──────────────────────────────────────────────────
# run.sh tourne en arriere-plan et cree la session tmux 'osmocom'
# (socket /tmp/osmocom_tmux). On attend qu'elle soit prete (run.sh termine)
# puis on s'y attache dans le terminal courant. $1 = log run.sh a surveiller.
#   AUTO_ATTACH=0   → desactive (reste en arriere-plan, message manuel)
#   Desactive aussi si pas de TTY (scripte/bg) ou deja dans un tmux.
```

3.229 /opt/GSM/osmo-operator/navigation/init.py

767 octets, 15 lignes → 15 lignes

```
# ss7console - console SS7 de osmo-operator.
#
# Le paquet est volontairement sans dependance : rien que la bibliotheque
# standard de Python 3. Il tourne sur l'HOTE, et parle au lab par deux
# chemins complementaires :
#
#   - les VTY Osmocom, atteints par "docker exec" (les VTY n'ecoutent que sur
#     la boucle locale DU NOEUD) - lecture et pilotage des demons ;
#   - un vrai lien M3UA/SCTP vers l'inter-STP, qui fait de la console un point
#     SS7 a part entiere : elle peut emettre du TCAP/MAP (sendRoutingInfo,
#     sendRoutingInfoForSM, anyTimeInterrogation...) et lire les reponses.
#
# Toute la topologie est HERITEE des configurations du depot et des noeuds :
# aucun point code, aucune adresse, aucun port n'est ecrit en dur ici.
__version__ = "1.0"
```

3.230 /opt/GSM/osmo-operator/navigation/ber.py

4918 octets, 181 lignes → 181 lignes

```
# ber.py - le strict minimum de BER (X.690) : de quoi ecrire et lire du TCAP
# et du MAP sans dependance externe.
#
# On reste volontairement petit : tags courts ou longs, longueurs courtes,
# longues et indefinies, primitifs et constructeurs. Rien de plus n'est
# necessaire pour les operations MAP que la console emet.

def enc_len(n):
    if n < 0x80:
        return bytes([n])
    out = b""
    while n:
        out = bytes([n & 0xFF]) + out
        n >>= 8
    return bytes([0x80 | len(out)]) + out


def tlv(tag, value):
    """tag : entier (0x30) ou sequence d'octets pour les tags longs."""
    if isinstance(tag, int):
        tag = bytes([tag])
    return bytes(tag) + enc_len(len(value)) + value


def enc_int(value, tag=0x02):
    if value == 0:
        return tlv(tag, b"\x00")
    neg = value < 0
    raw = b""
    v = value
    if neg:
        v = -value - 1
    while v:
        raw = bytes([v & 0xFF]) + raw
        v >>= 8
    if neg:
        raw = bytes([(~b) & 0xFF for b in raw])
        if not raw or not (raw[0] & 0x80):
            raw = b"\xff" + raw
    elif raw[0] & 0x80:
        raw = b"\x00" + raw
    return tlv(tag, raw)


def enc_bool(value, tag=0x01):
    return tlv(tag, b"\xff" if value else b"\x00")


def enc_null(tag=0x05):
    return tlv(tag, b"")


def enc_oid(dotted, tag=0x06):
    parts = [int(x) for x in dotted.split(".")]
    body = bytes([parts[0] * 40 + parts[1]])
    for p in parts[2:]:
        if p < 0x80:
            body += bytes([p])
            continue
        chunk = []
        while p:
            chunk.insert(0, (p & 0x7F) | 0x80)
```

```
            p >>= 7
        chunk[-1] &= 0x7F
        body += bytes(chunk)
    return tlv(tag, body)


def ctx(num, value, constructed=True):
    """Tag contexte [num], constructeur ou primitif."""
    tag = 0x80 | num | (0x20 if constructed else 0x00)
    return tlv(tag, value)


class Elem(object):
    __slots__ = ("tag", "constructed", "value", "children", "raw")

    def __init__(self, tag, constructed, value, children, raw):
        self.tag = tag
        self.constructed = constructed
        self.value = value
        self.children = children
        self.raw = raw

    @property
    def cls(self):
        return (self.tag[0] & 0xC0) >> 6      # 0 univ, 1 appli, 2 ctx, 3 priv

    @property
    def num(self):
        t = self.tag[0] & 0x1F
        if t != 0x1F:
            return t
        n = 0
        for b in self.tag[1:]:
            n = (n << 7) | (b & 0x7F)
        return n

    def find(self, cls, num, deep=True):
        """Premier element d'une classe/numero donnes."""
        for e in self.walk() if deep else self.children:
            if e.cls == cls and e.num == num:
                return e
        return None

    def walk(self):
        for c in self.children:
            yield c
            for sub in c.walk():
                yield sub

    def tagstr(self):
        names = {0: "univ", 1: "appl", 2: "ctx", 3: "priv"}
        return "%s[%d]%s" % (names[self.cls], self.num,
                             "c" if self.constructed else "p")

    def __repr__(self):
        return "<%s len=%d>" % (self.tagstr(), len(self.value))


def _read_tag(data, i):
    start = i
    b0 = data[i]
    i += 1
    if (b0 & 0x1F) == 0x1F:
        while i < len(data) and data[i] & 0x80:
            i += 1
        i += 1
```

```
    return data[start:i], i


def _read_len(data, i):
    b = data[i]
    i += 1
    if b < 0x80:
        return b, i
    if b == 0x80:
        return -1, i                    # longueur indefinie
    n = b & 0x7F
    val = int.from_bytes(data[i:i + n], "big")
    return val, i + n


def decode(data, limit=None):
    """Rend la liste des elements de premier niveau."""
    out = []
    i = 0
    end = len(data) if limit is None else min(limit, len(data))
    while i < end:
        try:
            tag, i = _read_tag(data, i)
            length, i = _read_len(data, i)
        except IndexError:
            break
        if length < 0:                  # indefinie : jusqu'au 00 00
            j = data.find(b"\x00\x00", i)
            if j < 0:
                j = end
            value = data[i:j]
            nxt = j + 2
        else:
            value = data[i:i + length]
            nxt = i + length
        constructed = bool(tag[0] & 0x20)
        children = decode(value) if constructed and value else []
        out.append(Elem(tag, constructed, value, children, data[:nxt]))
        i = nxt
    return out


def dump(elems, indent=0, out=None):
    """Arborescence lisible - utilisee pour montrer une reponse MAP brute."""
    out = [] if out is None else out
    for e in elems:
        head = "%s%s len=%d" % ("  " * indent, e.tagstr(), len(e.value))
        if e.constructed and e.children:
            out.append(head)
            dump(e.children, indent + 1, out)
        else:
            out.append("%s  %s" % (head, e.value.hex()))
    return out
```

3.231 /opt/GSM/osmo-operator/navigation/diagram.py

7920 octets, 209 lignes → 209 lignes

```
# diagram.py - le schema du lab, dessine en caracteres, et navigable.
#
# Chaque element du reseau devient une BOITE placee sur une grille de
# caracteres. Les fleches ne suivent pas une liste : elles se deplacent dans
# l'espace, vers la boite la plus proche dans la direction demandee. C'est ce
# qui fait qu'on "circule" dans le schema comme sur une carte.
```

```
#
# Les boites sont construites a partir de la topologie decouverte : rien
# n'est dessine qui n'existe pas dans le lab.

BOX_W = 20

# Ordre d'affichage des services : le coeur d'abord, la radio ensuite, le
# reste apres. Ce qui n'est pas liste garde son ordre de decouverte.
FAMILY_ORDER = {"SS7": 0, "coeur": 1, "abonnes": 2, "radio": 3, "media": 4,
                "data": 5, "voix": 6, "mobile": 7, "autre": 8}


class Box(object):
    __slots__ = ("key", "x", "y", "w", "h", "kind", "title", "lines", "ref",
                 "node", "color")

    def __init__(self, key, x, y, w, h, kind, title, lines, ref=None,
                 node=None, color=0):
        self.key = key
        self.x = x
        self.y = y
        self.w = w
        self.h = h
        self.kind = kind          # hub | stp | service | console | note
        self.title = title
        self.lines = lines
        self.ref = ref            # Service, Node, ou None
        self.node = node          # Node auquel la boite appartient
        self.color = color

    @property
    def cx(self):
        return self.x + self.w // 2

    @property
    def cy(self):
        return self.y + self.h // 2

    def contains(self, x, y):
        return self.x <= x < self.x + self.w and self.y <= y < self.y + self.h


class Canvas(object):
    def __init__(self, width, height):
        self.w = width
        self.h = height
        self.grid = [[" "] * width for _ in range(height)]

    def put(self, x, y, text):
        if y < 0 or y >= self.h:
            return
        for i, ch in enumerate(text):
            if 0 <= x + i < self.w:
                self.grid[y][x + i] = ch

    def hline(self, x1, x2, y, ch="-"):
        for x in range(min(x1, x2), max(x1, x2) + 1):
            if 0 <= x < self.w and 0 <= y < self.h:
                cur = self.grid[y][x]
                self.grid[y][x] = "┼" if cur in "│┤" else ch

    def vline(self, y1, y2, x, ch="│"):
        for y in range(min(y1, y2), max(y1, y2) + 1):
            if 0 <= x < self.w and 0 <= y < self.h:
                cur = self.grid[y][x]
                self.grid[y][x] = "┼" if cur in "─┬┴" else ch
```

```
    def box(self, b):
        top = "┌" + "─" * (b.w - 2) + "┐"
        bot = "└" + "─" * (b.w - 2) + "┘"
        self.put(b.x, b.y, top)
        rows = [b.title] + b.lines
        for i in range(b.h - 2):
            text = rows[i] if i < len(rows) else ""
            self.put(b.x, b.y + 1 + i, "│" + text.ljust(b.w - 2)[:b.w - 2] + "│")
        self.put(b.x, b.y + b.h - 1, bot)

    def render(self):
        return ["".join(row).rstrip() for row in self.grid]


def _service_sort(service):
    return (FAMILY_ORDER.get(service.family, 9), service.port)


def build(topo, extra=None):
    """Rend (lignes, boites). extra : dict d'etat de la console SS7."""
    extra = extra or {}
    ops = topo.operators
    per_op_cols = 2
    svc_rows = max([1] + [(len(o.services) + per_op_cols - 1) // per_op_cols
                          for o in ops])

    col_w = per_op_cols * (BOX_W + 2) + 2
    width = max(80, len(ops) * col_w + 8)
    height = 12 + svc_rows * 4 + 8
    cv = Canvas(width, height)
    boxes = []

    # ── bandeau ──────────────────────────────────────────────────────
    cv.put(2, 0, "RESEAU SS7 - %s" % ("docker" if topo.docker else "natif"))

    # ── hub inter-STP ────────────────────────────────────────────────
    hub = topo.hub
    hub_x = max(2, width // 2 - BOX_W // 2)
    hub_y = 2
    hub_lines = []
    if hub:
        hub_lines.append("PC %s" % (hub.point_code or "?"))
        hub_lines.append("%s:%d" % (topo.hub_ip or "?", topo.hub_port))
        hub_lines.append("M3UA/SCTP" if not topo.hub_local else "local")
    else:
        hub_lines = ["aucun hub", "declare", ""]
    hb = Box("hub", hub_x, hub_y, BOX_W + 6, 6, "hub", " INTER-STP (hub) ",
             hub_lines, ref=hub, node=hub)
    cv.box(hb)
    boxes.append(hb)

    # ── console SS7 (nous) ───────────────────────────────────────────
    con_x = max(2, hub_x - BOX_W - 10)
    con_lines = [
        "PC %s" % extra.get("local_pc", topo.free_point_code()),
        "SSN %s" % extra.get("local_ssn", 8),
        extra.get("state", "hors ligne"),
    ]
    cb = Box("console", con_x, hub_y, BOX_W + 4, 6, "console", " CONSOLE SS7 ",
             con_lines)
    cv.box(cb)
    boxes.append(cb)
    cv.hline(cb.x + cb.w, hb.x - 1, hub_y + 2)
    cv.put(cb.x + cb.w + 1, hub_y + 1, "M3UA")
```

```
    # ── bus vers les operateurs ──────────────────────────────────────
    bus_y = hub_y + 7
    cv.vline(hub_y + 6, bus_y, hb.cx)
    if ops:
        first_x = 4
        xs = []
        for i, _ in enumerate(ops):
            xs.append(first_x + i * col_w + (col_w - BOX_W) // 2)
        cv.hline(min(xs) + BOX_W // 2, max(xs) + BOX_W // 2, bus_y)
        cv.put(hb.cx - 4, bus_y - 1, "")

        for i, node in enumerate(ops):
            x = xs[i]
            stp_y = bus_y + 2
            cv.vline(bus_y, stp_y, x + BOX_W // 2)
            lines = ["PC %s" % (node.point_code or "?"),
                     "M3UA %s" % node.local_m3ua,
                     "noeud WAN %s" % (node.wan_node or "-")]
            sb = Box("stp:%s" % node.name, x, stp_y, BOX_W + 4, 6, "stp",
                     " %s - STP " % node.title, lines, ref=node, node=node)
            cv.box(sb)
            boxes.append(sb)

            svc_y = stp_y + 7
            cv.vline(stp_y + 6, svc_y - 1, x + BOX_W // 2)
            services = sorted(node.services, key=_service_sort)
            for j, svc in enumerate(services):
                row, col = divmod(j, per_op_cols)
                bx = x - 2 + col * (BOX_W + 2)
                by = svc_y + row * 4
                lines = ["VTY %d  %s" % (svc.port, svc.family)]
                box = Box("svc:%s:%d" % (node.name, svc.port), bx, by,
                          BOX_W, 4, "service", " %s" % svc.label, lines,
                          ref=svc, node=node)
                cv.box(box)
                boxes.append(box)
    else:
        cv.put(4, bus_y + 2, "aucun operateur en cours d'execution")

    lines = cv.render()
    while lines and not lines[-1].strip():
        lines.pop()
    return lines, boxes


def navigate(boxes, current, direction):
    """Boite la plus proche dans une direction. dx/dy en {-1,0,1}."""
    if not boxes:
        return current
    cur = boxes[current]
    dx, dy = direction
    best = None
    best_score = None
    for i, b in enumerate(boxes):
        if i == current:
            continue
        vx = b.cx - cur.cx
        vy = b.cy - cur.cy
        if dx and (vx * dx) <= 0:
            continue
        if dy and (vy * dy) <= 0:
            continue
        # Distance dans l'axe demande, penalisee par l'ecart lateral : on
        # prefere ce qui est "droit devant" a ce qui est loin sur le cote.
        along = abs(vx) if dx else abs(vy)
        across = abs(vy) if dx else abs(vx)
```

```
        score = along + across * 2.5
        if best_score is None or score < best_score:
            best_score = score
            best = i
    return best if best is not None else current
```

3.232 /opt/GSM/osmo-operator/navigation/m3ua.py

18922 octets, 463 lignes → 463 lignes

```
# m3ua.py - un ASP M3UA (RFC 4666) en Python, sur SCTP, sans dependance.
#
# La console devient un vrai point SS7 du lab : elle se connecte a l'inter-STP
# (ou au STP d'un operateur), s'annonce (ASPUP), enregistre dynamiquement une
# routing key pour SON point code (RKM - le hub du depot est configure en
# "routing-key-allocation dynamic-permitted", cf. helpers/create_interop.sh),
# passe active (ASPAC), puis emet et recoit du SCCP.
#
# Le lien est du SCTP : socket(AF_INET, SOCK_STREAM, IPPROTO_SCTP). Le PPID
# 3 (M3UA) est pose par sendmsg() - un STP qui filtre sur le PPID jetterait
# sinon nos messages.

import queue
import socket
import struct
import threading
import time

IPPROTO_SCTP = 132
SCTP_SNDRCV = 1          # cmsg type (linux/sctp.h) : 0 = INIT, 1 = SNDRCV
PPID_M3UA = 3

# Decalage des routing contexts de la console : hors de portee du plan du lab
# (n*1000 + o*100 + 50, donc au plus 9950 pour 9 noeuds et 9 operateurs).
CONSOLE_RCTX_BASE = 900000

# Classes et types de messages.
CLS_MGMT, CLS_TRANSFER, CLS_SSNM, CLS_ASPSM, CLS_ASPTM, CLS_RKM = 0, 1, 2, 3, 4, 9
ERR, NTFY = 0, 1
DATA = 1
DUNA, DAVA, DAUD, SCON, DUPU, DRST = 1, 2, 3, 4, 5, 6
ASPUP, ASPDN, BEAT, ASPUP_ACK, ASPDN_ACK, BEAT_ACK = 1, 2, 3, 4, 5, 6
ASPAC, ASPIA, ASPAC_ACK, ASPIA_ACK = 1, 2, 3, 4
REG_REQ, REG_RSP, DEREG_REQ, DEREG_RSP = 1, 2, 3, 4

# Parametres.
P_INFO = 0x0004
P_ROUTING_CTX = 0x0006
P_DIAG = 0x0007
P_HEARTBEAT = 0x0009
P_TRAFFIC_MODE = 0x000B
P_ERROR_CODE = 0x000C
P_STATUS = 0x000D
P_ASP_ID = 0x0011
P_AFFECTED_PC = 0x0012
P_NET_APPEARANCE = 0x0200
P_ROUTING_KEY = 0x0207
P_REG_RESULT = 0x0208
P_LOCAL_RK_ID = 0x020A
P_DPC = 0x020B
P_SERVICE_IND = 0x020C
P_OPC_LIST = 0x020E
P_PROTOCOL_DATA = 0x0210
P_REG_STATUS = 0x0212
```

```
MSG_NAMES = {
    (CLS_MGMT, ERR): "ERR", (CLS_MGMT, NTFY): "NTFY",
    (CLS_TRANSFER, DATA): "DATA",
    (CLS_SSNM, DUNA): "DUNA", (CLS_SSNM, DAVA): "DAVA", (CLS_SSNM, DAUD): "DAUD",
    (CLS_SSNM, SCON): "SCON", (CLS_SSNM, DUPU): "DUPU", (CLS_SSNM, DRST): "DRST",
    (CLS_ASPSM, ASPUP): "ASPUP", (CLS_ASPSM, ASPUP_ACK): "ASPUP_ACK",
    (CLS_ASPSM, ASPDN): "ASPDN", (CLS_ASPSM, ASPDN_ACK): "ASPDN_ACK",
    (CLS_ASPSM, BEAT): "BEAT", (CLS_ASPSM, BEAT_ACK): "BEAT_ACK",
    (CLS_ASPTM, ASPAC): "ASPAC", (CLS_ASPTM, ASPAC_ACK): "ASPAC_ACK",
    (CLS_ASPTM, ASPIA): "ASPIA", (CLS_ASPTM, ASPIA_ACK): "ASPIA_ACK",
    (CLS_RKM, REG_REQ): "REG_REQ", (CLS_RKM, REG_RSP): "REG_RSP",
    (CLS_RKM, DEREG_REQ): "DEREG_REQ", (CLS_RKM, DEREG_RSP): "DEREG_RSP",
}

# RFC 4666 3.6.2 - Registration Status.
REG_STATUS = {
    0: "enregistree", 1: "erreur inconnue", 2: "DPC invalide",
    3: "network appearance invalide", 4: "routing key invalide",
    5: "permission refusee", 6: "routage unique impossible",
    7: "routing key non provisionnee", 8: "ressources insuffisantes",
    9: "parametre de routing key non supporte",
    10: "mode de trafic non supporte", 11: "changement de RK refuse",
    12: "routing key deja enregistree",
}

ERROR_CODES = {
    0x01: "Invalid Version", 0x03: "Unsupported Message Class",
    0x04: "Unsupported Message Type", 0x05: "Unsupported Traffic Mode",
    0x06: "Unexpected Message", 0x07: "Protocol Error",
    0x09: "Invalid Stream Identifier", 0x0D: "Refused - Management Blocking",
    0x0E: "ASP Identifier Required", 0x0F: "Invalid ASP Identifier",
    0x11: "Invalid Parameter Value", 0x12: "Parameter Field Error",
    0x13: "Unexpected Parameter", 0x14: "Destination Status Unknown",
    0x15: "Invalid Network Appearance", 0x16: "Missing Parameter",
    0x19: "Invalid Routing Context", 0x1A: "No Configured AS for ASP",
}


def param(tag, value):
    body = struct.pack("!HH", tag, 4 + len(value)) + value
    pad = (-len(body)) % 4
    return body + b"\x00" * pad


def parse_params(raw):
    out = {}
    i = 0
    while i + 4 <= len(raw):
        tag, length = struct.unpack("!HH", raw[i:i + 4])
        if length < 4:
            break
        value = raw[i + 4:i + length]
        out.setdefault(tag, []).append(value)
        i += length + ((-length) % 4)
    return out


def message(cls, typ, params_bytes=b""):
    return struct.pack("!BBBBI", 1, 0, cls, typ, 8 + len(params_bytes)) + params_bytes


class M3uaMessage(object):
    def __init__(self, cls, typ, params):
        self.cls = cls
        self.typ = typ
```

```
        self.params = params

    @property
    def name(self):
        return MSG_NAMES.get((self.cls, self.typ), "cls%d/typ%d" % (self.cls, self.typ))

    def first(self, tag):
        vals = self.params.get(tag)
        return vals[0] if vals else None

    def __str__(self):
        return "M3UA %s (%d parametre(s))" % (self.name, len(self.params))


class M3uaError(Exception):
    pass


class Asp(object):
    """ASP M3UA client. Un fil de lecture, une file par usage."""

    def __init__(self, remote_ip, remote_port=2908, local_pc=None,
                 routing_ctx=None, traffic_mode=1, log=None):
        # Duree de vie observee d'une association depuis l'hote vers l'inter-STP
        # du lab : une trentaine de secondes. Passe ce delai, le hub cesse de
        # repondre et l'association SCTP expire cote noyau. On la remonte donc
        # au lieu de faire semblant de tenir un lien permanent (voir renew()).
        self.opened_at = 0.0
        self.remote = (remote_ip, int(remote_port))
        self.local_pc = local_pc
        self.routing_ctx = routing_ctx
        self.traffic_mode = traffic_mode
        self.sock = None
        self.rx = queue.Queue()          # DATA (SCCP) recus
        self.ctrl = queue.Queue()        # messages de controle
        self.events = []                 # journal lisible
        self.reader = None
        self.beater = None
        self.beat_interval = 15.0
        self.running = False
        self.state = "deconnecte"
        self._log = log

    # -- journal ------------------------------------------------------
    def log(self, text):
        stamp = time.strftime("%H:%M:%S")
        self.events.append("%s %s" % (stamp, text))
        del self.events[:-400]
        if self._log:
            try:
                self._log(text)
            except Exception:
                pass

    # -- transport ----------------------------------------------------
    def connect(self, timeout=6.0):
        self.sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM, IPPROTO_SCTP)
        self.sock.settimeout(timeout)
        self.sock.connect(self.remote)
        self.sock.settimeout(1.0)
        self.running = True
        self.opened_at = time.time()
        self.state = "connecte"
        self.reader = threading.Thread(target=self._read_loop, daemon=True)
        self.reader.start()
        # Battement de coeur M3UA. L'inter-STP ferme l'association d'un ASP
```

```
    # silencieux au bout d'une trentaine de secondes : sans ces BEAT, un
    # lien parfaitement etabli tombe tout seul entre deux commandes, et la
    # destination passe DUNA sans rien dans les journaux.
    self.beater = threading.Thread(target=self._beat_loop, daemon=True)
    self.beater.start()
    self.log("SCTP etabli vers %s:%d" % self.remote)

def _send(self, data, stream=0):
    if not self.sock:
        raise M3uaError("pas de lien SCTP")
    info = struct.pack("HHHIIIIIi", stream, 0, 0, socket.htonl(PPID_M3UA),
                       0, 0, 0, 0, 0)
    try:
        self.sock.sendmsg([data], [(IPPROTO_SCTP, SCTP_SNDRCV, info)])
    except (OSError, ValueError):
        self.sock.sendall(data)

def send_msg(self, cls, typ, params_bytes=b""):
    raw = message(cls, typ, params_bytes)
    # RFC 4666 1.4.4 : le flux SCTP 0 est RESERVE a la gestion. Le trafic
    # DATA emis dessus se fait refuser avec "Invalid Stream Identifier" -
    # le lien reste up, et les messages disparaissent en silence.
    self._send(raw, stream=(1 if cls == CLS_TRANSFER else 0))
    name = MSG_NAMES.get((cls, typ), "cls%d/typ%d" % (cls, typ))
    if name not in ("DATA", "BEAT"):
        self.log("-> %s" % name)
    return raw

def _beat_loop(self):
    while self.running:
        for _ in range(int(self.beat_interval * 4)):
            if not self.running:
                return
            time.sleep(0.25)
        if not self.running:
            return
        try:
            self.send_msg(CLS_ASPSM, BEAT)
        except Exception:
            return

def _read_loop(self):
    buf = b""
    while self.running:
        try:
            data = self.sock.recv(65536)
        except socket.timeout:
            continue
        except OSError:
            break
        if not data:
            break
        buf += data
        while len(buf) >= 8:
            ver, res, cls, typ = buf[0], buf[1], buf[2], buf[3]
            length = struct.unpack("!I", buf[4:8])[0]
            if length < 8 or len(buf) < length:
                break
            body = buf[8:length]
            buf = buf[length:]
            msg = M3uaMessage(cls, typ, parse_params(body))
            self._dispatch(msg)
    self.running = False
    self.state = "deconnecte"
    self.log("lien SCTP ferme")
```

```
def _dispatch(self, msg):
    if msg.cls == CLS_TRANSFER and msg.typ == DATA:
        pd = msg.first(P_PROTOCOL_DATA)
        if pd and len(pd) >= 12:
            opc, dpc = struct.unpack("!II", pd[0:8])
            si, ni, mp, sls = pd[8], pd[9], pd[10], pd[11]
            self.rx.put({"opc": opc, "dpc": dpc, "si": si, "ni": ni,
                         "sls": sls, "data": pd[12:]})
            self.log("<- DATA opc=%s si=%d (%d octets)" %
                     (_pc(opc), si, len(pd) - 12))
        return
    if msg.cls == CLS_ASPSM and msg.typ == BEAT:
        self.send_msg(CLS_ASPSM, BEAT_ACK, param(P_HEARTBEAT,
                                                 msg.first(P_HEARTBEAT) or b""))
        return
    if msg.cls == CLS_ASPSM and msg.typ == BEAT_ACK:
        # Reponse a NOTRE battement : rien a en faire, et surtout pas a
        # laisser s'accumuler dans la file de controle.
        return
    name = msg.name
    detail = ""
    if msg.cls == CLS_MGMT and msg.typ == ERR:
        code = msg.first(P_ERROR_CODE)
        if code:
            val = struct.unpack("!I", code)[0]
            detail = " : %s" % ERROR_CODES.get(val, "code 0x%02x" % val)
    if msg.cls == CLS_SSNM:
        apc = msg.first(P_AFFECTED_PC)
        if apc and len(apc) >= 4:
            detail = " : PC %s" % _pc(struct.unpack("!I", apc[:4])[0])
    self.log("<- %s%s" % (name, detail))
    self.ctrl.put(msg)

def wait_ctrl(self, wanted, timeout=5.0):
    """Attend un (classe, type) precis. Les autres sont conserves."""
    end = time.time() + timeout
    keep = []
    found = None
    while time.time() < end:
        try:
            msg = self.ctrl.get(timeout=0.2)
        except queue.Empty:
            continue
        if (msg.cls, msg.typ) in wanted:
            found = msg
            break
        keep.append(msg)
    for m in keep:
        self.ctrl.put(m)
    return found

# ── sequence d'activation ──────────────────────────────────────
def asp_up(self, timeout=5.0):
    self.send_msg(CLS_ASPSM, ASPUP)
    msg = self.wait_ctrl({(CLS_ASPSM, ASPUP_ACK), (CLS_MGMT, ERR)}, timeout)
    if msg is None:
        raise M3uaError("pas d'ASPUP_ACK - le STP ne repond pas")
    if msg.cls == CLS_MGMT:
        raise M3uaError("ASPUP refuse par le STP")
    self.state = "ASP_INACTIVE"
    return True

def register(self, point_code, service_ind=None, rk_id=None, timeout=5.0):
    """RKM : demande dynamique d'une routing key pour NOTRE point code.

    Le parametre "Service Indicators" est OMIS par defaut : l'inter-STP
```

```
    d'Osmocom le refuse (registration status 9, "parametre de routing key
    non supporte") et l'enregistrement echoue alors en entier, alors que
    la meme cle sans ce champ passe sans discuter."""
    # L'identifiant local de routing key DOIT etre propre a cette cle :
    # l'inter-STP s'en sert pour reconnaitre une re-registration. Deux ASP
    # qui envoient tous deux l'identifiant 1 se retrouvent dans LE MEME AS
    # (meme routing context), et en traffic-mode override le dernier arrive
    # vole le trafic du premier - la destination du premier devient DUNA.
    # On derive donc l'identifiant du point code demande.
    if rk_id is None:
        rk_id = point_code & 0x00FFFFFF
    # On PROPOSE aussi le routing context, derive du point code. Sans lui,
    # l'inter-STP alloue lui-meme un contexte et rend le MEME a deux ASP
    # differents : le second prend alors la place du premier dans l'AS
    # (traffic-mode override) et le point code du premier passe DUNA. Avec
    # un contexte propre a chaque point code, chacun garde son trafic.
    # Le contexte propose vit dans une plage que le lab n'utilise pas : les
    # noeuds calculent le leur par n*1000 + o*100 + 50 (au plus 9950), et
    # le hub ne distingue pas deux AS qui portent le meme contexte. On
    # ajoute donc un decalage franc au point code plutot que de risquer de
    # tomber sur la routing key d'un operateur.
    rctx = self.routing_ctx if self.routing_ctx is not None else \
        (CONSOLE_RCTX_BASE + (point_code & 0x00FFFFFF))
    dpc = struct.pack("!I", point_code & 0x00FFFFFF)
    rk = param(P_LOCAL_RK_ID, struct.pack("!I", rk_id))
    rk += param(P_ROUTING_CTX, struct.pack("!I", rctx))
    rk += param(P_TRAFFIC_MODE, struct.pack("!I", self.traffic_mode))
    rk += param(P_DPC, dpc)
    if service_ind is not None:
        rk += param(P_SERVICE_IND, bytes([service_ind]))
    self.send_msg(CLS_RKM, REG_REQ, param(P_ROUTING_KEY, rk))
    msg = self.wait_ctrl({(CLS_RKM, REG_RSP), (CLS_MGMT, ERR)}, timeout)
    if msg is None:
        raise M3uaError("pas de REG_RSP - RKM refuse ou non supporte")
    if msg.cls == CLS_MGMT:
        raise M3uaError("REG_REQ rejete (ERR)")
    result = msg.first(P_REG_RESULT)
    status = None
    if result:
        sub = parse_params(result)
        st = sub.get(P_REG_STATUS)
        if st:
            status = struct.unpack("!I", st[0])[0]
        rc = sub.get(P_ROUTING_CTX)
        if rc:
            self.routing_ctx = struct.unpack("!I", rc[0][:4])[0]
    if status not in (0, None):
        raise M3uaError("enregistrement refuse : %s" %
                        REG_STATUS.get(status, status))
    self.log("routing key enregistree : PC %s -> contexte %s (rk id %s)"
             % (_pc(point_code), self.routing_ctx, rk_id))
    return self.routing_ctx

def activate(self, timeout=5.0):
    body = param(P_TRAFFIC_MODE, struct.pack("!I", self.traffic_mode))
    if self.routing_ctx is not None:
        body += param(P_ROUTING_CTX, struct.pack("!I", self.routing_ctx))
    self.send_msg(CLS_ASPTM, ASPAC, body)
    msg = self.wait_ctrl({(CLS_ASPTM, ASPAC_ACK), (CLS_MGMT, ERR)}, timeout)
    if msg is None:
        raise M3uaError("pas d'ASPAC_ACK")
    if msg.cls == CLS_MGMT:
        raise M3uaError("ASPAC refuse")
    self.state = "ASP_ACTIVE"
    return True
```

```
@property
def age(self):
    return time.time() - self.opened_at if self.opened_at else 0.0

def bring_up(self, point_code, register=True):
    self.last_pc = point_code
    self.last_register = register
    self.connect()
    self.asp_up()
    if register:
        try:
            self.register(point_code)
        except M3uaError as exc:
            # Un STP deja provisionne pour ce PC refuse le RKM : on garde
            # la main, l'activation peut tres bien passer sans.
            self.log("RKM : %s (on continue sans)" % exc)
    self.activate()
    return self.state

def renew(self, max_age=3.0):
    """Rouvre le lien s'il est mort, ou s'il approche de sa duree de vie.

    Le seuil est court (3 s) : c'est la duree pendant laquelle l'inter-STP
    du lab repond effectivement a une association venue de l'hote. Une
    reconnexion coute environ 0,3 s (SCTP + ASPUP + RKM + ASPAC).

    Rend True si une reconnexion a eu lieu. Le point code et le contexte
    de routage sont les memes : de l'exterieur, rien ne bouge."""
    if self.state == "ASP_ACTIVE" and self.age < max_age:
        return False
    pc = getattr(self, "last_pc", None)
    if pc is None:
        return False
    try:
        self.close()
    except Exception:
        pass
    self.routing_ctx = None
    self.rx = queue.Queue()
    self.ctrl = queue.Queue()
    self.bring_up(pc, getattr(self, "last_register", True))
    self.log("lien renouvele (PC %s)" % _pc(pc))
    return True

# ── donnees ──────────────────────────────────────────────────────
def send_data(self, opc, dpc, payload, si=3, ni=0, sls=0, mp=0):
    pd = struct.pack("!IIBBBB", opc & 0xFFFFFF, dpc & 0xFFFFFF, si, ni, mp, sls)
    pd += payload
    body = b""
    if self.routing_ctx is not None:
        body += param(P_ROUTING_CTX, struct.pack("!I", self.routing_ctx))
    body += param(P_PROTOCOL_DATA, pd)
    self.send_msg(CLS_TRANSFER, DATA, body)
    self.log("-> DATA vers %s (%d octets SCCP)" % (_pc(dpc), len(payload)))

def recv_data(self, timeout=5.0):
    try:
        return self.rx.get(timeout=timeout)
    except queue.Empty:
        return None

def close(self):
    self.running = False
    if self.sock:
        try:
            self.send_msg(CLS_ASPTM, ASPIA)
```

```
                self.send_msg(CLS_ASPSM, ASPDN)
            except Exception:
                pass
            try:
                self.sock.close()
            except Exception:
                pass
        self.sock = None
        self.state = "deconnecte"


def _pc(val):
    return "%d.%d.%d" % ((val >> 11) & 0x07, (val >> 3) & 0xFF, val & 0x07)
```

3.233 /opt/GSM/osmo-operator/navigation/mapops.py

12184 octets, 302 lignes → 302 lignes

```
# mapops.py - catalogue des operations MAP que la console sait emettre.
#
# Chaque operation declare : son opcode, son contexte applicatif, le SSN de la
# machine a qui elle s'adresse (HLR 6, VLR 7, MSC 8, gsmSCF 147), la liste de
# ses parametres - c'est cette liste que le formulaire de la console affiche -
# et la facon de lire la reponse.
#
# Les encodages suivent 3GPP TS 29.002 (MAP v3).

from . import ber
from . import sccp

# -- types elementaires ------------------------------------------------------

def address_string(number, nai=1, npi=1):
    """ISDN-AddressString : un octet nature/plan, puis les chiffres en TBCD.
    nai 1 = international, npi 1 = E.164."""
    digits = "".join(c for c in str(number) if c.isdigit())
    head = 0x80 | ((nai & 0x07) << 4) | (npi & 0x0F)
    return bytes([head]) + sccp.tbcd(digits)


def read_address_string(raw):
    if not raw:
        return ""
    nai = (raw[0] >> 4) & 0x07
    num = sccp.untbcd(raw[1:])
    prefix = "+" if nai == 1 else ""
    return prefix + num


def imsi(value):
    return sccp.tbcd(value)


def read_imsi(raw):
    return sccp.untbcd(raw)


# -- contextes applicatifs MAP (0.4.0.0.1.0.<contexte>.<version>) ------------
AC = {
    "networkLocUp": "0.4.0.0.1.0.1.3",
    "roamingNumberEnquiry": "0.4.0.0.1.0.3.3",
    "locationInfoRetrieval": "0.4.0.0.1.0.5.3",
    "infoRetrieval": "0.4.0.0.1.0.14.3",
    "shortMsgGateway": "0.4.0.0.1.0.20.3",
```

```
    "shortMsgMT-Relay": "0.4.0.0.1.0.25.3",
    "imsiRetrieval": "0.4.0.0.1.0.26.3",
    "subscriberInfoEnquiry": "0.4.0.0.1.0.28.3",
    "anyTimeInfoEnquiry": "0.4.0.0.1.0.29.3",
}

# ── erreurs MAP ──────────────────────────────────────────────────────
MAP_ERRORS = {
    1: "unknownSubscriber", 3: "unknownMSC", 5: "unidentifiedSubscriber",
    6: "absentSubscriberSM", 7: "unknownEquipment", 8: "roamingNotAllowed",
    9: "illegalSubscriber", 10: "bearerServiceNotProvisioned",
    11: "teleserviceNotProvisioned", 12: "illegalEquipment", 13: "callBarred",
    21: "facilityNotSupported", 27: "absentSubscriber",
    31: "subscriberBusyForMT-SMS", 32: "sm-DeliveryFailure",
    33: "messageWaitingListFull", 34: "systemFailure", 35: "dataMissing",
    36: "unexpectedDataValue", 44: "numberChanged", 45: "busySubscriber",
    46: "noSubscriberReply", 49: "forwardingViolation",
    51: "resourceLimitation", 52: "noRoamingNumberAvailable",
    53: "subscriberBusyForMT-SMS", 71: "unknownAlphabet",
    74: "absentSubscriberSM",
}

SSN_HLR, SSN_VLR, SSN_MSC, SSN_SCF = 6, 7, 8, 147


class Param(object):
    """Un champ du formulaire."""

    def __init__(self, key, label, default="", kind="text", help=""):
        self.key = key
        self.label = label
        self.default = default
        self.kind = kind          # text | number | bool | choice
        self.help = help


class Operation(object):
    def __init__(self, key, name, opcode, ac, ssn, params, encoder,
                 decoder=None, summary=""):
        self.key = key
        self.name = name
        self.opcode = opcode
        self.ac = ac
        self.ssn = ssn
        self.params = params
        self.encoder = encoder
        self.decoder = decoder or (lambda elem: [])
        self.summary = summary

    def encode(self, values):
        return self.encoder(values)

    def defaults(self):
        return dict((p.key, p.default) for p in self.params)


# ── encodeurs ────────────────────────────────────────────────────────

def _enc_sri_sm(v):
    body = ber.ctx(0, address_string(v["msisdn"]), constructed=False)
    body += ber.tlv(0x81, b"\xff" if str(v.get("sm_rp_pri", "1")) in
                    ("1", "true", "oui", "True") else b"\x00")
    body += ber.ctx(2, address_string(v.get("sc_addr") or "0"), constructed=False)
    return ber.tlv(0x30, body)
```

```
def _enc_sri(v):
    body = ber.ctx(0, address_string(v["msisdn"]), constructed=False)
    itype = {"basicCall": 0, "forwarding": 1}.get(v.get("interrogation", "basicCall"), 0)
    body += ber.tlv(0x83, bytes([itype]))
    body += ber.ctx(6, address_string(v.get("gmsc") or "0"), constructed=False)
    return ber.tlv(0x30, body)


def _enc_ati(v):
    ident = v.get("identity_type", "msisdn")
    if ident == "imsi":
        sub = ber.ctx(0, imsi(v["identity"]), constructed=False)
    else:
        sub = ber.ctx(1, address_string(v["identity"]), constructed=False)
    subscriber = ber.ctx(0, sub)
    req = b""
    if str(v.get("location", "1")) in ("1", "true", "oui"):
        req += ber.ctx(0, b"", constructed=False)          # locationInformation
    if str(v.get("state", "1")) in ("1", "true", "oui"):
        req += ber.ctx(1, b"", constructed=False)          # subscriberState
    requested = ber.ctx(1, req)
    scf = ber.ctx(3, address_string(v.get("scf") or "0"), constructed=False)
    return ber.tlv(0x30, subscriber + requested + scf)


def _enc_sai(v):
    body = ber.ctx(0, imsi(v["imsi"]), constructed=False)
    body += ber.ctx(1, bytes([int(v.get("vectors", 1) or 1)]), constructed=False)
    return ber.tlv(0x30, body)


def _enc_send_imsi(v):
    # L'argument est directement une ISDN-AddressString.
    return ber.tlv(0x04, address_string(v["msisdn"]))


def _enc_psi(v):
    body = ber.ctx(0, imsi(v["imsi"]), constructed=False)
    req = b""
    if str(v.get("location", "1")) in ("1", "true", "oui"):
        req += ber.ctx(0, b"", constructed=False)
    if str(v.get("state", "1")) in ("1", "true", "oui"):
        req += ber.ctx(1, b"", constructed=False)
    body += ber.ctx(2, req)
    return ber.tlv(0x30, body)


def _enc_ul(v):
    body = ber.tlv(0x04, imsi(v["imsi"]))
    body += ber.ctx(1, address_string(v.get("msc") or "0"), constructed=False)
    body += ber.tlv(0x04, address_string(v.get("vlr") or "0"))
    return ber.tlv(0x30, body)


def _enc_raw(v):
    txt = (v.get("hex") or "").replace(" ", "")
    return bytes.fromhex(txt) if txt else b""


# ── lecteurs de resultat ──────────────────────────────────────────────

def _walk_pairs(elem, out, path=""):
    """Extraction opportuniste : tout ce qui ressemble a un IMSI ou a un
    numero est rendu lisible, avec le chemin de tags qui l'a produit."""
    for child in (elem.children if elem else []):
        tag = "%s/%s" % (path, child.tagstr())
```

```
        if child.constructed:
            _walk_pairs(child, out, tag)
        else:
            raw = child.value
            if not raw:
                out.append((tag, "(vide)"))
                continue
            txt = raw.hex()
            digits = sccp.untbcd(raw)
            if len(raw) >= 3 and raw[0] & 0x80 and digits[1:].isdigit():
                out.append((tag, "numero %s" % read_address_string(raw)))
            elif digits.isdigit() and len(digits) >= 5:
                out.append((tag, "chiffres %s (hex %s)" % (digits, txt)))
            else:
                out.append((tag, txt))
    return out


def _dec_sri_sm(elem):
    res = []
    if elem is None:
        return res
    kids = elem.children
    if kids and not kids[0].constructed:
        res.append(("IMSI", read_imsi(kids[0].value)))
    for e in (elem.walk() if elem else []):
        if e.cls == 2 and e.num == 1 and not e.constructed and e.value:
            res.append(("Noeud de service (networkNode-Number)",
                        read_address_string(e.value)))
    if not res:
        res = _walk_pairs(elem, [])
    return res


def _dec_sri(elem):
    res = []
    for e in (elem.walk() if elem else []):
        if e.cls == 2 and e.num == 5 and not e.constructed and e.value:
            res.append(("MSRN (roamingNumber)", read_address_string(e.value)))
        if e.cls == 0 and e.num == 4 and len(e.value) > 3:
            res.append(("IMSI", read_imsi(e.value)))
    if not res:
        res = _walk_pairs(elem, [])
    return res


def _dec_generic(elem):
    return _walk_pairs(elem, [])


# ── catalogue ────────────────────────────────────────────────────────

OPERATIONS = [
    Operation(
        "sri-sm", "sendRoutingInfoForSM", 45, AC["shortMsgGateway"], SSN_HLR,
        [Param("msisdn", "MSISDN de l'abonne", "", "text",
               "numero appele, en international sans +"),
         Param("sc_addr", "Adresse du SMSC (GT)", "", "text",
               "serviceCentreAddress - l'adresse qui recevra la reponse"),
         Param("sm_rp_pri", "Priorite SM-RP-PRI", "1", "bool", "")],
        _enc_sri_sm, _dec_sri_sm,
        "Demande au HLR ou joindre l'abonne pour lui remettre un SMS "
        "(IMSI + numero du MSC/VLR de service)."),
    Operation(
        "sri", "sendRoutingInfo", 22, AC["locationInfoRetrieval"], SSN_HLR,
        [Param("msisdn", "MSISDN appele", "", "text", ""),
```

```
         Param("gmsc", "Adresse du GMSC (GT)", "", "text", ""),
         Param("interrogation", "Type d'interrogation", "basicCall", "choice",
               "basicCall | forwarding")],
        _enc_sri, _dec_sri,
        "Interrogation d'acheminement d'appel : le HLR rend le MSRN, "
        "ou une erreur (absentSubscriber, unknownSubscriber...)."),
    Operation(
        "ati", "anyTimeInterrogation", 71, AC["anyTimeInfoEnquiry"], SSN_HLR,
        [Param("identity", "IMSI ou MSISDN", "", "text", ""),
         Param("identity_type", "Type d'identite", "msisdn", "choice",
               "msisdn | imsi"),
         Param("scf", "Adresse du gsmSCF (GT)", "", "text", ""),
         Param("location", "Demander la localisation", "1", "bool", ""),
         Param("state", "Demander l'etat de l'abonne", "1", "bool", "")],
        _enc_ati, _dec_generic,
        "Localisation a la demande : cellule, VLR, etat de l'abonne. "
        "Un HLR reel la refuse souvent (facilityNotSupported)."),
    Operation(
        "sai", "sendAuthenticationInfo", 56, AC["infoRetrieval"], SSN_HLR,
        [Param("imsi", "IMSI", "", "text", ""),
         Param("vectors", "Nombre de vecteurs", "1", "number", "")],
        _enc_sai, _dec_generic,
        "Demande de triplets/quintuplets d'authentification au HLR/AuC."),
    Operation(
        "send-imsi", "sendIMSI", 58, AC["imsiRetrieval"], SSN_HLR,
        [Param("msisdn", "MSISDN", "", "text", "")],
        _enc_send_imsi, _dec_generic,
        "Resolution MSISDN -> IMSI par le HLR."),
    Operation(
        "psi", "provideSubscriberInfo", 70, AC["subscriberInfoEnquiry"], SSN_VLR,
        [Param("imsi", "IMSI", "", "text", ""),
         Param("location", "Localisation", "1", "bool", ""),
         Param("state", "Etat de l'abonne", "1", "bool", "")],
        _enc_psi, _dec_generic,
        "Demande au VLR l'etat et la position d'un abonne."),
    Operation(
        "ul", "updateLocation", 2, AC["networkLocUp"], SSN_HLR,
        [Param("imsi", "IMSI", "", "text", ""),
         Param("msc", "Numero du MSC (GT)", "", "text", ""),
         Param("vlr", "Numero du VLR (GT)", "", "text", "")],
        _enc_ul, _dec_generic,
        "Mise a jour de localisation - la meme que celle d'un VLR reel. "
        "A n'emettre que sur un lab."),
    Operation(
        "raw", "invoke brut", 0, "", SSN_HLR,
        [Param("opcode", "Opcode MAP", "45", "number", ""),
         Param("ac", "Contexte applicatif (OID)", AC["shortMsgGateway"], "text", ""),
         Param("hex", "Parametre BER en hexa", "", "text",
               "colle tel quel apres l'opcode")],
        _enc_raw, _dec_generic,
        "Pour tout ce qui n'est pas au catalogue : opcode et parametre a la main."),
]

BY_KEY = dict((o.key, o) for o in OPERATIONS)


def error_name(code):
    return MAP_ERRORS.get(code, "erreur MAP %s" % code)
```

3.234 /opt/GSM/osmo-operator/navigation/quickcmd.py

2325 octets, 72 lignes → 72 lignes

```
# quickcmd.py - les commandes VTY qui servent tous les jours, par demon.
# Elles alimentent le menu "commandes rapides" de la console : on choisit a la
# fleche, la sortie s'affiche, et on reste dans le schema.

QUICK = {
    "OsmoSTP": [
        ("AS (application servers)", "show cs7 instance 0 as all"),
        ("ASP (points d'acces)", "show cs7 instance 0 asp"),
        ("Table de routage", "show cs7 instance 0 route"),
        ("Utilisateurs SCCP", "show cs7 instance 0 users"),
        ("Connexions SCCP", "show cs7 instance 0 sccp connections"),
        ("SSN locaux", "show cs7 instance 0 sccp ssn"),
    ],
    "OsmoMSC": [
        ("Abonnes en VLR", "show subscriber cache"),
        ("Connexions actives", "show connection"),
        ("Transactions", "show transaction"),
        ("Liens BSC", "show bsc"),
        ("Statistiques", "show statistics"),
    ],
    "OsmoHLR": [
        ("Connexions GSUP (VLR/SMSC)", "show gsup-connections"),
        ("Statistiques", "show stats"),
    ],
    "OsmoBSC": [
        ("Etat des BTS", "show bts"),
        ("Canaux logiques", "show lchan"),
        ("Canaux occupes", "show lchan summary"),
        ("Liens vers le MSC", "show msc connection"),
        ("Timeslots", "show timeslot"),
    ],
    "OsmoBTS": [
        ("Etat du BTS", "show bts 0"),
        ("Canaux logiques", "show lchan summary"),
        ("Liens PHY", "show phy"),
    ],
    "OsmoMGW": [
        ("Points d'extremite", "show mgcp"),
        ("Statistiques", "show mgcp stats"),
    ],
    "OsmoSGSN": [
        ("Abonnes GPRS", "show subscriber cache"),
        ("Contextes PDP", "show pdp-context all"),
        ("Liens BSSGP", "show ns"),
    ],
    "OsmoGGSN": [
        ("Contextes PDP", "show pdp-context all"),
        ("APN", "show apn"),
    ],
    "OsmoPCU": [
        ("Etat TBF", "show tbf all"),
        ("BTS PCU", "show bts"),
    ],
    "OsmoSIPConnector": [
        ("Appels en cours", "show calls"),
        ("Statistiques", "show stats"),
    ],
    "OsmoMobile": [
        ("Etat du mobile", "show ms"),
        ("Reseaux vus", "show subscriber"),
    ],
}

DEFAULT = [
    ("Version", "show version"),
    ("Journal en cours", "show logging vty"),
```

```
    ("Configuration en vigueur", "show running-config"),
]


def for_daemon(daemon):
    return QUICK.get(daemon, []) + DEFAULT
```

3.235 /opt/GSM/osmo-operator/navigation/responder.py

11431 octets, 270 lignes → 270 lignes

```
# responder.py - le repondeur MAP de la console.
#
# Pourquoi : dans ce lab, Osmocom fait dialoguer MSC et HLR en GSUP, pas en
# MAP. Un sendRoutingInfoForSM envoye vers le HLR reel ne trouve donc personne
# a l'ecoute, et le SS7 semble muet alors qu'il achemine parfaitement.
#
# Le repondeur comble ce trou : il s'attache au meme inter-STP, avec son propre
# point code et le SSN 6 (HLR), et repond aux interrogations MAP en puisant les
# donnees dans le VRAI HLR par son VTY. Le trajet, lui, est bien reel : le
# message traverse le hub SS7 comme n'importe quel trafic entre operateurs.
#
#   Terminal A :  ./navigation/ss7-console.py serve --pc 1.11.61
#   Terminal B :  ./navigation/ss7-console.py map sri-sm --pc 1.11.61 \
#                     msisdn=100101 sc_addr=600999

import threading
import time

from . import ber
from . import m3ua
from . import mapops
from . import sccp
from . import tcap
from . import vty as vty_mod

OP_SRI_SM = 45
OP_SRI = 22
OP_SEND_IMSI = 58
OP_SAI = 56
OP_ATI = 71


class Subscriber(object):
    def __init__(self, msisdn, imsi, imei="", nam=""):
        self.msisdn = msisdn
        self.imsi = imsi
        self.imei = imei
        self.nam = nam


class MapResponder(object):
    def __init__(self, topo, local_pc, local_ssn=mapops.SSN_HLR, node=None,
                 hlr_port=4258, msc_number=None, log=None):
        self.topo = topo
        self.local_pc = local_pc
        self.local_ssn = local_ssn
        self.node = node or (topo.operators[0].name if topo.operators else None)
        self.hlr_port = hlr_port
        self.msc_number = msc_number
        self.pool = vty_mod.VtyPool(docker=topo.docker)
        self.asp = None
        self.running = False
        self.thread = None
```

```
    self.events = []
    self._log = log
    self._cache = ([], 0.0)

def log(self, text):
    line = "%s %s" % (time.strftime("%H:%M:%S"), text)
    self.events.append(line)
    del self.events[:-400]
    if self._log:
        self._log(line)

# -- donnees d'abonnes, lues dans le vrai HLR ----------------------------
def subscribers(self, max_age=10.0):
    rows, stamp = self._cache
    if rows and (time.time() - stamp) < max_age:
        return rows
    rows = []
    if self.node:
        try:
            out = self.pool.cmd(self.node, self.hlr_port, "show subscribers all")
        except vty_mod.VtyError as exc:
            self.log("HLR injoignable : %s" % exc)
            out = ""
        for line in out.split("\n"):
            parts = line.split()
            if len(parts) >= 3 and parts[0].isdigit() and parts[2].isdigit():
                rows.append(Subscriber(parts[1], parts[2],
                                       parts[3] if len(parts) > 3 else "",
                                       parts[4] if len(parts) > 4 else ""))
    self._cache = (rows, time.time())
    return rows

def find(self, msisdn=None, imsi=None):
    for sub in self.subscribers():
        if msisdn and sub.msisdn == str(msisdn).lstrip("+"):
            return sub
        if imsi and sub.imsi == str(imsi):
            return sub
    return None

# -- lien ----------------------------------------------------------------
def start(self):
    self.asp = m3ua.Asp(self.topo.hub_ip, self.topo.hub_port, log=self.log)
    self.asp.bring_up(sccp.pc_to_int(self.local_pc))
    self.running = True
    self.thread = threading.Thread(target=self._loop, daemon=True)
    self.thread.start()
    # Le hub ne sert une association venue de l'hote que quelques secondes :
    # sans ce renouvellement, le repondeur serait injoignable au bout d'une
    # dizaine de secondes, sans que rien ne le signale.
    self.keeper = threading.Thread(target=self._keep_alive, daemon=True)
    self.keeper.start()
    self.log("repondeur MAP actif : PC %s, SSN %d, %d abonne(s) connus"
             % (self.local_pc, self.local_ssn, len(self.subscribers())))
    return self.asp.state

def stop(self):
    self.running = False
    if self.asp:
        self.asp.close()
    self.pool.close_all()

def _keep_alive(self):
    while self.running:
        time.sleep(1.0)
        if not self.running:
```

```
            return
        try:
            self.asp.renew(max_age=2.5)
        except Exception as exc:
            self.log("renouvellement impossible : %s" % exc)
            time.sleep(2.0)

def _loop(self):
    while self.running:
        data = self.asp.recv_data(timeout=0.5)
        if data is None:
            continue
        try:
            self._handle(data)
        except Exception as exc:
            self.log("erreur de traitement : %s: %s" % (type(exc).__name__, exc))

# ── traitement ──────────────────────────────────────────────
def _handle(self, data):
    dec = sccp.decode(data["data"])
    if dec is None or dec.kind not in ("UDT", "XUDT"):
        return
    if dec.called.ssn not in (None, self.local_ssn):
        self.log("recu pour SSN %s, ignore (je suis SSN %d)"
                 % (dec.called.ssn, self.local_ssn))
        return
    msg = tcap.parse(dec.data)
    if msg is None or not msg.components:
        return
    comp = msg.components[0]
    if comp.kind != "invoke":
        return
    self.log("<- %s opcode %s de %s" % (msg.kind, comp.opcode, dec.calling))
    answer = self._answer(comp, msg.app_context)
    if answer is None:
        return
    payload = tcap.end_response(msg.otid, answer, msg.app_context or None)
    called = sccp.Address(pc=dec.calling.pc, ssn=dec.calling.ssn,
                          gt=dec.calling.gt,
                          route_on_ssn=dec.calling.gt is None)
    calling = sccp.Address(pc=self.local_pc, ssn=self.local_ssn)
    udt = sccp.udt(called, calling, payload)
    self.asp.send_data(sccp.pc_to_int(self.local_pc),
                       data["opc"], udt)
    self.log("-> reponse envoyee vers PC %s" % m3ua._pc(data["opc"]))

@staticmethod
def _ctx(elem, num):
    """Fils portant le tag contexte [num], en surface puis en profondeur."""
    if elem is None:
        return None
    for child in elem.children:
        if child.cls == 2 and child.num == num:
            return child
    for child in elem.walk():
        if child.cls == 2 and child.num == num:
            return child
    return None

def _param_values(self, comp):
    """Numeros et IMSI trouves dans le parametre de l'invoke, DANS L'ORDRE
    de la sequence : le premier champ d'un SRI-SM est le MSISDN, le
    troisieme l'adresse du SMSC - les confondre repondrait sur le mauvais
    abonne."""
    found = {"numbers": [], "digits": []}
    if comp.parameter is None:
```

```
            return found
        stack = [comp.parameter]
        while stack:
            elem = stack.pop(0)
            if elem.constructed:
                stack = list(elem.children) + stack
                continue
            raw = elem.value
            if not raw:
                continue
            if raw[0] & 0x80 and len(raw) >= 2:
                found["numbers"].append(mapops.read_address_string(raw))
            digits = sccp.untbcd(raw)
            if digits.isdigit():
                found["digits"].append(digits)
        return found

    def _answer(self, comp, app_context):
        op = comp.opcode
        vals = self._param_values(comp)
        wanted = (vals["numbers"] + vals["digits"] + [""])[0].lstrip("+")
        # Champ nomme quand la structure le permet : SRI-SM et SRI portent le
        # MSISDN en [0], l'ATI son identite d'abonne en [0]/[0-1].
        if op in (OP_SRI_SM, OP_SRI):
            champ = self._ctx(comp.parameter, 0)
            if champ is not None and champ.value:
                wanted = mapops.read_address_string(champ.value).lstrip("+")
        elif op == OP_ATI:
            ident = self._ctx(comp.parameter, 0)
            if ident is not None:
                imsi_champ = self._ctx(ident, 0)
                msisdn_champ = self._ctx(ident, 1)
                if msisdn_champ is not None and msisdn_champ.value:
                    wanted = mapops.read_address_string(msisdn_champ.value).lstrip("+")
                elif imsi_champ is not None and imsi_champ.value:
                    wanted = mapops.read_imsi(imsi_champ.value)

        if op == OP_SRI_SM:
            sub = self.find(msisdn=wanted)
            if sub is None:
                self.log("SRI-SM %s : abonne inconnu" % wanted)
                return tcap.return_error(comp.invoke_id, 1)      # unknownSubscriber
            node_number = self.msc_number or sub.msisdn
            info = ber.ctx(1, mapops.address_string(node_number), constructed=False)
            body = ber.tlv(0x04, mapops.imsi(sub.imsi)) + ber.ctx(0, info)
            self.log("SRI-SM %s -> IMSI %s" % (wanted, sub.imsi))
            return tcap.return_result_last(comp.invoke_id, op, ber.tlv(0x30, body))

        if op == OP_SRI:
            sub = self.find(msisdn=wanted)
            if sub is None:
                return tcap.return_error(comp.invoke_id, 1)
            msrn = self.msc_number or ("99" + sub.msisdn)
            body = ber.ctx(9, mapops.imsi(sub.imsi), constructed=False)
            body += ber.tlv(0x04, mapops.address_string(msrn))
            self.log("SRI %s -> MSRN %s" % (wanted, msrn))
            return tcap.return_result_last(comp.invoke_id, op, ber.tlv(0x30, body))

        if op == OP_SEND_IMSI:
            sub = self.find(msisdn=wanted)
            if sub is None:
                return tcap.return_error(comp.invoke_id, 1)
            self.log("sendIMSI %s -> %s" % (wanted, sub.imsi))
            return tcap.return_result_last(comp.invoke_id, op,
                                           ber.tlv(0x04, mapops.imsi(sub.imsi)))
```

```
        if op == OP_ATI:
            sub = self.find(msisdn=wanted) or self.find(imsi=wanted)
            if sub is None:
                return tcap.return_error(comp.invoke_id, 1)
            # locationInformation [0] { vlr-number [1] } - le minimum utile.
            loc = ber.ctx(1, mapops.address_string(self.msc_number or sub.msisdn),
                          constructed=False)
            body = ber.ctx(0, ber.ctx(0, loc))
            self.log("ATI %s -> localisation" % wanted)
            return tcap.return_result_last(comp.invoke_id, op, ber.tlv(0x30, body))

        if op == OP_SAI:
            # Pas d'AuC ici : on le dit franchement plutot que d'inventer des
            # vecteurs qui ne s'authentifieraient nulle part.
            self.log("sendAuthenticationInfo : refuse (pas d'AuC dans ce repondeur)")
            return tcap.return_error(comp.invoke_id, 34)          # systemFailure

        self.log("opcode %s non gere -> facilityNotSupported" % op)
        return tcap.return_error(comp.invoke_id, 21)
```

3.236 /opt/GSM/osmo-operator/navigation/sccp.py

7550 octets, 222 lignes → 222 lignes

```
# sccp.py - SCCP sans connexion (UDT / UDTS / XUDT), juste ce qu'il faut pour
# transporter du TCAP et pour COMPRENDRE un refus.
#
# Le "return cause" d'un UDTS est souvent la seule reponse qu'on obtient d'un
# reseau de labo : "no translation for this address", "unequipped user"...
# Le decoder proprement evite de conclure a tort a un silence.

MSG_UDT = 0x09
MSG_UDTS = 0x0A
MSG_XUDT = 0x11
MSG_XUDTS = 0x12

RETURN_CAUSE = {
    0x00: "no translation for an address of such nature",
    0x01: "no translation for this specific address",
    0x02: "subsystem congestion",
    0x03: "subsystem failure",
    0x04: "unequipped user (SSN absent)",
    0x05: "MTP failure",
    0x06: "network congestion",
    0x07: "unqualified",
    0x08: "error in message transport",
    0x09: "error in local processing",
    0x0A: "destination cannot perform reassembly",
    0x0B: "SCCP failure",
    0x0C: "hop counter violation",
    0x0D: "segmentation not supported",
    0x0E: "segmentation failure",
}

# SSN usuels (Q.713 + usages Osmocom).
SSN = {
    0: "inconnu", 1: "SCCP mgmt", 6: "HLR", 7: "VLR", 8: "MSC", 9: "EIR",
    10: "AUC", 142: "RANAP", 143: "RNSAP", 145: "GMLC", 146: "CAP",
    147: "gsmSCF", 148: "SIWF", 149: "SGSN", 150: "GGSN", 254: "BSSAP",
}

NAI = {0: "inconnu", 1: "abonne", 2: "reserve national", 3: "national",
       4: "international"}
```

```
def tbcd(digits):
    """Chiffres -> TBCD (demi-octets inverses, complement F)."""
    digits = "".join(c for c in str(digits) if c in "0123456789*#abc")
    out = bytearray()
    for i in range(0, len(digits), 2):
        lo = int(digits[i], 16)
        hi = int(digits[i + 1], 16) if i + 1 < len(digits) else 0x0F
        out.append((hi << 4) | lo)
    return bytes(out)


def untbcd(raw):
    out = []
    for b in raw:
        for nib in (b & 0x0F, (b >> 4) & 0x0F):
            if nib == 0x0F:
                continue
            out.append("%x" % nib)
    return "".join(out)


def pc_to_int(pc):
    """'1.11.2' (format ITU 3-8-3) ou entier -> entier 14 bits."""
    if isinstance(pc, int):
        return pc
    pc = str(pc).strip()
    if pc.isdigit():
        return int(pc)
    a, b, c = (int(x) for x in pc.split("."))
    return (a << 11) | (b << 3) | c


def int_to_pc(val):
    return "%d.%d.%d" % ((val >> 11) & 0x07, (val >> 3) & 0xFF, val & 0x07)


class Address(object):
    """Adresse SCCP : point code, SSN, et/ou global title (GTI 4)."""

    def __init__(self, pc=None, ssn=None, gt=None, tt=0, np=1, es=2, nai=4,
                 route_on_ssn=None):
        self.pc = pc
        self.ssn = ssn
        self.gt = gt
        self.tt = tt
        self.np = np          # 1 = ISDN/telephony E.164
        self.es = es          # 2 = BCD, nombre pair (ajuste a l'encodage)
        self.nai = nai        # 4 = international
        if route_on_ssn is None:
            route_on_ssn = gt is None
        self.route_on_ssn = route_on_ssn

    def encode(self):
        ai = 0
        body = b""
        if self.pc is not None:
            ai |= 0x01
            val = pc_to_int(self.pc)
            body += bytes([val & 0xFF, (val >> 8) & 0x3F])
        if self.ssn is not None:
            ai |= 0x02
            body += bytes([self.ssn & 0xFF])
        if self.gt:
            ai |= (4 << 2)                      # GTI = 4
            digits = "".join(c for c in str(self.gt) if c.isdigit())
```

```
            es = 1 if len(digits) % 2 else 2
            body += bytes([self.tt & 0xFF,
                           ((self.np & 0x0F) << 4) | (es & 0x0F),
                           self.nai & 0x7F]) + tbcd(digits)
        if self.route_on_ssn:
            ai |= 0x40
        return bytes([ai]) + body

    @classmethod
    def decode(cls, raw):
        if not raw:
            return cls()
        ai = raw[0]
        i = 1
        pc = ssn = gt = None
        tt = np = es = nai = 0
        if ai & 0x01:
            pc = int_to_pc(raw[i] | ((raw[i + 1] & 0x3F) << 8))
            i += 2
        if ai & 0x02:
            ssn = raw[i]
            i += 1
        gti = (ai >> 2) & 0x0F
        if gti == 4 and len(raw) > i + 2:
            tt = raw[i]
            np = (raw[i + 1] >> 4) & 0x0F
            es = raw[i + 1] & 0x0F
            nai = raw[i + 2] & 0x7F
            gt = untbcd(raw[i + 3:])
        elif gti == 1 and len(raw) > i:
            nai = raw[i] & 0x7F
            gt = untbcd(raw[i + 1:])
        elif gti == 2 and len(raw) > i:
            tt = raw[i]
            gt = untbcd(raw[i + 1:])
        addr = cls(pc=pc, ssn=ssn, gt=gt, tt=tt, np=np, es=es, nai=nai,
                   route_on_ssn=bool(ai & 0x40))
        return addr

    def __str__(self):
        bits = []
        if self.pc is not None:
            bits.append("PC %s" % self.pc)
        if self.ssn is not None:
            bits.append("SSN %d (%s)" % (self.ssn, SSN.get(self.ssn, "?")))
        if self.gt:
            bits.append("GT %s (nai %s)" % (self.gt, NAI.get(self.nai, self.nai)))
        bits.append("route sur %s" % ("SSN/PC" if self.route_on_ssn else "GT"))
        return ", ".join(bits)


def udt(called, calling, data, pclass=0):
    """UDT : 3 pointeurs relatifs, puis les trois champs de longueur variable."""
    c1 = called.encode()
    c2 = calling.encode()
    f1 = bytes([len(c1)]) + c1
    f2 = bytes([len(c2)]) + c2
    f3 = bytes([len(data)]) + data
    p1 = 3
    p2 = p1 + len(f1) - 1
    p3 = p2 + len(f2) - 1
    return bytes([MSG_UDT, pclass & 0xFF, p1, p2, p3]) + f1 + f2 + f3


class Decoded(object):
    def __init__(self, kind, called=None, calling=None, data=b"", cause=None):
```

```
        self.kind = kind
        self.called = called
        self.calling = calling
        self.data = data
        self.cause = cause

    @property
    def cause_text(self):
        if self.cause is None:
            return ""
        return RETURN_CAUSE.get(self.cause, "cause 0x%02x" % self.cause)

    def __str__(self):
        if self.kind == "UDTS":
            return "UDTS - refus SCCP : %s" % self.cause_text
        return "%s vers %s, de %s, %d octets utiles" % (
            self.kind, self.called, self.calling, len(self.data))


def decode(raw):
    if not raw:
        return None
    mtype = raw[0]
    if mtype == MSG_UDT:
        p1, p2, p3 = raw[2], raw[3], raw[4]
        o1, o2, o3 = 2 + p1, 3 + p2, 4 + p3
        called = Address.decode(raw[o1 + 1:o1 + 1 + raw[o1]])
        calling = Address.decode(raw[o2 + 1:o2 + 1 + raw[o2]])
        data = raw[o3 + 1:o3 + 1 + raw[o3]]
        return Decoded("UDT", called, calling, data)
    if mtype == MSG_UDTS:
        cause = raw[1]
        p1, p2, p3 = raw[2], raw[3], raw[4]
        o1, o2, o3 = 2 + p1, 3 + p2, 4 + p3
        called = Address.decode(raw[o1 + 1:o1 + 1 + raw[o1]])
        calling = Address.decode(raw[o2 + 1:o2 + 1 + raw[o2]])
        data = raw[o3 + 1:o3 + 1 + raw[o3]]
        return Decoded("UDTS", called, calling, data, cause)
    if mtype in (MSG_XUDT, MSG_XUDTS):
        # XUDT : classe, hop counter, puis 4 pointeurs. On ne lit que l'utile.
        off = 3
        p1, p2, p3 = raw[off], raw[off + 1], raw[off + 2]
        o1, o2, o3 = off + p1, off + 1 + p2, off + 2 + p3
        called = Address.decode(raw[o1 + 1:o1 + 1 + raw[o1]])
        calling = Address.decode(raw[o2 + 1:o2 + 1 + raw[o2]])
        data = raw[o3 + 1:o3 + 1 + raw[o3]]
        return Decoded("XUDT" if mtype == MSG_XUDT else "XUDTS",
                       called, calling, data,
                       raw[1] if mtype == MSG_XUDTS else None)
    return Decoded("SCCP 0x%02x" % mtype, data=raw[1:])
```

3.237 /opt/GSM/osmo-operator/navigation/ss7-console.py

9623 octets, 254 lignes → 254 lignes

```
#!/usr/bin/env python3
# ss7-console.py - point d'entree de la console SS7 de osmo-operator.
#
#   ./navigation/ss7-console.py                le schema navigable (fleches)
#   ./navigation/ss7-console.py --list         l'inventaire, en texte
#   ./navigation/ss7-console.py audit 1.11.1   audit SS7 d'un point code
#   ./navigation/ss7-console.py vty 1 4258 "show gsup-connections"
#   ./navigation/ss7-console.py map sri-sm --pc 1.11.1 msisdn=112233445566
#
```

```
# Le mode texte sert aux scripts et aux machines sans terminal ; le mode
# schema est celui du quotidien.

import argparse
import os
import sys
import time

HERE = os.path.dirname(os.path.abspath(__file__))
REPO = os.path.dirname(HERE)
if REPO not in sys.path:
    sys.path.insert(0, REPO)

from navigation import mapops, responder as responder_mod, ss7   # noqa: E402
from navigation.ss7 import Ss7Error                               # noqa: E402
from navigation import topo as topo_mod, vty as vty_mod           # noqa: E402


def cmd_list(topo, args):
    print("Mode          : %s" % ("docker" if topo.docker else "natif"))
    print("Hub inter-STP : %s:%s (%s)" % (
        topo.hub_ip or "-", topo.hub_port,
        "local" if topo.hub_local else "distant"))
    print("PC libre      : %s" % topo.free_point_code())
    for node in topo.operators:
        print("")
        print("%s (%s)" % (node.title, node.name))
        print("  point code  : %s   M3UA local %s   noeud WAN %s"
              % (node.point_code, node.local_m3ua, node.wan_node or "-"))
        print("  hub vu d'ici: %s:%s  routing key %s"
              % (node.hub_ip, node.hub_port, node.routing_ctx))
        for name, addr in sorted(node.sccp.items()):
            if name.startswith("_self_"):
                continue
            print("  sccp %-14s PC %-8s SSN %s" % (name, addr.get("pc") or "-",
                                                   addr.get("ssn") or "-"))
        if node.services:
            print("  VTY : " + ", ".join("%s:%d" % (s.label, s.port)
                                         for s in node.services))
    return 0


def _client(topo, args):
    local = getattr(args, "from_pc", None) or getattr(args, "global_pc", None) \
        or topo.free_point_code()
    client = ss7.Ss7Client(topo.hub_ip, topo.hub_port, local,
                           log=(lambda t: print("  [m3ua] %s" % t)) if args.verbose
                           else None)
    print("Lien M3UA vers %s:%s ..." % (topo.hub_ip or "?", topo.hub_port))
    try:
        print("  etat : %s" % client.connect())
    except (Ss7Error, OSError) as exc:
        raise SystemExit("  impossible : %s" % exc)
    return client


def cmd_audit(topo, args):
    client = _client(topo, args)
    try:
        for pc in args.point_code:
            print("")
            print(client.audit(pc))
    finally:
        client.close()
    return 0
```

```
def cmd_map(topo, args):
    if args.operation not in mapops.BY_KEY:
        print("operation inconnue. Disponibles : %s"
              % ", ".join(sorted(mapops.BY_KEY)))
        return 2
    op = mapops.BY_KEY[args.operation]
    values = op.defaults()
    for item in args.params:
        if "=" not in item:
            print("parametre attendu sous la forme cle=valeur : %s" % item)
            return 2
        key, val = item.split("=", 1)
        values[key] = val
    if not args.pc and not args.gt:
        print("il faut --pc (point code) ou --gt (global title)")
        return 2
    client = _client(topo, args)
    try:
        rep = client.send_map(op, values, args.pc or "0.0.0",
                              args.ssn, args.gt or "",
                              route_on_gt=bool(args.gt and not args.pc),
                              timeout=args.timeout)
        print("")
        print(rep)
    finally:
        client.close()
    return 0 if rep.ok else 1


def cmd_vty(topo, args):
    node = args.node
    if node.isdigit():
        node = "osmo-operator-%s" % node
    pool = vty_mod.VtyPool(docker=topo.docker)
    try:
        print(pool.cmd(node, args.port, " ".join(args.command)))
    except vty_mod.VtyError as exc:
        print("VTY %s:%s injoignable : %s" % (node, args.port, exc),
              file=sys.stderr)
        return 1
    finally:
        pool.close_all()
    return 0


def cmd_serve(topo, args):
    node = args.node
    if node and node.isdigit():
        node = "osmo-operator-%s" % node
    rep = responder_mod.MapResponder(
        topo, args.pc or getattr(args, "global_pc", None) or _second_pc(topo),
        args.ssn, node=node,
        msc_number=args.msc_number, log=lambda t: print("  %s" % t, flush=True))
    print("Repondeur MAP : PC %s, SSN %s, donnees du HLR de %s"
          % (rep.local_pc, rep.local_ssn,
             node or (topo.operators[0].name if topo.operators else "-")))
    try:
        print("Etat du lien : %s" % rep.start())
    except Exception as exc:
        print("  impossible de monter le repondeur : %s" % exc, file=sys.stderr)
        return 2
    print("Ctrl-C pour arreter.")
    try:
        while True:
            time.sleep(0.5)
```

```
            if rep.asp is not None and rep.asp.state == "deconnecte" \
                    and not rep.running:
                break
    except KeyboardInterrupt:
        print("")
    finally:
        rep.stop()
    return 0


def _second_pc(topo):
    """Un point code distinct de celui que prend la console emettrice : le
    deuxieme de la liste des libres, jamais un increment (le troisieme membre
    d'un point code ITU tient sur 3 bits, .8 n'existe pas)."""
    libres = topo.free_point_codes(2)
    return libres[1] if len(libres) > 1 else libres[0]


def cmd_ops(topo, args):
    for op in mapops.OPERATIONS:
        print("%-10s %-24s opcode %-3s SSN %-3s" % (op.key, op.name, op.opcode, op.ssn))
        print("           %s" % op.summary)
        print("           parametres : %s"
              % ", ".join(p.key for p in op.params))
    return 0


def main(argv=None):
    ap = argparse.ArgumentParser(
        description="Console SS7 de osmo-operator : schema navigable, VTY et MAP.")
    ap.add_argument("--pc", dest="global_pc",
                    help="point code de la console (avant la sous-commande)")
    ap.add_argument("--no-probe", action="store_true",
                    help="ne pas sonder les VTY a la decouverte (plus rapide)")
    ap.add_argument("-v", "--verbose", action="store_true",
                    help="journal du lien M3UA")
    sub = ap.add_subparsers(dest="cmd")

    sub.add_parser("list", help="inventaire du lab, en texte")
    sub.add_parser("ops", help="catalogue des operations MAP")

    p = sub.add_parser("audit", help="audit SS7 (DAUD) d'un ou plusieurs PC")
    p.add_argument("point_code", nargs="+")
    p.add_argument("--from-pc", dest="from_pc",
                   help="point code de la console (defaut : calcule)")

    p = sub.add_parser("map", help="emettre une operation MAP")
    p.add_argument("operation")
    p.add_argument("params", nargs="*", help="cle=valeur (n'importe ou)")
    p.add_argument("--from-pc", dest="from_pc",
                   help="point code de la console (defaut : calcule)")
    p.add_argument("--pc", dest="pc", help="point code destination")
    p.add_argument("--ssn", type=int, help="SSN destination")
    p.add_argument("--gt", help="global title destination")
    p.add_argument("--timeout", type=float, default=6.0)

    p = sub.add_parser("serve", help="repondeur MAP (HLR simule) sur le SS7 du lab")
    p.add_argument("--pc", dest="pc", help="point code du repondeur")
    p.add_argument("--ssn", type=int, default=mapops.SSN_HLR,
                   help="SSN ecoute (6 = HLR)")
    p.add_argument("--node", help="operateur dont le HLR fournit les donnees")
    p.add_argument("--msc-number", help="numero rendu comme noeud de service")

    p = sub.add_parser("vty", help="une commande VTY, sans interface")
    p.add_argument("node", help="numero d'operateur ou nom de conteneur")
    p.add_argument("port", type=int)
```

```
    p.add_argument("command", nargs="+")

    args, extra = ap.parse_known_args(argv)
    # "map sri-sm --pc 1.11.61 msisdn=100101" : argparse laisse les positionnels
    # qui suivent une option de cote. On les recupere plutot que de les perdre.
    leftovers = [x for x in extra if "=" in x and not x.startswith("-")]
    unknown = [x for x in extra if x not in leftovers]
    if unknown:
        ap.error("arguments inconnus : %s" % " ".join(unknown))
    if leftovers:
        if getattr(args, "cmd", None) == "map":
            args.params = list(getattr(args, "params", [])) + leftovers
        else:
            ap.error("arguments inconnus : %s" % " ".join(leftovers))
    if not hasattr(args, "pc"):
        args.pc = None

    probe = not args.no_probe and args.cmd in (None, "list")
    topo = topo_mod.discover(probe=probe)
    if topo.errors:
        for err in topo.errors:
            print("attention : %s" % err, file=sys.stderr)

    if args.cmd == "list":
        return cmd_list(topo, args)
    if args.cmd == "ops":
        return cmd_ops(topo, args)
    if args.cmd == "audit":
        return cmd_audit(topo, args)
    if args.cmd == "map":
        return cmd_map(topo, args)
    if args.cmd == "serve":
        return cmd_serve(topo, args)
    if args.cmd == "vty":
        return cmd_vty(topo, args)

    if not sys.stdout.isatty():
        print("pas de terminal : utilisez 'list', 'audit', 'map' ou 'vty'.",
              file=sys.stderr)
        return 2
    from navigation import tui
    tui.main(topo)
    return 0


if __name__ == "__main__":
    sys.exit(main())
```

3.238 /opt/GSM/osmo-operator/navigation/ss7-diag.py

16400 octets, 415 lignes → 415 lignes

```
#!/usr/bin/env python3
# ss7-diag.py - diagnostic SS7 complet du lab, depuis l'hote.
#
# Le script se situe TOUT SEUL : il deduit le depot de son propre chemin, et
# la topologie des conteneurs et de leurs configurations. Aucun argument n'est
# necessaire.
#
#   ./navigation/ss7-diag.py               diagnostic complet
#   ./navigation/ss7-diag.py --rapide      sans les tests M3UA/MAP
#   ./navigation/ss7-diag.py --log FICHIER copie integrale dans un fichier
#
# Ce qui est verifie, dans l'ordre ou une panne se lit :
```

```
#   1. l'environnement (docker, SCTP, modules) ;
#   2. la topologie heritee des configurations ;
#   3. par operateur : STP, MSC, BSC, HLR - VTY, AS, ASP, routes, SSN ;
#   4. le lien vers l'inter-STP : association SCTP, etat de l'AS ;
#   5. la console comme ASP M3UA : ASPUP, RKM, ASPAC, puis audit de chaque
#      point code connu ;
#   6. un aller-retour MAP reel (sendRoutingInfoForSM) a travers le hub.
#
# Code de sortie = nombre d'echecs.

import argparse
import os
import re
import subprocess
import sys
import time

HERE = os.path.dirname(os.path.abspath(__file__))
REPO = os.path.dirname(HERE)
if REPO not in sys.path:
    sys.path.insert(0, REPO)

from navigation import mapops, responder as responder_mod, sccp, ss7   # noqa: E402
from navigation import topo as topo_mod, vty as vty_mod                # noqa: E402

G, Y, R, C, B, N = ("\033[0;32m", "\033[1;33m", "\033[0;31m", "\033[0;36m",
                    "\033[1m", "\033[0m")

STATE = {"pass": 0, "warn": 0, "fail": 0, "skip": 0, "log": None}


def out(text=""):
    print(text, flush=True)
    if STATE["log"]:
        STATE["log"].write(re.sub(r"\033\[[0-9;]*m", "", text) + "\n")
        STATE["log"].flush()


def section(title):
    out("")
    out("%s%s%s" % (B, title, N))
    out("-" * 70)


def ok(text):
    STATE["pass"] += 1
    out("  %s✓%s %s" % (G, N, text))


def warn(text):
    STATE["warn"] += 1
    out("  %s!%s %s" % (Y, N, text))


def fail(text):
    STATE["fail"] += 1
    out("  %s✗%s %s" % (R, N, text))


def skip(text):
    STATE["skip"] += 1
    out("  %s-%s %s" % (Y, N, text))


def info(text):
    out("    %s" % text)
```

```
def sh(cmd):
    try:
        p = subprocess.run(cmd, shell=True, stdout=subprocess.PIPE,
                           stderr=subprocess.DEVNULL, timeout=30)
        return p.stdout.decode("utf-8", "replace")
    except Exception:
        return ""


# ── 1. environnement ──────────────────────────────────────────────
def check_env():
    section("1. ENVIRONNEMENT")
    if sh("docker ps --format '{{.Names}}'").strip():
        ok("docker repond")
    else:
        warn("docker ne rend aucun conteneur - mode natif, ou docker arrete")
    import socket
    try:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM, 132)
        s.close()
        ok("SCTP disponible sur l'hote (IPPROTO_SCTP)")
    except Exception as exc:
        fail("SCTP indisponible : %s - le lien M3UA sera impossible" % exc)
        info("modprobe sctp")
    out("    depot   : %s" % REPO)
    out("    python  : %s" % sys.version.split()[0])


# ── 2. topologie ──────────────────────────────────────────────────
def check_topo(topo):
    section("2. TOPOLOGIE (heritee des configurations)")
    if not topo.operators:
        fail("aucun operateur trouve")
        return
    ok("%d operateur(s) : %s" % (len(topo.operators),
                                 ", ".join(n.name for n in topo.operators)))
    if topo.hub_ip:
        ok("inter-STP : %s:%s (%s)" % (topo.hub_ip, topo.hub_port,
                                       "local" if topo.hub_local else "distant"))
    else:
        fail("aucun inter-STP connu - ni conteneur local, ni OSMO_HUB_IP")
    vus = {}
    for node in topo.operators:
        if not node.point_code:
            fail("%s : aucun point code lu dans osmo-stp.cfg" % node.name)
            continue
        if node.point_code in vus:
            fail("%s et %s partagent le point code %s - collision SS7"
                 % (node.name, vus[node.point_code], node.point_code))
        else:
            vus[node.point_code] = node.name
            ok("%s : PC %s, routing key %s, noeud WAN %s"
               % (node.title, node.point_code, node.routing_ctx,
                  node.wan_node or "-"))
        for nom, addr in sorted(node.sccp.items()):
            if nom.startswith("_self_") or not addr.get("pc"):
                continue
            info("%-10s %-10s PC %-8s SSN %s" % (node.title, nom, addr["pc"],
                                                 addr.get("ssn")))
        # Format ITU 3-8-3 : le troisieme membre tient sur 3 bits.
        membres = node.point_code.split(".")
        if len(membres) == 3:
            a, b, c = (int(x) for x in membres)
            if a > 7 or b > 255 or c > 7:
```

```
                fail("%s : point code %s hors format ITU 3-8-3 (max 7.255.7)"
                     % (node.title, node.point_code))
    if topo.docker:
        libres = topo.free_point_codes(2)
        info("point codes libres pour la console : %s" % ", ".join(libres))


# ── 3. les demons, par operateur ──────────────────────────────────────
def check_nodes(topo, pool):
    for node in topo.operators:
        section("3. %s (%s)" % (node.title, node.name))
        stp = node.service("OsmoSTP") or type("S", (), {"port": 4239})()
        try:
            as_out = pool.cmd(node.name, stp.port, "show cs7 instance 0 as all")
        except vty_mod.VtyError as exc:
            fail("VTY STP (%s) inaccessible : %s" % (stp.port, exc))
            continue
        actifs = len(re.findall(r"AS_ACTIVE", as_out))
        inter = re.search(r"as-inter\s+(\S+)", as_out)
        if inter and inter.group(1) == "AS_ACTIVE":
            ok("as-inter -> hub : ACTIVE")
        elif inter:
            fail("as-inter : %s" % inter.group(1))
        else:
            warn("aucun as-inter declare (pas de lien vers un hub)")
        if actifs >= 2:
            ok("AS actifs sur le STP local : %d" % actifs)
        else:
            warn("AS actifs sur le STP local : %d" % actifs)

        asp_out = pool.cmd(node.name, stp.port, "show cs7 instance 0 asp")
        asp_actifs = len(re.findall(r"ASP_ACTIVE", asp_out))
        (ok if asp_actifs >= 2 else fail)("ASP actifs : %d" % asp_actifs)

        route_out = pool.cmd(node.name, stp.port, "show cs7 instance 0 route")
        if re.search(r"0\.0\.0/0.*avail", route_out):
            ok("route par defaut vers l'inter-STP : presente et avail")
        else:
            fail("route par defaut vers l'inter-STP absente ou indisponible")
        # PROHIB : on distingue une vraie panne (un noeud DU LAB devenu
        # inaccessible) d'une route dynamique laissee par un point code de
        # passage - une console SS7 arretee, un noeud WAN eteint. Les
        # confondre ferait crier au feu a chaque fois qu'on ferme la console.
        connus = set()
        for autre in topo.operators:
            if autre.point_code:
                connus.add(autre.point_code)
            for a in autre.sccp.values():
                if a.get("pc"):
                    connus.add(a["pc"])
        durs, mous = [], []
        for ligne in route_out.split("\n"):
            if "PROHIB" not in ligne:
                continue
            dest = ligne.split("/")[0].strip()
            (durs if dest in connus else mous).append(dest)
        if durs:
            fail("routes PROHIB vers des noeuds du lab : %s" % ", ".join(durs))
        elif mous:
            warn("%d route(s) PROHIB, toutes vers des point codes de passage : %s"
                 % (len(mous), ", ".join(mous)))
            info("dynamiques : elles repasseront avail des que ces PC reviendront")
        else:
            ok("aucune route PROHIB")

        users = pool.cmd(node.name, stp.port, "show cs7 instance 0 users")
```

```
        for ssn, nom in ((254, "BSSAP"), (6, "HLR"), (8, "MSC")):
            if re.search(r"\b%d\b" % ssn, users):
                info("SSN %s (%s) enregistre sur le STP" % (ssn, nom))

        # Sans sondage des VTY (--no-probe), on retombe sur les ports usuels
        # plutot que d'annoncer un MSC ou un HLR "absent" qui tourne tres bien.
        def service(daemon, defaut):
            svc = node.service(daemon)
            if svc:
                return svc
            return type("S", (), {"port": defaut, "label": daemon})()

        try:
            msc = service("OsmoMSC", 4254)
            cache = pool.cmd(node.name, msc.port, "show subscriber cache")
            n = len(re.findall(r"IMSI", cache))
            ok("MSC : VTY %s, %d abonne(s) en VLR" % (msc.port, n))
        except vty_mod.VtyError as exc:
            fail("MSC : VTY injoignable (%s)" % exc)

        try:
            hlr = service("OsmoHLR", 4258)
            subs = pool.cmd(node.name, hlr.port, "show subscribers all")
            n = len(re.findall(r"^\s*\d+\s+\d+", subs, re.M))
            gsup = pool.cmd(node.name, hlr.port, "show gsup-connections")
            vlr = "VLR" in gsup
            (ok if n else warn)("HLR : %d abonne(s) provisionne(s)" % n)
            (ok if vlr else fail)("HLR : VLR %s en GSUP"
                                  % ("connecte" if vlr else "absent"))
        except vty_mod.VtyError as exc:
            fail("HLR : VTY injoignable (%s)" % exc)

        try:
            bsc = service("OsmoBSC", 4242)
            bts = pool.cmd(node.name, bsc.port, "show bts")
            up = len(re.findall(r"(?i)OML.*Link.*connected|BTS is in service", bts))
            info("BSC : VTY %s%s" % (bsc.port,
                                     ", %d BTS en service" % up if up else ""))
        except vty_mod.VtyError as exc:
            warn("BSC : VTY injoignable (%s)" % exc)


# -- 4. lien vers le hub, vu des noeuds ----------------------------------------
def check_hub_link(topo):
    section("4. LIEN VERS L'INTER-STP (vu des noeuds)")
    if not topo.hub_ip:
        skip("pas de hub connu")
        return
    port = topo.hub_port
    for node in topo.operators:
        out_ = sh("docker exec %s ss -an --sctp 2>/dev/null" % node.name) \
            if topo.docker else sh("ss -an --sctp")
        if not out_.strip():
            skip("%s : ss --sctp indisponible" % node.title)
            continue
        motif = re.escape("%s:%d" % (topo.hub_ip, port))
        if re.search(r"ESTAB.*%s" % motif, out_):
            ok("%s : association SCTP ETABLIE vers %s:%d"
               % (node.title, topo.hub_ip, port))
        else:
            fail("%s : aucune association SCTP vers %s:%d - l'ASP est DOWN"
                 % (node.title, topo.hub_ip, port))


# -- 5. la console comme ASP ---------------------------------------------------
def check_console_asp(topo):
```

```
    section("5. LA CONSOLE COMME ASP M3UA")
    if not topo.hub_ip:
        skip("pas de hub : aucun lien a monter")
        return None
    pc = topo.free_point_codes(2)[0]
    client = ss7.Ss7Client(topo.hub_ip, topo.hub_port, pc)
    try:
        etat = client.connect()
    except Exception as exc:
        fail("attachement impossible : %s" % exc)
        info("verifiez que %s:%s accepte les ASP dynamiques" % (topo.hub_ip,
                                                                topo.hub_port))
        return None
    if etat == "ASP_ACTIVE":
        ok("ASP actif sur le hub : PC %s, routing context %s"
           % (pc, client.asp.routing_ctx))
    else:
        fail("ASP dans l'etat %s" % etat)
        return client

    cibles = []
    for node in topo.operators:
        if node.point_code:
            cibles.append((node.title + " STP", node.point_code))
        for nom, addr in sorted(node.sccp.items()):
            if not nom.startswith("_self_") and addr.get("pc"):
                cibles.append(("%s %s" % (node.title, nom), addr["pc"]))
    vus = set()
    for label, cible in cibles:
        if cible in vus:
            continue
        vus.add(cible)
        rep = client.audit(cible)
        if rep.ok:
            ok("DAUD %-10s %-8s : joignable a travers le hub" % (label, cible))
        else:
            warn("DAUD %-10s %-8s : %s" % (label, cible, rep.lines[0]))
    return client


# ── 6. aller-retour MAP reel ──────────────────────────────────────────
def check_map(topo, client, pool):
    section("6. ALLER-RETOUR MAP (sendRoutingInfoForSM)")
    if client is None:
        skip("pas de lien M3UA")
        return
    if not topo.operators:
        skip("aucun operateur")
        return
    pc_rep = topo.free_point_codes(2)[1]
    node = topo.operators[0]
    rep = responder_mod.MapResponder(topo, pc_rep, mapops.SSN_HLR, node=node.name)
    try:
        rep.start()
    except Exception as exc:
        fail("repondeur MAP impossible a demarrer : %s" % exc)
        return
    abonnes = rep.subscribers()
    if not abonnes:
        skip("aucun abonne dans le HLR de %s : rien a interroger" % node.name)
        rep.stop()
        return
    ok("repondeur MAP en place : PC %s, SSN 6, %d abonne(s) du HLR reel"
       % (pc_rep, len(abonnes)))
    cible = abonnes[0]
    info("interrogation du MSISDN %s (IMSI attendu %s)" % (cible.msisdn, cible.imsi))
```

```
    time.sleep(0.5)
    rapport = client.send_map("sri-sm", {"msisdn": cible.msisdn,
                                         "sc_addr": cible.msisdn,
                                         "sm_rp_pri": "1"},
                              dest_pc=pc_rep, timeout=5.0)
    trouve = dict(rapport.result).get("IMSI", "")
    if trouve == cible.imsi:
        ok("MAP complet : SRI-SM -> IMSI %s (le TCAP a bien traverse le hub)"
           % trouve)
    elif rapport.ok:
        warn("reponse MAP recue, mais IMSI inattendu : %s" % trouve)
    else:
        fail("aucune reponse MAP exploitable")
        for ligne in rapport.lines[-6:]:
            info(ligne)
    rep.stop()


def main():
    ap = argparse.ArgumentParser(description="Diagnostic SS7 complet du lab.")
    ap.add_argument("--rapide", action="store_true",
                    help="sauter les tests M3UA et MAP")
    ap.add_argument("--log", help="ecrire aussi le rapport dans ce fichier")
    ap.add_argument("--no-probe", action="store_true",
                    help="ne pas sonder les ports VTY")
    args = ap.parse_args()

    if args.log:
        STATE["log"] = open(args.log, "w")

    out("")
    out("%s============================================================%s" % (C, N))
    out("%s  DIAGNOSTIC SS7 - osmo-operator - %s%s"
        % (C, time.strftime("%Y-%m-%d %H:%M:%S"), N))
    out("%s============================================================%s" % (C, N))

    check_env()
    topo = topo_mod.discover(probe=not args.no_probe)
    check_topo(topo)
    pool = vty_mod.VtyPool(docker=topo.docker)
    client = None
    try:
        check_nodes(topo, pool)
        check_hub_link(topo)
        if not args.rapide:
            client = check_console_asp(topo)
            check_map(topo, client, pool)
        else:
            section("5-6. M3UA ET MAP")
            skip("ignores (--rapide)")
    finally:
        pool.close_all()
        if client:
            client.close()

    section("RESUME")
    out("  %s%d ok%s   %s%d avertissement(s)%s   %s%d echec(s)%s   %d ignore(s)"
        % (G, STATE["pass"], N, Y, STATE["warn"], N, R, STATE["fail"], N,
           STATE["skip"]))
    if STATE["fail"] == 0 and STATE["warn"] == 0:
        out("  %sLe SS7 du lab est sain de bout en bout.%s" % (G, N))
    elif STATE["fail"] == 0:
        out("  %sPas de panne, mais des points a surveiller.%s" % (Y, N))
    else:
        out("  %sDes elements du SS7 sont en panne (voir ci-dessus).%s" % (R, N))
    out("")
```

```
    if STATE["log"]:
        STATE["log"].close()
    return STATE["fail"]


if __name__ == "__main__":
    sys.exit(main())
```

3.239 /opt/GSM/osmo-operator/navigation/ss7.py

9284 octets, 225 lignes → 225 lignes

```
# ss7.py - la couche qui relie tout : M3UA + SCCP + TCAP + MAP.
#
# C'est l'objet que manipulent la console et les scripts : on lui donne une
# operation du catalogue, une destination (point code, SSN, GT), il fabrique
# l'UDT, l'emet sur le lien M3UA et rend un compte rendu lisible de ce qui
# revient - resultat MAP, erreur MAP, ou refus SCCP.

import os
import random
import time

from . import m3ua
from . import mapops
from . import sccp
from . import tcap


class Ss7Error(Exception):
    pass


class Report(object):
    def __init__(self):
        self.lines = []
        self.ok = False
        self.raw_tx = b""
        self.raw_rx = b""
        self.result = []

    def add(self, text):
        self.lines.append(text)

    def __str__(self):
        return "\n".join(self.lines)


class Ss7Client(object):
    def __init__(self, hub_ip, hub_port=2908, local_pc="1.11.60",
                 local_ssn=mapops.SSN_MSC, local_gt="", log=None):
        self.hub_ip = hub_ip
        self.hub_port = int(hub_port)
        self.local_pc = local_pc
        self.local_ssn = local_ssn
        self.local_gt = local_gt
        self.asp = None
        self.invoke_id = 1
        self.log_cb = log

    # -- lien ----------------------------------------------------------
    @property
    def connected(self):
        return self.asp is not None and self.asp.state == "ASP_ACTIVE"
```

```
@property
def events(self):
    return self.asp.events if self.asp else []

def connect(self, register=True):
    if not self.hub_ip:
        raise Ss7Error(
            "aucun inter-STP connu : ni conteneur osmo-inter-stp, ni "
            "OSMO_HUB_IP dans les noeuds. Demarrez le lab, ou donnez "
            "l'adresse du hub avec --hub-ip.")
    self.asp = m3ua.Asp(self.hub_ip, self.hub_port, log=self.log_cb)
    self.asp.bring_up(sccp.pc_to_int(self.local_pc), register=register)
    return self.asp.state

def fresh(self):
    """Un lien jeune avant chaque echange.

    L'inter-STP du lab ne sert une association venue de l'hote qu'une
    trentaine de secondes ; au-dela, les messages partent dans le vide.
    Plutot que d'afficher un faux "pas de reponse", on remonte le lien -
    c'est invisible pour l'appelant, et le point code ne change pas."""
    if self.asp is None:
        self.connect()
        return True
    return self.asp.renew()

def close(self):
    if self.asp:
        self.asp.close()
    self.asp = None

# ── audit de destination (le "ping" du SS7) ──────────────────────
def audit(self, point_code, timeout=3.0):
    """DAUD : le STP repond DAVA (joignable) ou DUNA (inaccessible)."""
    rep = Report()
    if not self.asp:
        rep.add("pas de lien M3UA")
        return rep
    self.fresh()
    val = sccp.pc_to_int(point_code)
    body = m3ua.param(m3ua.P_AFFECTED_PC, bytes([0]) + val.to_bytes(3, "big"))
    if self.asp.routing_ctx is not None:
        body += m3ua.param(m3ua.P_ROUTING_CTX,
                           self.asp.routing_ctx.to_bytes(4, "big"))
    self.asp.send_msg(m3ua.CLS_SSNM, m3ua.DAUD, body)
    msg = self.asp.wait_ctrl({(m3ua.CLS_SSNM, m3ua.DAVA),
                              (m3ua.CLS_SSNM, m3ua.DUNA),
                              (m3ua.CLS_SSNM, m3ua.SCON),
                              (m3ua.CLS_MGMT, m3ua.ERR)}, timeout)
    if msg is None:
        rep.add("DAUD %s : aucune reponse du STP" % point_code)
        return rep
    if msg.typ == m3ua.DAVA and msg.cls == m3ua.CLS_SSNM:
        rep.ok = True
        rep.add("DAVA - le point code %s est JOIGNABLE via le hub" % point_code)
    elif msg.typ == m3ua.DUNA:
        rep.add("DUNA - le point code %s est INACCESSIBLE" % point_code)
    elif msg.typ == m3ua.SCON:
        rep.add("SCON - congestion vers %s" % point_code)
    else:
        rep.add("reponse inattendue : %s" % msg.name)
    return rep

# ── envoi d'une operation MAP ────────────────────────────────────
def send_map(self, op, values, dest_pc, dest_ssn=None, dest_gt="",
```

```
         route_on_gt=False, timeout=6.0, essais=2):
if isinstance(op, str):
    op = mapops.BY_KEY[op]
rep = Report()
if not self.asp:
    rep.add("pas de lien M3UA - connectez d'abord la console au STP")
    return rep
self.fresh()

ssn = int(dest_ssn if dest_ssn is not None else op.ssn)
param_ber = op.encode(values)
opcode = int(values.get("opcode", op.opcode) or op.opcode)
ac = values.get("ac") or op.ac

otid = os.urandom(4)
comp = tcap.invoke(self.invoke_id, opcode, param_ber)
self.invoke_id = (self.invoke_id % 127) + 1
payload = tcap.begin(otid, comp, ac or None)

called = sccp.Address(pc=None if route_on_gt else dest_pc, ssn=ssn,
                      gt=dest_gt or None, route_on_ssn=not route_on_gt)
calling = sccp.Address(pc=self.local_pc, ssn=self.local_ssn,
                       gt=self.local_gt or None,
                       route_on_ssn=not self.local_gt)
udt = sccp.udt(called, calling, payload)
rep.raw_tx = udt

rep.add("Emission")
rep.add("  operation   : %s (opcode %d)" % (op.name, opcode))
if ac:
    rep.add("  contexte    : %s" % ac)
rep.add("  destination : %s" % called)
rep.add("  origine     : %s" % calling)
rep.add("  TCAP begin  : otid %s, %d octets" % (otid.hex(), len(payload)))

self.asp.send_data(sccp.pc_to_int(self.local_pc),
                   sccp.pc_to_int(dest_pc if not route_on_gt else dest_pc),
                   udt)

deadline = time.time() + timeout
while time.time() < deadline:
    data = self.asp.recv_data(timeout=max(0.2, deadline - time.time()))
    if data is None:
        continue
    dec = sccp.decode(data["data"])
    rep.raw_rx = data["data"]
    if dec is None:
        continue
    rep.add("")
    rep.add("Reponse recue de PC %s" % m3ua._pc(data["opc"]))
    rep.add("  SCCP : %s" % dec)
    if dec.kind.endswith("UDTS"):
        rep.add("  -> le reseau refuse : %s" % dec.cause_text)
        return rep
    msg = tcap.parse(dec.data)
    if msg is None:
        rep.add("  charge non TCAP : %s" % dec.data.hex())
        return rep
    rep.add("  TCAP : %s" % msg)
    for comp in msg.components:
        if comp.kind == "returnResultLast":
            rep.ok = True
            rep.add("  resultat (opcode %s) :" % comp.opcode)
            for label, value in op.decoder(comp.parameter):
                rep.add("     %-28s %s" % (label, value))
                rep.result.append((label, value))
```

```
                elif comp.kind == "returnError":
                    rep.add("  ERREUR MAP %s : %s" %
                            (comp.error, mapops.error_name(comp.error)))
                elif comp.kind == "reject":
                    rep.add("  REJECT TCAP : %s" %
                            (comp.parameter.value.hex() if comp.parameter else ""))
                else:
                    rep.add("  composant %s" % comp.kind)
            if msg.kind == "abort":
                rep.add("  ABORT TCAP : %s" % msg.abort_reason)
            return rep

        if essais > 1:
            # Le lien vers le hub ne vit que quelques secondes : un envoi peut
            # tomber pendant que l'un des deux bouts renouvelle son
            # association. Un second essai, sur un lien neuf, tranche entre
            # "personne n'ecoute" et "mauvais moment".
            rep.add("")
            rep.add("Pas de reponse - second essai sur un lien neuf ...")
            self.asp.renew(max_age=0)
            suite = self.send_map(op, values, dest_pc, dest_ssn, dest_gt,
                                  route_on_gt, timeout, essais=essais - 1)
            rep.lines.extend([""] + suite.lines)
            rep.ok = suite.ok
            rep.result = suite.result
            return rep
        rep.add("")
        rep.add("Aucune reponse au bout de %.0f s." % timeout)
        rep.add("Sur ce lab, un silence signifie en general qu'aucun demon")
        rep.add("n'ecoute ce SSN : Osmocom parle GSUP entre MSC et HLR, et")
        rep.add("reserve le SCCP a l'interface A (SSN 254).")
        return rep


def default_client(topo, local_pc=None, log=None):
    """Fabrique un client a partir de la topologie decouverte : hub, port et
    point code libre sortent tous de la configuration du lab."""
    pc = local_pc or topo.free_point_code()
    gt = ""
    return Ss7Client(topo.hub_ip or "127.0.0.1", topo.hub_port, pc,
                     local_gt=gt, log=log)
```

3.240 /opt/GSM/osmo-operator/navigation/tcap.py

6611 octets, 188 lignes → 188 lignes

```
# tcap.py - TCAP (Q.773) : begin / continue / end / abort, et les composants
# invoke / returnResultLast / returnError / reject.
#
# La console emet un begin contenant UN invoke, et lit ce qui revient. Le
# dialogue (AARQ/AARE) porte le contexte applicatif MAP : sans lui, un vrai
# HLR repond "abort" - on le met donc systematiquement.

from . import ber

# Tags applicatifs des messages TCAP.
BEGIN, END, CONTINUE, ABORT = 0x62, 0x64, 0x65, 0x67
TID_ORIG, TID_DEST = 0x48, 0x49
DIALOGUE, COMPONENTS = 0x6B, 0x6C

# Composants.
INVOKE, RET_RES_LAST, RET_ERROR, REJECT = 0xA1, 0xA2, 0xA3, 0xA4

DIALOGUE_AS_ID = "0.0.17.773.1.1.1"
```

```
MSG_NAME = {BEGIN: "begin", END: "end", CONTINUE: "continue", ABORT: "abort"}
COMP_NAME = {INVOKE: "invoke", RET_RES_LAST: "returnResultLast",
             RET_ERROR: "returnError", REJECT: "reject"}


def dialogue_request(app_context_oid):
    """dialoguePortion contenant un AARQ (demande d'ouverture)."""
    aarq = ber.tlv(0x60,
                   ber.tlv(0x80, b"\x07\x80") +              # version 1
                   ber.ctx(1, ber.enc_oid(app_context_oid)))
    external = ber.tlv(0x28,
                       ber.enc_oid(DIALOGUE_AS_ID) +
                       ber.ctx(0, aarq))
    return ber.tlv(DIALOGUE, external)


def invoke(invoke_id, opcode, parameter=b""):
    body = ber.enc_int(invoke_id) + ber.enc_int(opcode)
    if parameter:
        body += parameter
    return ber.tlv(INVOKE, body)


def begin(otid, components, app_context_oid=None):
    body = ber.tlv(TID_ORIG, otid)
    if app_context_oid:
        body += dialogue_request(app_context_oid)
    body += ber.tlv(COMPONENTS, components)
    return ber.tlv(BEGIN, body)


def end(dtid, components=b"", app_context_oid=None):
    body = ber.tlv(TID_DEST, dtid)
    if app_context_oid:
        body += dialogue_request(app_context_oid)
    if components:
        body += ber.tlv(COMPONENTS, components)
    return ber.tlv(END, body)


class Component(object):
    def __init__(self, kind, invoke_id=None, opcode=None, error=None,
                 parameter=None, raw=b""):
        self.kind = kind
        self.invoke_id = invoke_id
        self.opcode = opcode
        self.error = error
        self.parameter = parameter
        self.raw = raw


class Message(object):
    def __init__(self, kind="?", otid=b"", dtid=b"", components=None,
                 app_context="", abort_reason="", raw=b""):
        self.kind = kind
        self.otid = otid
        self.dtid = dtid
        self.components = components or []
        self.app_context = app_context
        self.abort_reason = abort_reason
        self.raw = raw

    def __str__(self):
        s = "TCAP %s" % self.kind
        if self.otid:
```

```
            s += " otid=%s" % self.otid.hex()
        if self.dtid:
            s += " dtid=%s" % self.dtid.hex()
        if self.app_context:
            s += " ac=%s" % self.app_context
        for c in self.components:
            s += " | %s" % c.kind
        return s


def _oid_str(raw):
    if not raw:
        return ""
    vals = [raw[0] // 40, raw[0] % 40]
    n = 0
    for b in raw[1:]:
        n = (n << 7) | (b & 0x7F)
        if not b & 0x80:
            vals.append(n)
            n = 0
    return ".".join(str(v) for v in vals)


def parse(data):
    """Rend un Message, ou None si ce n'est pas du TCAP."""
    top = ber.decode(data)
    if not top:
        return None
    e = top[0]
    kind = MSG_NAME.get(e.tag[0], "0x%02x" % e.tag[0])
    msg = Message(kind=kind, raw=data)
    for child in e.children:
        t = child.tag[0]
        if t == TID_ORIG:
            msg.otid = child.value
        elif t == TID_DEST:
            msg.dtid = child.value
        elif t == DIALOGUE:
            for sub in child.walk():
                if sub.cls == 0 and sub.num == 6 and len(sub.value) > 3:
                    oid = _oid_str(sub.value)
                    if oid != DIALOGUE_AS_ID:
                        msg.app_context = oid
        elif t == COMPONENTS:
            for comp in child.children:
                msg.components.append(_parse_component(comp))
        elif kind == "abort":
            msg.abort_reason = child.tagstr() + " " + child.value.hex()
    return msg


def _parse_component(comp):
    kind = COMP_NAME.get(comp.tag[0], comp.tagstr())
    invoke_id = opcode = error = None
    parameter = None
    kids = comp.children
    if kind == "invoke" and len(kids) >= 2:
        invoke_id = int.from_bytes(kids[0].value, "big", signed=True)
        opcode = int.from_bytes(kids[1].value, "big", signed=True)
        parameter = kids[2] if len(kids) > 2 else None
    elif kind == "returnResultLast" and kids:
        invoke_id = int.from_bytes(kids[0].value, "big", signed=True)
        if len(kids) > 1 and kids[1].constructed and kids[1].children:
            inner = kids[1].children
            opcode = int.from_bytes(inner[0].value, "big", signed=True)
            parameter = inner[1] if len(inner) > 1 else None
```

```
    elif kind == "returnError" and len(kids) >= 2:
        invoke_id = int.from_bytes(kids[0].value, "big", signed=True)
        error = int.from_bytes(kids[1].value, "big", signed=True)
        parameter = kids[2] if len(kids) > 2 else None
    elif kind == "reject" and kids:
        invoke_id = int.from_bytes(kids[0].value, "big", signed=True)
        parameter = kids[1] if len(kids) > 1 else None
    return Component(kind, invoke_id, opcode, error, parameter, comp.value)


def dialogue_response(app_context_oid, result=0, diagnostic=0):
    """dialoguePortion contenant un AARE (acceptation d'ouverture)."""
    aare = ber.tlv(0x61,
                   ber.tlv(0x80, b"\x07\x80") +
                   ber.ctx(1, ber.enc_oid(app_context_oid)) +
                   ber.ctx(2, ber.enc_int(result)) +
                   ber.ctx(3, ber.ctx(1, ber.enc_int(diagnostic))))
    external = ber.tlv(0x28, ber.enc_oid(DIALOGUE_AS_ID) + ber.ctx(0, aare))
    return ber.tlv(DIALOGUE, external)


def return_result_last(invoke_id, opcode, parameter=b""):
    inner = ber.enc_int(opcode) + parameter
    return ber.tlv(RET_RES_LAST, ber.enc_int(invoke_id) + ber.tlv(0x30, inner))


def return_error(invoke_id, error_code, parameter=b""):
    return ber.tlv(RET_ERROR,
                   ber.enc_int(invoke_id) + ber.enc_int(error_code) + parameter)


def end_response(dtid, components, app_context_oid=None):
    """end + AARE : la reponse complete a un begin."""
    body = ber.tlv(TID_DEST, dtid)
    if app_context_oid:
        body += dialogue_response(app_context_oid)
    body += ber.tlv(COMPONENTS, components)
    return ber.tlv(END, body)
```

3.241 /opt/GSM/osmo-operator/navigation/test-console.sh

8261 octets, 205 lignes → 205 lignes

```
#!/usr/bin/env bash
# test-console.sh - passe en revue CHAQUE commande de la console SS7 et de ses
# outils, garde un journal par commande, et rend un rapport.
#
#   ./navigation/test-console.sh              tout, y compris le SS7 reel
#   ./navigation/test-console.sh --rapide     sans M3UA/MAP (lab a l'arret)
#   ./navigation/test-console.sh --dir CHEMIN journaux ailleurs
#
# Un test = une commande + un critere de reussite (motif attendu dans la
# sortie, ou code de retour). Chaque commande a SON fichier de journal, nomme
# par le numero et le nom du test : on peut relire exactement ce qui s'est
# passe sans relancer quoi que ce soit.
#
# Codes de sortie : 0 tout passe, sinon le nombre d'echecs.

set -u

HERE="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
REPO="$(dirname "$HERE")"
CONSOLE="$HERE/ss7-console.py"
DIAG="$HERE/ss7-diag.py"
```

```
RAPIDE=0
LOGDIR=""
while [ $# -gt 0 ]; do
    case "$1" in
        --rapide) RAPIDE=1 ;;
        --dir)    LOGDIR="${2:-}"; shift ;;
        --dir=*)  LOGDIR="${1#*=}" ;;
        -h|--help)
            sed -n '2,14p' "${BASH_SOURCE[0]}" | sed 's/^# \{0,1\}//'
            exit 0 ;;
        *) echo "argument inconnu : $1" >&2; exit 2 ;;
    esac
    shift
done

STAMP="$(date +%Y%m%d-%H%M%S)"
LOGDIR="${LOGDIR:-$HERE/logs/$STAMP}"
mkdir -p "$LOGDIR"
RAPPORT="$LOGDIR/rapport.txt"

G='\033[0;32m'; R='\033[0;31m'; Y='\033[1;33m'; C='\033[0;36m'; B='\033[1m'; N='\033[0m'
PASS=0; FAIL=0; SKIP=0; NUM=0

dire() { echo -e "$*"; echo -e "$*" | sed 's/\x1b\[[0-9;]*m//g' >> "$RAPPORT"; }

# essai <nom> <motif attendu|-> <commande...>
#   motif "-" : on ne juge que le code de retour.
essai() {
    local nom="$1" motif="$2"; shift 2
    NUM=$((NUM + 1))
    local fichier
    fichier="$(printf '%02d-%s.log' "$NUM" "$nom")"
    local chemin="$LOGDIR/$fichier"
    {
        echo "# test   : $nom"
        echo "# commande : $*"
        echo "# attendu : ${motif}"
        echo "# date   : $(date '+%Y-%m-%d %H:%M:%S')"
        echo "# ----------------------------------------------------------------"
    } > "$chemin"
    local debut fin rc
    debut=$(date +%s)
    timeout 240 "$@" >> "$chemin" 2>&1
    rc=$?
    fin=$(date +%s)
    echo "# ----------------------------------------------------------------" >> "$chemin"
    echo "# code de retour : $rc   duree : $((fin - debut)) s" >> "$chemin"

    if [ "$rc" -eq 124 ]; then
        FAIL=$((FAIL + 1))
        dire "  ${R}✗${N} ${nom} — delai depasse (240 s)   ${C}${fichier}${N}"
        return
    fi
    if [ "$motif" = "-" ]; then
        if [ "$rc" -eq 0 ]; then
            PASS=$((PASS + 1)); dire "  ${G}✓${N} ${nom}   ${C}${fichier}${N}"
        else
            FAIL=$((FAIL + 1)); dire "  ${R}✗${N} ${nom} — code $rc   ${C}${fichier}${N}"
        fi
        return
    fi
    if grep -qE "$motif" "$chemin"; then
        PASS=$((PASS + 1)); dire "  ${G}✓${N} ${nom}   ${C}${fichier}${N}"
    else
        FAIL=$((FAIL + 1))
```

```
        dire "  ${R}✗${N} ${nom} — motif absent : ${motif}   ${C}${fichier}${N}"
    fi
}

ignore() { NUM=$((NUM + 1)); SKIP=$((SKIP + 1)); dire "  ${Y}-${N} $1"; }

titre() { dire ""; dire "${B}$1${N}"; dire
 ↪  "------------------------------------------------------------------------"; }

# ── le lab est-il debout ? ──────────────────────────────────────────
CONTENEURS="$(docker ps --format '{{.Names}}' 2>/dev/null | grep -c '^osmo-' || true)"
HUB=""
if [ "${CONTENEURS:-0}" -gt 0 ]; then
    HUB="$(docker exec "$(docker ps --format '{{.Names}}' | grep '^osmo-operator-' | head -1)" \
           printenv OSMO_HUB_IP 2>/dev/null | tr -d '\r')"
fi

dire ""
dire "${C}========================================================================${N}"
dire "${C}  TEST DE LA CONSOLE SS7 - $(date '+%Y-%m-%d %H:%M:%S')${N}"
dire "${C}========================================================================${N}"
dire "  depot      : $REPO"
dire "  journaux   : $LOGDIR"
dire "  conteneurs : ${CONTENEURS:-0}   hub : ${HUB:-aucun}"
[ "$RAPIDE" = "1" ] && dire "  mode       : rapide (ni M3UA ni MAP)"

titre "1. LE PAQUET S'IMPORTE"
essai "import-modules" "modules ok" python3 -c "
import sys; sys.path.insert(0, '$REPO')
from navigation import ber, sccp, tcap, mapops, m3ua, ss7, topo, vty, diagram, quickcmd, responder
print('modules ok')"
essai "encodage-ber-sccp" "encodage ok" python3 -c "
import sys; sys.path.insert(0, '$REPO')
from navigation import sccp, tcap, mapops, ber
a = sccp.Address(pc='1.11.2', ssn=6)
assert sccp.Address.decode(a.encode()).pc == '1.11.2'
assert sccp.pc_to_int('1.11.2') == (1 << 11) | (11 << 3) | 2
assert sccp.untbcd(sccp.tbcd('600101')) == '600101'
u = sccp.udt(a, sccp.Address(pc='1.11.7', ssn=8), b'\x62\x03\x48\x01\x01')
d = sccp.decode(u); assert d.kind == 'UDT' and d.called.ssn == 6
m = tcap.parse(tcap.begin(b'\x01\x02\x03\x04', tcap.invoke(1, 45, b''), mapops.AC['shortMsgGateway']))
assert m.kind == 'begin' and m.components[0].opcode == 45
print('encodage ok')"
essai "aide-console" "usage: ss7-console" "$CONSOLE" --help
essai "aide-diag" "usage: ss7-diag" "$DIAG" --help

titre "2. INVENTAIRE ET CATALOGUE"
essai "console-ops" "sendRoutingInfoForSM" "$CONSOLE" ops
essai "console-list" "Hub inter-STP" "$CONSOLE" --no-probe list
essai "console-list-sonde" "Hub inter-STP" "$CONSOLE" list

titre "3. VTY (un test par demon)"
if [ "${CONTENEURS:-0}" -gt 0 ]; then
    essai "vty-stp"   "Routing|AS Name"   "$CONSOLE" vty 1 4239 "show cs7 instance 0 as all"
    essai "vty-stp-route" "Routing table" "$CONSOLE" vty 1 4239 "show cs7 instance 0 route"
    essai "vty-msc"   "IMSI|Subscriber"   "$CONSOLE" vty 1 4254 "show subscriber cache"
    essai "vty-hlr"   "IMSI|MSISDN"       "$CONSOLE" vty 1 4258 "show subscribers all"
    essai "vty-hlr-gsup" "VLR|GSUP|from"  "$CONSOLE" vty 1 4258 "show gsup-connections"
    essai "vty-bsc"   "BTS|bts"           "$CONSOLE" vty 1 4242 "show bts"
    essai "vty-inconnu-propre" "injoignable" "$CONSOLE" vty 1 4299 "show version"
else
    ignore "VTY : aucun conteneur ne tourne"
fi

titre "4. SS7 REEL (M3UA, DAUD, MAP)"
if [ "$RAPIDE" = "1" ]; then
```

```
    ignore "M3UA et MAP : mode rapide"
elif [ -z "$HUB" ]; then
    ignore "M3UA et MAP : aucun hub connu (lab arrete ?)"
else
    essai "m3ua-audit-stp" "JOIGNABLE|DAVA" "$CONSOLE" audit 1.11.2
    essai "m3ua-audit-msc" "JOIGNABLE|DAVA" "$CONSOLE" audit 1.11.1
    essai "m3ua-audit-inconnu" "INACCESSIBLE|DUNA|aucune reponse" "$CONSOLE" audit 7.255.7
    essai "map-sans-destination" "point code|global title" "$CONSOLE" map sri-sm msisdn=600101
    essai "map-operation-inconnue" "operation inconnue" "$CONSOLE" map pas-une-op --pc 1.11.2
    # Aller-retour complet : le repondeur repond a l'emetteur a travers le hub.
    essai "map-aller-retour" "IMSI" "$DIAG" --no-probe
fi

titre "5. DIAGNOSTIC COMPLET"
essai "diag-rapide" "RESUME" "$DIAG" --rapide --no-probe --log "$LOGDIR/diag-rapide.txt"
if [ "$RAPIDE" != "1" ] && [ -n "$HUB" ]; then
    essai "diag-complet" "RESUME" "$DIAG" --log "$LOGDIR/diag-complet.txt"
else
    ignore "diagnostic complet : lab a l'arret ou mode rapide"
fi

titre "6. LE SCHEMA (interface aux fleches)"
essai "schema-rendu" "INTER-STP" python3 -c "
import sys; sys.path.insert(0, '$REPO')
from navigation import topo, diagram
t = topo.discover(probe=False)
lignes, boites = diagram.build(t)
print('\n'.join(lignes))
print('%d boites' % len(boites))
assert boites, 'aucune boite'
i = 0
for direction in ((0, 1), (1, 0), (0, -1), (-1, 0)):
    i = diagram.navigate(boites, i, direction)
print('navigation ok')"
essai "schema-navigation" "navigation ok" python3 -c "
import sys; sys.path.insert(0, '$REPO')
from navigation import topo, diagram
t = topo.discover(probe=False)
_, boites = diagram.build(t)
vus = set()
i = 0
for _ in range(40):
    for d in ((0,1),(1,0),(0,-1),(-1,0)):
        i = diagram.navigate(boites, i, d); vus.add(i)
assert len(vus) >= min(3, len(boites))
print('navigation ok')"

titre "RESUME"
dire "  ${G}${PASS} reussi(s)${N}   ${R}${FAIL} echec(s)${N}   ${Y}${SKIP} ignore(s)${N}"
dire "  journaux : ${C}${LOGDIR}${N}  (un fichier par commande)"
dire "  rapport  : ${C}${RAPPORT}${N}"
dire ""
exit "$FAIL"
```

3.242 /opt/GSM/osmo-operator/navigation/topo.py

14766 octets, 425 lignes → 425 lignes

```
# topo.py - inventaire du lab, HERITE des configurations.
#
# Rien n'est ecrit en dur : les point codes, les adresses, les ports et les
# services sont lus la ou ils font foi.
#
#   1. "docker ps"                      -> quels noeuds tournent ici ;
```

```
#   2. l'environnement du conteneur     -> OSMO_HUB_IP, WAN_NODE_ID (poses par
#                                          start-docker.sh, cf. --hub-ip / --wan) ;
#   3. /etc/osmocom/*.cfg DANS le noeud -> point code, routing keys, listeners
#                                          M3UA, adresses SCCP, liens vers le
#                                          hub. C'est la configuration REELLE,
#                                          pas le gabarit du depot ;
#   4. /proc/net/tcp DANS le noeud      -> quels VTY ecoutent vraiment, et la
#                                          banniere de chacun dit QUEL demon
#                                          repond (OsmoMSC, OsmoHLR, ...).
#
# Les gabarits de configs/ ne servent que de repli quand aucun noeud ne tourne.

import json
import os
import re
import subprocess

REPO = os.path.dirname(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))

# Sonde poussee dans le noeud : liste les VTY qui ecoutent et lit leur
# banniere. Une seule execution par noeud, tout en parallele.
_PROBE = r'''
import json, socket, sys, threading

def listening():
    ports = set()
    for path in ("/proc/net/tcp", "/proc/net/tcp6"):
        try:
            with open(path) as fh:
                next(fh, None)
                for line in fh:
                    f = line.split()
                    if len(f) < 4 or f[3] != "0A":
                        continue
                    try:
                        p = int(f[1].split(":")[1], 16)
                    except Exception:
                        continue
                    ports.add(p)
        except Exception:
            pass
    return sorted(p for p in ports if 4200 <= p <= 4299)

res = {}
lock = threading.Lock()

def probe(p):
    banner = ""
    try:
        s = socket.create_connection(("127.0.0.1", p), 1.5)
        s.settimeout(1.2)
        data = b""
        try:
            while len(data) < 4096:
                d = s.recv(4096)
                if not d:
                    break
                data += d
                if b"VTY interface" in data:
                    break
        except socket.timeout:
            pass
        s.close()
        for line in data.decode("utf-8", "replace").splitlines():
            if "VTY interface" in line:
                banner = line.strip()
```

```
                break
    except Exception:
        return
    with lock:
        res[p] = banner

th = [threading.Thread(target=probe, args=(p,)) for p in listening()]
for t in th:
    t.start()
for t in th:
    t.join(4)
print(json.dumps(res))
'''

# Nom court affiche dans le schema, deduit de la banniere ("OsmoMSC" -> MSC).
_PRETTY = {
    "OsmoSTP": ("STP", "SS7"),
    "OsmoMSC": ("MSC/VLR", "coeur"),
    "OsmoHLR": ("HLR", "abonnes"),
    "OsmoBSC": ("BSC", "radio"),
    "OsmoBTS": ("BTS", "radio"),
    "OsmoMGW": ("MGW", "media"),
    "OsmoPCU": ("PCU", "data"),
    "OsmoSGSN": ("SGSN", "data"),
    "OsmoGGSN": ("GGSN", "data"),
    "OsmoUPF": ("UPF", "data"),
    "OsmoTRX": ("TRX", "radio"),
    "OsmoSIPConnector": ("SIP-CONN", "voix"),
    "OsmoMobile": ("MS", "mobile"),
    "mobile": ("MS", "mobile"),
}

# Repli quand la banniere ne dit rien (VTY muet, demon en cours de demarrage).
_BY_PORT = {
    4238: "OsmoHNBGW", 4239: "OsmoSTP", 4240: "OsmoPCU", 4241: "OsmoBTS",
    4242: "OsmoBSC", 4243: "OsmoMGW", 4245: "OsmoSGSN", 4246: "OsmoGbProxy",
    4247: "OsmoMobile", 4248: "OsmoMobile", 4249: "OsmoSIPConnector",
    4250: "OsmoBTS", 4251: "OsmoMSC", 4254: "OsmoMSC", 4255: "OsmoMGW",
    4256: "OsmoUECUPS", 4257: "OsmoE1D", 4258: "OsmoHLR", 4259: "OsmoUPF",
    4260: "OsmoTRX", 4267: "OsmoGGSN",
}


def sh(args, timeout=20):
    try:
        out = subprocess.run(args, stdout=subprocess.PIPE,
                             stderr=subprocess.DEVNULL, timeout=timeout)
        return out.stdout.decode("utf-8", "replace")
    except Exception:
        return ""


class Service(object):
    """Un demon joignable par VTY sur un noeud."""

    def __init__(self, node, port, daemon, label, family):
        self.node = node
        self.port = int(port)
        self.daemon = daemon
        self.label = label
        self.family = family

    def __repr__(self):
        return "<Service %s:%s %s>" % (self.node, self.port, self.label)
```

```
class Node(object):
    """Un noeud du lab : conteneur operateur, ou hub inter-STP."""

    def __init__(self, name, role, index=None):
        self.name = name            # nom du conteneur, ou "local"
        self.role = role            # "operator" | "interstp"
        self.index = index          # numero d'operateur
        self.point_code = ""        # 1.11.2
        self.hub_ip = ""            # adresse de l'inter-STP vu par ce noeud
        self.hub_port = 2908
        self.local_m3ua = 2905
        self.routing_ctx = None     # routing key vers le hub
        self.wan_node = ""
        self.services = []          # [Service]
        self.sccp = {}              # nom -> {"pc":..., "ssn":..., "ri":...}
        self.env = {}
        self.cfg = {}               # fichier -> texte

    @property
    def title(self):
        if self.role == "interstp":
            return "INTER-STP"
        return "OP%s" % (self.index if self.index is not None else "?")

    def service(self, daemon):
        for s in self.services:
            if s.daemon == daemon:
                return s
        return None

    def __repr__(self):
        return "<Node %s pc=%s svc=%d>" % (self.name, self.point_code, len(self.services))


class Topology(object):
    def __init__(self):
        self.docker = True
        self.nodes = []
        self.hub_ip = ""
        self.hub_port = 2908
        self.hub_local = False
        self.repo = REPO
        self.errors = []

    @property
    def operators(self):
        return [n for n in self.nodes if n.role == "operator"]

    @property
    def hub(self):
        for n in self.nodes:
            if n.role == "interstp":
                return n
        return None

    def node(self, name):
        for n in self.nodes:
            if n.name == name:
                return n
        return None

    def free_point_codes(self, count=1):
        """Point codes libres pour la console, au FORMAT DU LAB (ITU 3-8-3).

        Le troisieme membre tient sur 3 bits : 0 a 7. Un "1.11.60" deborderait
        et serait recu comme un autre point code (1.15.4) - le message partirait
```

```
        vers un voisin, ou nulle part. On choisit donc dans les valeurs libres
        du meme groupe que les STP operateur (qui utilisent .1 a .3)."""
        used = set(n.point_code for n in self.nodes if n.point_code)
        for node in self.nodes:
            for addr in node.sccp.values():
                if addr.get("pc"):
                    used.add(addr["pc"])
        base = None
        for n in self.operators:
            if n.point_code:
                base = n.point_code
                break
        out = []
        heads = []
        if base and base.count(".") == 2:
            a, b, _ = base.split(".")
            heads.append((int(a), int(b)))
        heads.append((1, 11))
        for a, b in heads:
            for last in range(7, 0, -1):
                cand = "%d.%d.%d" % (a, b, last)
                if cand not in used and cand not in out:
                    out.append(cand)
                    if len(out) >= count:
                        return out
        while len(out) < count:
            out.append("1.11.7")
        return out

    def free_point_code(self):
        return self.free_point_codes(1)[0]


# ── lecture des configurations dans un noeud ─────────────────────────

_CFG_FILES = ["osmo-stp.cfg", "osmo-msc.cfg", "osmo-bsc.cfg", "osmo-hlr.cfg",
              "osmo-sgsn.cfg", "osmo-ggsn.cfg", "osmo-mgw.cfg"]


def _read_cfgs(name, docker):
    cfg = {}
    for f in _CFG_FILES:
        path = "/etc/osmocom/" + f
        if docker:
            txt = sh(["docker", "exec", name, "cat", path], timeout=10)
        else:
            try:
                txt = open(path).read()
            except Exception:
                txt = ""
        if txt.strip():
            cfg[f] = txt
    return cfg


def _parse_stp(node):
    txt = node.cfg.get("osmo-stp.cfg", "")
    if not txt:
        return
    m = re.search(r"^\s*point-code\s+(\S+)", txt, re.M)
    if m:
        node.point_code = m.group(1)
    m = re.search(r"^\s*listen\s+m3ua\s+(\d+)", txt, re.M)
    if m:
        node.local_m3ua = int(m.group(1))
    m = re.search(r"^\s*asp\s+asp-to-inter\s+(\d+)\s+(\d+)", txt, re.M)
```

```
    if m:
        node.hub_port = int(m.group(1))
    m = re.search(r"asp\s+asp-to-inter.*?^\s*remote-ip\s+(\S+)", txt, re.M | re.S)
    if m:
        node.hub_ip = m.group(1)
    m = re.search(r"as\s+as-inter.*?^\s*routing-key\s+(\d+)\s+(\S+)", txt, re.M | re.S)
    if m:
        node.routing_ctx = int(m.group(1))


def _parse_sccp(node):
    """Adresses SCCP declarees (osmo-bsc.cfg / osmo-msc.cfg) : c'est par la
    qu'on connait les SSN et les point codes des paires A-interface."""
    for fname in ("osmo-bsc.cfg", "osmo-msc.cfg"):
        txt = node.cfg.get(fname, "")
        if not txt:
            continue
        for blk in re.finditer(
                r"^\s*sccp-address\s+(\S+)\s*\n((?:\s+\S.*\n)*)", txt, re.M):
            name, body = blk.group(1), blk.group(2)
            entry = {"ssn": None, "pc": None, "ri": None, "source": fname}
            m = re.search(r"^\s*point-code\s+(\S+)", body, re.M)
            if m:
                entry["pc"] = m.group(1)
            m = re.search(r"^\s*subsystem-number\s+(\d+)", body, re.M)
            if m:
                entry["ssn"] = int(m.group(1))
            m = re.search(r"^\s*routing-indicator\s+(\S+)", body, re.M)
            if m:
                entry["ri"] = m.group(1)
            node.sccp[name] = entry
        # cs7 instance N / point-code du demon lui-meme
        m = re.search(r"^\s*point-code\s+(\S+)", txt, re.M)
        if m:
            node.sccp.setdefault("_self_" + fname, {"pc": m.group(1), "ssn": None,
                                                    "ri": None, "source": fname})


def _probe_vty(node, docker):
    if docker:
        raw = sh(["docker", "exec", "-i", node.name, "python3", "-c", _PROBE], timeout=30)
    else:
        raw = sh(["python3", "-c", _PROBE], timeout=30)
    try:
        found = json.loads(raw.strip().splitlines()[-1]) if raw.strip() else {}
    except Exception:
        found = {}
    for port, banner in sorted((int(p), b) for p, b in found.items()):
        daemon = ""
        m = re.search(r"Welcome to the (\S+) VTY interface", banner)
        if m:
            daemon = m.group(1)
        if not daemon:
            daemon = _BY_PORT.get(port, "VTY")
        label, family = _PRETTY.get(daemon, (daemon.replace("Osmo", ""), "autre"))
        node.services.append(Service(node.name, port, daemon, label, family))


def _env(name):
    raw = sh(["docker", "inspect", "-f", "{{json .Config.Env}}", name], timeout=10)
    env = {}
    try:
        for item in json.loads(raw or "[]"):
            if "=" in item:
                k, v = item.split("=", 1)
                env[k] = v
```

```
    except Exception:
        pass
    return env


def _native_present():
    """Cette machine porte-t-elle une pile Osmocom native ?"""
    if os.path.isdir("/etc/osmocom") and os.listdir("/etc/osmocom"):
        return True
    return bool(sh(["pgrep", "-f", "osmo-stp"]).strip())


def discover(docker=None, probe=True):
    topo = Topology()
    if docker is None:
        docker = bool(sh(["docker", "ps", "--format", "{{.Names}}"]).strip())
    topo.docker = docker

    names = []
    if docker:
        names = [n for n in sh(["docker", "ps", "--format", "{{.Names}}"]).split()
                 if n.startswith("osmo-")]
    if docker and not names:
        topo.errors.append("aucun conteneur osmo-* en cours d'execution")

    for name in sorted(names):
        if name == "osmo-inter-stp":
            node = Node(name, "interstp")
            topo.hub_local = True
        else:
            m = re.match(r"osmo-operator-(\d+)$", name)
            if not m:
                continue
            node = Node(name, "operator", int(m.group(1)))
        node.env = _env(name)
        node.cfg = _read_cfgs(name, True)
        _parse_stp(node)
        _parse_sccp(node)
        if probe:
            _probe_vty(node, True)
        node.wan_node = node.env.get("OSMO_WAN_NODE", node.env.get("WAN_NODE_ID", ""))
        if not node.hub_ip:
            node.hub_ip = node.env.get("OSMO_HUB_IP", "")
        topo.nodes.append(node)

    # Repli natif : SEULEMENT si cette machine porte vraiment une pile Osmocom.
    # Sans cette verification, un lab docker eteint se presentait comme un
    # "operateur natif" sans point code ni service - un inventaire imaginaire,
    # plus trompeur qu'un franc "il n'y a rien ici".
    if not docker and not _native_present():
        topo.errors.append(
            "aucun noeud : ni conteneur osmo-* en cours, ni /etc/osmocom sur "
            "cette machine (le lab est arrete ?)")
    elif not docker:
        node = Node("local", "operator", 1)
        node.cfg = _read_cfgs("local", False)
        _parse_stp(node)
        _parse_sccp(node)
        if probe:
            _probe_vty(node, False)
        # Un repertoire /etc/osmocom peut n'etre qu'un reste d'installation :
        # sans point code lisible ni VTY qui repond, il n'y a pas de noeud.
        if node.point_code or node.services:
            topo.nodes.append(node)
        else:
            topo.errors.append(
```

```
                "/etc/osmocom existe mais aucun demon Osmocom ne repond ici "
                "(ni point code, ni VTY) : rien a inventorier")

    for n in topo.operators:
        if n.hub_ip:
            topo.hub_ip = n.hub_ip
            topo.hub_port = n.hub_port
            break
    if topo.hub_local and not topo.hub_ip:
        topo.hub_ip = "172.20.0.10"

    hub = topo.hub
    if hub is None and topo.hub_ip:
        # Hub distant (WAN) : il ne tourne pas ici, mais il fait partie du
        # schema - c'est par lui que passe tout le SS7 inter-operateur.
        hub = Node("hub-distant", "interstp")
        hub.point_code = "0.0.0"
        hub.hub_ip = topo.hub_ip
        hub.hub_port = topo.hub_port
        topo.nodes.append(hub)
    elif hub is not None and not hub.point_code:
        hub.point_code = "0.0.0"
    return topo
```

3.243 /opt/GSM/osmo-operator/navigation/tui.py

40179 octets, 916 lignes → 916 lignes

```
# tui.py - la console : un schema du reseau ou l'on circule aux fleches, et
# d'ou l'on ouvre soit un VTY, soit un script SS7.
#
# Principes de navigation, tenus partout :
#   fleches      se deplacer (dans le schema : de boite en boite, dans l'espace)
#   Entree       ouvrir / valider
#   Echap        revenir en arriere ; DANS UN VTY, Echap ferme la session et
#                rend la main au schema - les autres VTY restent disponibles
#   q            quitter (depuis le schema)
#
# Le reste (v, s, a, c, r, ?) est du raccourci : tout est aussi atteignable a
# la fleche depuis le menu d'une boite.

import curses
import locale
import os
import time

from . import diagram
from . import mapops
from . import quickcmd
from . import sccp
from . import ss7
from . import topo as topo_mod
from . import vty as vty_mod

locale.setlocale(locale.LC_ALL, "")

C_DEFAULT, C_HUB, C_STP, C_CORE, C_RADIO, C_DATA, C_CONSOLE, C_BAR, C_WARN, C_OK = range(1, 11)

FAMILY_COLOR = {
    "SS7": C_STP, "coeur": C_CORE, "abonnes": C_CORE, "radio": C_RADIO,
    "media": C_DATA, "data": C_DATA, "voix": C_CORE, "mobile": C_RADIO,
}

HELP = [
```

```
    "NAVIGATION",
    "  fleches            se deplacer dans le schema (haut/bas/gauche/droite)",
    "  Entree             ouvrir le menu de l'element selectionne",
    "  Echap              revenir en arriere ; dans un VTY : fermer la session",
    "  Tab                element suivant, Maj-Tab precedent",
    "",
    "RACCOURCIS",
    "  v                  ouvrir directement le VTY de l'element",
    "  s                  scripts SS7 (operations MAP) sur l'element",
    "  a                  audit SS7 (DAUD) d'un point code",
    "  c                  connecter / deconnecter le lien M3UA de la console",
    "  j                  journal du lien M3UA",
    "  r                  relire la topologie (redecouverte complete)",
    "  ?                  cette aide            q  quitter",
    "",
    "DANS UN VTY",
    "  Entree             envoyer la commande",
    "  fleches haut/bas   historique des commandes",
    "  PagePrec/PageSuiv  faire defiler la sortie",
    "  Echap              fermer CE VTY et revenir au schema",
    "",
    "SCRIPTS SS7",
    "  La console s'enregistre sur l'inter-STP comme un ASP M3UA a part",
    "  entiere (RKM dynamique), avec son propre point code. Elle emet du",
    "  TCAP/MAP reel : sendRoutingInfoForSM, sendRoutingInfo, ATI, SAI...",
    "  Chaque envoi peut etre exporte en script Python autonome.",
]


# -- lecture des touches ------------------------------------------------------
# ncurses ne reconnait les fleches que dans la forme prevue par le terminfo du
# terminal (ici "\x1bOB", mode curseur applicatif). Un multiplexeur, un SSH mal
# configure ou un terminal qui reste en mode normal envoient "\x1b[B" : ncurses
# rend alors trois touches brutes et la navigation semble morte. On decode donc
# nous-memes les deux formes. Echap seul reste Echap : c'est l'absence de suite
# dans les 40 ms qui le distingue d'un debut de sequence.
_CSI_FINAL = {
    "A": curses.KEY_UP, "B": curses.KEY_DOWN, "C": curses.KEY_RIGHT,
    "D": curses.KEY_LEFT, "H": curses.KEY_HOME, "F": curses.KEY_END,
    "Z": curses.KEY_BTAB, "P": curses.KEY_F1, "Q": curses.KEY_F2,
    "R": curses.KEY_F3, "S": curses.KEY_F4,
}
_CSI_TILDE = {1: curses.KEY_HOME, 2: curses.KEY_IC, 3: curses.KEY_DC,
              4: curses.KEY_END, 5: curses.KEY_PPAGE, 6: curses.KEY_NPAGE,
              15: curses.KEY_F5, 17: curses.KEY_F6, 18: curses.KEY_F7,
              19: curses.KEY_F8, 20: curses.KEY_F9, 21: curses.KEY_F10}


def read_key(win):
    key = win.getch()
    if key != 27:
        return key
    win.timeout(40)
    try:
        nxt = win.getch()
        if nxt == -1:
            return 27
        if nxt not in (ord("["), ord("O")):
            return 27
        params = ""
        for _ in range(8):
            ch = win.getch()
            if ch == -1:
                return 27
            c = chr(ch)
            if c.isdigit() or c == ";":
```

```
                params += c
                continue
            if c == "~":
                try:
                    return _CSI_TILDE.get(int(params.split(";")[0]), 27)
                except ValueError:
                    return 27
            return _CSI_FINAL.get(c, 27)
        return 27
    finally:
        win.timeout(-1)


def init_colors():
    curses.start_color()
    curses.use_default_colors()
    pairs = {
        C_DEFAULT: (curses.COLOR_WHITE, -1),
        C_HUB: (curses.COLOR_MAGENTA, -1),
        C_STP: (curses.COLOR_CYAN, -1),
        C_CORE: (curses.COLOR_GREEN, -1),
        C_RADIO: (curses.COLOR_YELLOW, -1),
        C_DATA: (curses.COLOR_BLUE, -1),
        C_CONSOLE: (curses.COLOR_WHITE, -1),
        C_BAR: (curses.COLOR_BLACK, curses.COLOR_CYAN),
        C_WARN: (curses.COLOR_RED, -1),
        C_OK: (curses.COLOR_GREEN, -1),
    }
    for idx, (fg, bg) in pairs.items():
        try:
            curses.init_pair(idx, fg, bg)
        except curses.error:
            pass


class App(object):
    def __init__(self, stdscr, topo):
        self.scr = stdscr
        self.topo = topo
        self.pool = vty_mod.VtyPool(docker=topo.docker)
        self.client = None
        self.local_pc = topo.free_point_code()
        self.local_ssn = mapops.SSN_MSC
        self.sel = 0
        self.top = 0
        self.left = 0
        self.status = "Bienvenue. Fleches pour circuler, Entree pour ouvrir, ? pour l'aide."
        self.status_attr = C_DEFAULT
        self.rebuild()

    # -- etat ---------------------------------------------------------
    def rebuild(self):
        extra = {"local_pc": self.local_pc, "local_ssn": self.local_ssn,
                 "state": (self.client.asp.state if self.client and self.client.asp
                           else "hors ligne")}
        self.lines, self.boxes = diagram.build(self.topo, extra)
        self.sel = min(self.sel, max(0, len(self.boxes) - 1))

    def say(self, text, attr=C_DEFAULT):
        self.status = text
        self.status_attr = attr

    # -- dessin -------------------------------------------------------
    def draw(self):
        self.scr.erase()
        h, w = self.scr.getmaxyx()
```

```
    title = " CONSOLE SS7 - osmo-operator "
    link = "M3UA %s" % (self.client.asp.state if self.client and self.client.asp
                        else "hors ligne")
    head = "%s| %s operateur(s) | hub %s:%s | console PC %s | %s" % (
        title, len(self.topo.operators), self.topo.hub_ip or "?",
        self.topo.hub_port, self.local_pc, link)
    self.scr.attron(curses.color_pair(C_BAR))
    self.scr.addstr(0, 0, head.ljust(w - 1)[:w - 1])
    self.scr.attroff(curses.color_pair(C_BAR))

    view_h = h - 3
    self.draw_map(1, view_h, w)

    foot = ("fleches:circuler  Entree:ouvrir  v:VTY  s:script SS7  "
            "a:audit  c:lien M3UA  r:relire  ?:aide  q:quitter")
    self.scr.attron(curses.color_pair(C_BAR))
    self.scr.addstr(h - 1, 0, foot.ljust(w - 1)[:w - 1])
    self.scr.attroff(curses.color_pair(C_BAR))
    self.scr.attron(curses.color_pair(self.status_attr))
    self.scr.addstr(h - 2, 0, self.status.ljust(w - 1)[:w - 1])
    self.scr.attroff(curses.color_pair(self.status_attr))
    self.scr.noutrefresh()
    curses.doupdate()

def _scroll_to_selection(self, view_h, view_w):
    if not self.boxes:
        return
    b = self.boxes[self.sel]
    if b.y < self.top:
        self.top = max(0, b.y - 1)
    if b.y + b.h > self.top + view_h:
        self.top = b.y + b.h - view_h
    if b.x < self.left:
        self.left = max(0, b.x - 2)
    if b.x + b.w > self.left + view_w:
        self.left = b.x + b.w - view_w + 2
    self.top = max(0, self.top)
    self.left = max(0, self.left)

def draw_map(self, y0, view_h, view_w):
    self._scroll_to_selection(view_h, view_w)
    for row in range(view_h):
        src = self.top + row
        if src >= len(self.lines):
            break
        text = self.lines[src][self.left:self.left + view_w - 1]
        try:
            self.scr.addstr(y0 + row, 0, text)
        except curses.error:
            pass
    for i, b in enumerate(self.boxes):
        attr = curses.color_pair(self._box_color(b))
        if i == self.sel:
            attr = curses.color_pair(self._box_color(b)) | curses.A_REVERSE | curses.A_BOLD
        for row in range(b.h):
            src = b.y + row
            if src < self.top or src >= self.top + view_h:
                continue
            if src >= len(self.lines):
                continue
            line = self.lines[src]
            seg = line[b.x:b.x + b.w].ljust(b.w)
            sx = b.x - self.left
            if sx < 0 or sx + b.w > view_w - 1:
                seg = seg[max(0, -sx):max(0, view_w - 1 - max(sx, 0))]
                sx = max(sx, 0)
```

```
            try:
                self.scr.addstr(y0 + src - self.top, sx, seg, attr)
            except curses.error:
                pass

def _box_color(self, b):
    if b.kind == "hub":
        return C_HUB
    if b.kind == "console":
        return C_CONSOLE
    if b.kind == "stp":
        return C_STP
    if b.ref is not None and getattr(b.ref, "family", None):
        return FAMILY_COLOR.get(b.ref.family, C_DEFAULT)
    return C_DEFAULT

# ── fenetres generiques ──────────────────────────────────────
def _win(self, height, width, title):
    h, w = self.scr.getmaxyx()
    height = min(height, h - 2)
    width = min(width, w - 2)
    y = max(0, (h - height) // 2)
    x = max(0, (w - width) // 2)
    win = curses.newwin(height, width, y, x)
    win.keypad(True)
    win.box()
    win.addstr(0, 2, " %s " % title[:width - 6], curses.A_BOLD)
    return win

def menu(self, title, items, footer="fleches + Entree, Echap pour revenir"):
    """items : liste de (etiquette, valeur). Rend la valeur, ou None."""
    if not items:
        self.show_text(title, ["(rien a proposer ici)"])
        return None
    idx = 0
    width = max([len(title) + 8] + [len(i[0]) + 8 for i in items] + [len(footer) + 6])
    win = self._win(len(items) + 4, min(width, 100), title)
    while True:
        win.erase()
        win.box()
        win.addstr(0, 2, " %s " % title, curses.A_BOLD)
        for i, (label, _val) in enumerate(items):
            attr = curses.A_REVERSE if i == idx else curses.A_NORMAL
            try:
                win.addstr(1 + i, 2, ("%-*s" % (win.getmaxyx()[1] - 4, label))[
                    :win.getmaxyx()[1] - 4], attr)
            except curses.error:
                pass
        try:
            win.addstr(win.getmaxyx()[0] - 2, 2, footer[:win.getmaxyx()[1] - 4],
                       curses.A_DIM)
        except curses.error:
            pass
        win.refresh()
        key = read_key(win)
        if key in (curses.KEY_UP, ord("k")):
            idx = (idx - 1) % len(items)
        elif key in (curses.KEY_DOWN, ord("j")):
            idx = (idx + 1) % len(items)
        elif key in (curses.KEY_HOME,):
            idx = 0
        elif key in (curses.KEY_END,):
            idx = len(items) - 1
        elif key in (10, 13, curses.KEY_ENTER, curses.KEY_RIGHT):
            return items[idx][1]
        elif key in (27, curses.KEY_LEFT, ord("q")):
```

```
            return None
        elif key == curses.KEY_RESIZE:
            self.draw()

def show_text(self, title, lines, wrap=True):
    """Zone de texte defilante. Echap ou q pour revenir."""
    h, w = self.scr.getmaxyx()
    win = self._win(h - 2, w - 2, title)
    inner_h = win.getmaxyx()[0] - 3
    inner_w = win.getmaxyx()[1] - 4
    flat = []
    for line in lines:
        line = line.replace("\t", "    ")
        if wrap:
            while len(line) > inner_w:
                flat.append(line[:inner_w])
                line = line[inner_w:]
        flat.append(line[:inner_w])
    pos = 0
    while True:
        win.erase()
        win.box()
        win.addstr(0, 2, " %s " % title[:inner_w], curses.A_BOLD)
        for i in range(inner_h):
            if pos + i >= len(flat):
                break
            try:
                win.addstr(1 + i, 2, flat[pos + i])
            except curses.error:
                pass
        hint = "fleches/PagePrec/PageSuiv pour defiler - Echap pour revenir"
        if len(flat) > inner_h:
            hint = "%d/%d - %s" % (pos + 1, len(flat), hint)
        try:
            win.addstr(win.getmaxyx()[0] - 2, 2, hint[:inner_w], curses.A_DIM)
        except curses.error:
            pass
        win.refresh()
        key = read_key(win)
        if key in (27, ord("q")):
            return
        elif key == curses.KEY_DOWN:
            pos = min(max(0, len(flat) - inner_h), pos + 1)
        elif key == curses.KEY_UP:
            pos = max(0, pos - 1)
        elif key == curses.KEY_NPAGE:
            pos = min(max(0, len(flat) - inner_h), pos + inner_h)
        elif key == curses.KEY_PPAGE:
            pos = max(0, pos - inner_h)
        elif key == curses.KEY_HOME:
            pos = 0
        elif key == curses.KEY_END:
            pos = max(0, len(flat) - inner_h)
        elif key == curses.KEY_RESIZE:
            self.draw()
            return

def form(self, title, fields, note=""):
    """fields : liste de dicts {key,label,value,help}. Rend un dict ou None.

    Navigation : haut/bas entre les champs, frappe pour editer, Entree sur
    [ Envoyer ] pour valider, Echap pour abandonner."""
    idx = 0
    rows = fields + [{"key": "__submit__", "label": "[ Envoyer ]", "value": "",
                      "help": "Entree pour lancer"}]
    height = len(rows) + (6 if note else 5)
```

```
    width = 78
    win = self._win(height, width, title)
    inner_w = win.getmaxyx()[1] - 4
    while True:
        win.erase()
        win.box()
        win.addstr(0, 2, " %s " % title, curses.A_BOLD)
        y = 1
        if note:
            win.addstr(y, 2, note[:inner_w], curses.A_DIM)
            y += 1
        for i, f in enumerate(rows):
            attr = curses.A_REVERSE if i == idx else curses.A_NORMAL
            if f["key"] == "__submit__":
                win.addstr(y + i, 2, ("%-*s" % (inner_w, f["label"]))[:inner_w],
                           attr | curses.A_BOLD)
            else:
                label = "%-26s" % f["label"][:26]
                val = str(f.get("value", ""))
                win.addstr(y + i, 2, label, curses.A_NORMAL)
                win.addstr(y + i, 2 + len(label),
                           ("%-*s" % (inner_w - len(label), val))[:inner_w - len(label)],
                           attr)
        cur = rows[idx]
        hint = cur.get("help") or "haut/bas : champ suivant - Echap : abandonner"
        try:
            win.addstr(win.getmaxyx()[0] - 2, 2, hint[:inner_w], curses.A_DIM)
        except curses.error:
            pass
        win.refresh()
        key = read_key(win)
        if key == 27:
            return None
        if key in (curses.KEY_UP,):
            idx = (idx - 1) % len(rows)
        elif key in (curses.KEY_DOWN, 9):
            idx = (idx + 1) % len(rows)
        elif key in (10, 13, curses.KEY_ENTER):
            if rows[idx]["key"] == "__submit__":
                return dict((f["key"], f.get("value", "")) for f in fields)
            idx = (idx + 1) % len(rows)
        elif key in (curses.KEY_BACKSPACE, 127, 8):
            if cur["key"] != "__submit__":
                cur["value"] = str(cur.get("value", ""))[:-1]
        elif key == 21:                     # Ctrl-U
            if cur["key"] != "__submit__":
                cur["value"] = ""
        elif key == curses.KEY_RESIZE:
            self.draw()
        elif 32 <= key < 127 and cur["key"] != "__submit__":
            cur["value"] = str(cur.get("value", "")) + chr(key)

def confirm(self, question):
    return self.menu(question, [("Oui", True), ("Non", False)]) is True

# ── VTY ──────────────────────────────────────────────────────────
def vty_view(self, node, port, label):
    try:
        sess = self.pool.get(node, port)
    except vty_mod.VtyError as exc:
        self.say("VTY %s:%s - %s" % (node, port, exc), C_WARN)
        return
    lines = [l for l in (sess.banner or "").split("\n") if l.strip()]
    lines.append("")
    lines.append("--- Echap ferme cette session et revient au schema ---")
    history = []
```

```
hpos = 0
current = ""
scroll = None
h, w = self.scr.getmaxyx()
win = curses.newwin(h, w, 0, 0)
win.keypad(True)
while True:
    win.erase()
    head = " VTY %s - %s (port %d) " % (label, node, port)
    win.attron(curses.color_pair(C_BAR))
    win.addstr(0, 0, head.ljust(w - 1)[:w - 1])
    win.attroff(curses.color_pair(C_BAR))
    body_h = h - 3
    view = lines if scroll is None else lines[:scroll + body_h]
    start = max(0, len(view) - body_h)
    for i, line in enumerate(view[start:start + body_h]):
        try:
            win.addstr(1 + i, 0, line[:w - 1])
        except curses.error:
            pass
    foot = ("Entree:envoyer  haut/bas:historique  PagePrec/Suiv:defiler  "
            "Echap:fermer ce VTY")
    win.attron(curses.color_pair(C_BAR))
    win.addstr(h - 2, 0, foot.ljust(w - 1)[:w - 1])
    win.attroff(curses.color_pair(C_BAR))
    prompt = "%s> " % label
    win.addstr(h - 1, 0, (prompt + current)[:w - 1], curses.A_BOLD)
    win.move(h - 1, min(w - 1, len(prompt) + len(current)))
    win.refresh()
    key = read_key(win)
    if key == 27:
        self.pool.drop(node, port)
        self.say("VTY %s:%s ferme - les autres VTY restent ouverts."
                 % (node, port), C_OK)
        return
    if key in (10, 13, curses.KEY_ENTER):
        cmd = current.strip()
        current = ""
        if not cmd:
            continue
        if cmd in ("quit", "exit"):
            self.pool.drop(node, port)
            return
        history.append(cmd)
        hpos = len(history)
        lines.append("")
        lines.append("%s> %s" % (label, cmd))
        try:
            out = sess.cmd(cmd)
        except vty_mod.VtyError as exc:
            out = "[session perdue : %s]" % exc
        rows = out.split("\n")
        # Le VTY renvoie la commande en echo : elle est deja affichee
        # au-dessus, on ne la montre pas deux fois.
        if rows and rows[0].strip() == cmd:
            rows = rows[1:]
        lines.extend(rows)
        scroll = None
    elif key in (curses.KEY_BACKSPACE, 127, 8):
        current = current[:-1]
    elif key == 21:
        current = ""
    elif key == curses.KEY_UP:
        if history:
            hpos = max(0, hpos - 1)
            current = history[hpos]
```

```
        elif key == curses.KEY_DOWN:
            if history:
                hpos = min(len(history), hpos + 1)
                current = history[hpos] if hpos < len(history) else ""
        elif key == curses.KEY_PPAGE:
            base = len(lines) if scroll is None else scroll
            scroll = max(0, base - (h - 3))
        elif key == curses.KEY_NPAGE:
            if scroll is not None:
                scroll += h - 3
                if scroll >= len(lines):
                    scroll = None
        elif key == curses.KEY_RESIZE:
            h, w = self.scr.getmaxyx()
            win = curses.newwin(h, w, 0, 0)
            win.keypad(True)
        elif 32 <= key < 127:
            current += chr(key)

# ── lien SS7 ──────────────────────────────────────────────────────
def ensure_link(self):
    if self.client and self.client.connected:
        return True
    if not self.topo.hub_ip:
        self.say("aucun hub SS7 connu : pas de lien M3UA possible", C_WARN)
        return False
    self.say("connexion M3UA vers %s:%s ..." % (self.topo.hub_ip, self.topo.hub_port))
    self.draw()
    try:
        self.client = ss7.Ss7Client(self.topo.hub_ip, self.topo.hub_port,
                                    self.local_pc, self.local_ssn)
        state = self.client.connect()
    except Exception as exc:
        self.say("lien M3UA impossible : %s" % exc, C_WARN)
        self.client = None
        return False
    self.rebuild()
    self.say("lien M3UA %s - la console est un ASP du hub (PC %s)"
             % (state, self.local_pc), C_OK)
    return True

def toggle_link(self):
    if self.client and self.client.connected:
        self.client.close()
        self.client = None
        self.rebuild()
        self.say("lien M3UA ferme", C_WARN)
    else:
        self.ensure_link()

def audit_dialog(self, default_pc=""):
    if not self.ensure_link():
        return
    vals = self.form("Audit SS7 (DAUD)", [
        {"key": "pc", "label": "Point code a auditer", "value": default_pc,
         "help": "ex 1.11.1 - le STP repond DAVA (joignable) ou DUNA"},
    ])
    if not vals or not vals["pc"].strip():
        return
    rep = self.client.audit(vals["pc"].strip())
    self.show_text("Audit %s" % vals["pc"], rep.lines + [""] +
                   ["Journal du lien :"] + self.client.events[-12:])

def map_dialog(self, default_pc="", default_ssn=None):
    if not self.ensure_link():
        return
```

```
    items = [("%-22s %s" % (o.key, o.name), o.key) for o in mapops.OPERATIONS]
    key = self.menu("Operations MAP - schema %s" % self.local_pc, items)
    if key is None:
        return
    op = mapops.BY_KEY[key]
    fields = []
    for p in op.params:
        fields.append({"key": p.key, "label": p.label, "value": p.default,
                       "help": p.help})
    fields.append({"key": "__pc", "label": "Point code destination",
                   "value": default_pc or (self.topo.operators[0].sccp.get(
                       "addr-msc", {}).get("pc", "") if self.topo.operators else ""),
                   "help": "ou vide si routage sur GT"})
    fields.append({"key": "__ssn", "label": "SSN destination",
                   "value": str(default_ssn if default_ssn else op.ssn),
                   "help": "6 HLR, 7 VLR, 8 MSC, 147 gsmSCF, 254 BSSAP"})
    fields.append({"key": "__gt", "label": "Global title destination",
                   "value": "", "help": "laisser vide pour router sur PC/SSN"})
    vals = self.form("%s (opcode %d)" % (op.name, op.opcode), fields,
                     note=op.summary[:74])
    if vals is None:
        return
    dest_pc = vals.pop("__pc").strip()
    dest_ssn = vals.pop("__ssn").strip()
    dest_gt = vals.pop("__gt").strip()
    if not dest_pc and not dest_gt:
        self.say("il faut au moins un point code ou un global title", C_WARN)
        return
    self.say("emission de %s ..." % op.name)
    self.draw()
    try:
        rep = self.client.send_map(op, vals, dest_pc or "0.0.0",
                                   int(dest_ssn) if dest_ssn else None,
                                   dest_gt, route_on_gt=bool(dest_gt and not dest_pc))
    except Exception as exc:
        self.show_text("Echec", ["%s : %s" % (type(exc).__name__, exc)])
        return
    lines = rep.lines + ["", "Journal du lien :"] + self.client.events[-10:]
    lines += ["", "e : exporter cet envoi en script Python autonome",
              "Echap : revenir"]
    self.show_text("%s -> %s" % (op.name, dest_gt or dest_pc), lines)
    if self.confirm("Exporter cet envoi en script Python ?"):
        path = export_script(op, vals, dest_pc, dest_ssn, dest_gt,
                             self.local_pc, self.topo)
        self.show_text("Script ecrit", [
            "Le script autonome est ecrit dans :", "", "  %s" % path, "",
            "Il rejoue exactement cet envoi :",
            "  python3 %s" % os.path.relpath(path, topo_mod.REPO)])

# ── menus des boites ──────────────────────────────────────────────
def open_box(self, box):
    if box.kind == "console":
        self.console_menu()
    elif box.kind == "hub":
        self.hub_menu(box)
    elif box.kind == "stp":
        self.stp_menu(box)
    elif box.kind == "service":
        self.service_menu(box)

def console_menu(self):
    connected = bool(self.client and self.client.connected)
    items = [
        ("Deconnecter le lien M3UA" if connected else "Connecter le lien M3UA",
         "link"),
        ("Operations MAP (sendRoutingInfo, SRI-SM, ATI...)", "map"),
```

```
        ("Audit d'un point code (DAUD)", "audit"),
        ("Journal du lien M3UA", "log"),
        ("Changer le point code de la console", "pc"),
        ("Informations", "info"),
    ]
    action = self.menu("CONSOLE SS7 (PC %s)" % self.local_pc, items)
    if action == "link":
        self.toggle_link()
    elif action == "map":
        self.map_dialog()
    elif action == "audit":
        self.audit_dialog()
    elif action == "log":
        self.show_text("Journal M3UA",
                       self.client.events if self.client else ["(pas de lien)"])
    elif action == "pc":
        vals = self.form("Point code de la console", [
            {"key": "pc", "label": "Point code", "value": self.local_pc,
             "help": "il sera enregistre sur le hub par RKM dynamique"},
            {"key": "ssn", "label": "SSN local", "value": str(self.local_ssn),
             "help": "identite SCCP de la console"}])
        if vals:
            self.local_pc = vals["pc"].strip() or self.local_pc
            try:
                self.local_ssn = int(vals["ssn"])
            except ValueError:
                pass
            if self.client:
                self.client.close()
                self.client = None
            self.rebuild()
            self.say("console : PC %s, SSN %d" % (self.local_pc, self.local_ssn))
    elif action == "info":
        self.show_text("Console SS7", [
            "La console se connecte a l'inter-STP en M3UA sur SCTP et",
            "enregistre dynamiquement une routing key pour son point code.",
            "",
            "  hub          : %s:%s" % (self.topo.hub_ip, self.topo.hub_port),
            "  point code   : %s" % self.local_pc,
            "  SSN local    : %s" % self.local_ssn,
            "  etat         : %s" % (self.client.asp.state if self.client
                                     and self.client.asp else "hors ligne"),
            "",
            "Tout ce qu'elle emet passe par le meme chemin SS7 que le trafic",
            "des operateurs : c'est le lab qui route, pas une simulation.",
        ])

def hub_menu(self, box):
    node = box.ref
    items = [("Etat du lien M3UA de la console", "link"),
             ("Auditer un point code a travers le hub", "audit"),
             ("Journal M3UA", "log")]
    if node and node.name != "hub-distant":
        items.insert(0, ("Ouvrir le VTY du hub (STP)", "vty"))
    else:
        items.append(("Pourquoi le hub n'a pas de VTY ici", "why"))
    action = self.menu("INTER-STP %s" % (self.topo.hub_ip or ""), items)
    if action == "vty":
        self.vty_view(node.name, 4239, "hub-STP")
    elif action == "audit":
        self.audit_dialog()
    elif action == "link":
        self.console_menu()
    elif action == "log":
        self.show_text("Journal M3UA",
                       self.client.events if self.client else ["(pas de lien)"])
```

```
    elif action == "why":
        self.show_text("Hub distant", [
            "Le hub tourne sur %s : ce n'est pas cette machine." % self.topo.hub_ip,
            "",
            "Le VTY d'Osmocom n'ecoute que sur la boucle locale de SON noeud :",
            "il n'est donc pas interrogeable d'ici. Ce qui l'est, en revanche :",
            "",
            "  - le lien M3UA/SCTP de chaque operateur vers le hub ;",
            "  - l'etat des AS et ASP vus du cote operateur (VTY du STP local) ;",
            "  - l'audit DAUD d'un point code a travers le hub.",
            "",
            "Sur le noeud du hub : ./launch/start-interstp.sh --status",
        ])

def stp_menu(self, box):
    node = box.ref
    items = [("Ouvrir le VTY du STP (4239)", "vty"),
             ("AS / ASP / routes (vue rapide)", "quick"),
             ("Auditer le point code %s" % node.point_code, "audit"),
             ("Cibler ce noeud en MAP", "map"),
             ("Configuration heritee (osmo-stp.cfg)", "cfg"),
             ("Informations", "info")]
    action = self.menu("%s - STP %s" % (node.title, node.point_code), items)
    if action == "vty":
        self.vty_view(node.name, 4239, "%s-STP" % node.title)
    elif action == "quick":
        self.quick_menu(node, "OsmoSTP", 4239)
    elif action == "audit":
        self.audit_dialog(node.point_code)
    elif action == "map":
        self.map_dialog(node.point_code)
    elif action == "cfg":
        self.show_text("osmo-stp.cfg (%s)" % node.name,
                       node.cfg.get("osmo-stp.cfg", "(non lu)").split("\n"),
                       wrap=False)
    elif action == "info":
        lines = ["noeud        : %s" % node.name,
                 "point code   : %s" % node.point_code,
                 "M3UA local   : %s" % node.local_m3ua,
                 "hub          : %s:%s" % (node.hub_ip, node.hub_port),
                 "routing key  : %s" % node.routing_ctx,
                 "noeud WAN    : %s" % (node.wan_node or "-"),
                 "", "Adresses SCCP heritees des configurations :"]
        for name, addr in sorted(node.sccp.items()):
            lines.append("  %-22s PC %-8s SSN %-4s (%s)" % (
                name, addr.get("pc") or "-", addr.get("ssn") or "-",
                addr.get("source")))
        lines += ["", "Services VTY decouverts :"]
        for s in node.services:
            lines.append("  %-12s port %s" % (s.label, s.port))
        self.show_text("Informations %s" % node.title, lines)

def service_menu(self, box):
    svc = box.ref
    node = box.node
    items = [("Ouvrir le VTY (%d)" % svc.port, "vty"),
             ("Commandes rapides", "quick"),
             ("Informations", "info")]
    if svc.daemon in ("OsmoMSC", "OsmoHLR", "OsmoSGSN"):
        items.insert(2, ("Cibler ce demon en MAP", "map"))
    action = self.menu("%s - %s" % (node.title, svc.label), items)
    if action == "vty":
        self.vty_view(node.name, svc.port, "%s-%s" % (node.title, svc.label))
    elif action == "quick":
        self.quick_menu(node, svc.daemon, svc.port)
    elif action == "map":
```

```
        ssn = {"OsmoHLR": mapops.SSN_HLR, "OsmoMSC": mapops.SSN_MSC,
               "OsmoSGSN": 149}.get(svc.daemon, mapops.SSN_HLR)
        pc = ""
        for key in ("addr-msc", "addr-bsc"):
            if key in node.sccp and node.sccp[key].get("pc"):
                pc = node.sccp[key]["pc"]
                break
        if svc.daemon == "OsmoMSC" and node.sccp.get("_self_osmo-msc.cfg"):
            pc = node.sccp["_self_osmo-msc.cfg"].get("pc") or pc
        self.map_dialog(pc or node.point_code, ssn)
    elif action == "info":
        self.show_text("%s sur %s" % (svc.label, node.name), [
            "demon        : %s" % svc.daemon,
            "port VTY     : %s" % svc.port,
            "famille      : %s" % svc.family,
            "noeud        : %s (%s)" % (node.name, node.point_code),
            "",
            "Echap ferme un VTY sans fermer les autres : la console garde",
            "une session par port, et le schema reste accessible.",
        ])

def quick_menu(self, node, daemon, port):
    cmds = quickcmd.for_daemon(daemon)
    items = [("%-32s %s" % (label, cmd), cmd) for label, cmd in cmds]
    cmd = self.menu("Commandes rapides - %s" % daemon, items)
    if cmd is None:
        return
    try:
        out = self.pool.cmd(node.name, port, cmd)
    except vty_mod.VtyError as exc:
        out = "[VTY indisponible : %s]" % exc
    self.show_text("%s : %s" % (node.name, cmd), out.split("\n"), wrap=False)

# ── boucle principale ─────────────────────────────────────────────
# Les raccourcis sont en MINUSCULES uniquement : les sequences de fleches
# se terminent par A, B, C ou D, et une sequence mal decoupee par le
# terminal arriverait sinon comme un raccourci - une fleche fermerait le
# lien M3UA. Avec des minuscules seules, le pire cas ne fait rien.
def run(self):
    while True:
        self.draw()
        key = read_key(self.scr)
        if key == ord("q"):
            if self.pool.sessions and not self.confirm(
                    "Des sessions VTY sont ouvertes. Quitter ?"):
                continue
            return
        elif key == curses.KEY_RESIZE:
            continue
        elif key == curses.KEY_UP:
            self.sel = diagram.navigate(self.boxes, self.sel, (0, -1))
        elif key == curses.KEY_DOWN:
            self.sel = diagram.navigate(self.boxes, self.sel, (0, 1))
        elif key == curses.KEY_LEFT:
            self.sel = diagram.navigate(self.boxes, self.sel, (-1, 0))
        elif key == curses.KEY_RIGHT:
            self.sel = diagram.navigate(self.boxes, self.sel, (1, 0))
        elif key == 9:
            self.sel = (self.sel + 1) % max(1, len(self.boxes))
        elif key == curses.KEY_BTAB:
            self.sel = (self.sel - 1) % max(1, len(self.boxes))
        elif key in (10, 13, curses.KEY_ENTER):
            if self.boxes:
                self.open_box(self.boxes[self.sel])
        elif key == ord("v"):
            self.direct_vty()
```

```
            elif key == ord("s"):
                box = self.boxes[self.sel] if self.boxes else None
                pc = box.node.point_code if box and box.node else ""
                self.map_dialog(pc)
            elif key == ord("a"):
                box = self.boxes[self.sel] if self.boxes else None
                self.audit_dialog(box.node.point_code if box and box.node else "")
            elif key == ord("c"):
                self.toggle_link()
            elif key == ord("j"):
                self.show_text("Journal M3UA",
                               self.client.events if self.client else ["(pas de lien)"])
            elif key == ord("r"):
                self.say("redecouverte de la topologie ...")
                self.draw()
                self.pool.close_all()
                self.topo = topo_mod.discover()
                self.rebuild()
                self.say("topologie relue : %d operateur(s), hub %s"
                         % (len(self.topo.operators), self.topo.hub_ip or "-"), C_OK)
            elif key in (ord("?"), ord("h")):
                self.show_text("Aide", HELP)
            elif key == 27:
                self.say("Echap : deja au schema. q pour quitter, ? pour l'aide.")

    def direct_vty(self):
        if not self.boxes:
            return
        box = self.boxes[self.sel]
        if box.kind == "service":
            self.vty_view(box.node.name, box.ref.port,
                          "%s-%s" % (box.node.title, box.ref.label))
        elif box.kind == "stp":
            self.vty_view(box.ref.name, 4239, "%s-STP" % box.ref.title)
        elif box.kind == "hub" and box.ref and box.ref.name != "hub-distant":
            self.vty_view(box.ref.name, 4239, "hub-STP")
        else:
            self.say("cet element n'a pas de VTY joignable depuis ici", C_WARN)


def export_script(op, values, dest_pc, dest_ssn, dest_gt, local_pc, topo):
    """Ecrit un script Python autonome qui rejoue exactement cet envoi."""
    out_dir = os.path.join(topo_mod.REPO, "navigation", "scripts")
    os.makedirs(out_dir, exist_ok=True)
    stamp = time.strftime("%Y%m%d-%H%M%S")
    path = os.path.join(out_dir, "%s-%s.py" % (op.key, stamp))
    body = '''#!/usr/bin/env python3
# Genere par la console SS7 (navigation/) le {stamp}.
# Rejoue un envoi MAP {name} vers {dest}.
import sys, os
sys.path.insert(0, os.path.join(os.path.dirname(os.path.abspath(__file__)), "..", ".."))
from navigation import ss7

client = ss7.Ss7Client({hub!r}, {port!r}, local_pc={local!r})
print("lien M3UA :", client.connect())
rapport = client.send_map({op!r}, {values!r},
                          dest_pc={pc!r}, dest_ssn={ssn!r}, dest_gt={gt!r},
                          route_on_gt={rogt!r})
print(rapport)
client.close()
'''.format(stamp=stamp, name=op.name, dest=dest_gt or dest_pc,
           hub=topo.hub_ip, port=topo.hub_port, local=local_pc, op=op.key,
           values=values, pc=dest_pc or "0.0.0",
           ssn=int(dest_ssn) if dest_ssn else op.ssn, gt=dest_gt,
           rogt=bool(dest_gt and not dest_pc))
    with open(path, "w") as fh:
```

```
        fh.write(body)
    os.chmod(path, 0o755)
    return path


def main(topo):
    def _run(stdscr):
        curses.curs_set(0)
        try:
            curses.set_escdelay(60)
        except Exception:
            pass
        init_colors()
        stdscr.keypad(True)
        app = App(stdscr, topo)
        try:
            app.run()
        finally:
            app.pool.close_all()
            if app.client:
                app.client.close()
    curses.wrapper(_run)
```

3.244 /opt/GSM/osmo-operator/navigation/vty.py

9494 octets, 272 lignes → 272 lignes

```
# vty.py - acces aux VTY Osmocom depuis l'hote.
#
# Les VTY du lab n'ecoutent que sur 127.0.0.1 A L'INTERIEUR du noeud : depuis
# l'hote, on passe donc par "docker exec". Les conteneurs n'ont pas tous nc ;
# ils ont python3, telnet et socat. On pousse donc un petit relais python dans
# le conteneur, ce qui donne en prime une session PERSISTANTE : le VTY garde
# son contexte (enable, node de config) d'une commande a l'autre, exactement
# comme un telnet a la main.
#
# En natif (pas de docker), la meme classe parle directement au socket local.

import os
import re
import shutil
import socket
import subprocess
import threading
import time

EOC = b"\x01EOC\x01"

# Relais execute DANS le noeud. Protocole : une commande par ligne sur stdin,
# la sortie brute du VTY sur stdout, terminee par le marqueur EOC.
_RELAY = r'''
import socket, sys, time
port = int(sys.argv[1])
EOC = b"\x01EOC\x01\n"
try:
    s = socket.create_connection(("127.0.0.1", port), 5)
except Exception as e:
    sys.stdout.buffer.write(("__ERR__ %s\n" % e).encode()); sys.stdout.buffer.write(EOC)
    sys.stdout.buffer.flush(); sys.exit(0)
s.settimeout(0.35)
def pump(idle=0.35, hard=8.0):
    out = b""; t0 = time.time(); last = time.time()
    while True:
        try:
```

```
            d = s.recv(65536)
            if not d:
                break
            out += d; last = time.time()
        except socket.timeout:
            if out and (time.time() - last) >= idle:
                break
            if (time.time() - t0) > hard:
                break
    return out
sys.stdout.buffer.write(pump()); sys.stdout.buffer.write(EOC); sys.stdout.buffer.flush()
for line in sys.stdin:
    cmd = line.rstrip("\n")
    try:
        s.sendall(cmd.encode() + b"\r\n")
    except Exception as e:
        sys.stdout.buffer.write(("__ERR__ %s\n" % e).encode())
        sys.stdout.buffer.write(EOC); sys.stdout.buffer.flush(); break
    sys.stdout.buffer.write(pump()); sys.stdout.buffer.write(EOC); sys.stdout.buffer.flush()
'''

_IAC = re.compile(rb"\xff[\xfb-\xfe].|\xff[\xf0-\xfa]")


def clean(raw):
    """Sortie VTY lisible : negociation telnet, CR et invites en moins."""
    if isinstance(raw, str):
        raw = raw.encode("utf-8", "replace")
    raw = _IAC.sub(b"", raw)
    txt = raw.decode("utf-8", "replace").replace("\r", "")
    out = []
    for line in txt.split("\n"):
        # L'invite ("OsmoSTP> ", "OsmoHLR# ") revient a chaque commande : elle
        # n'apporte rien dans un panneau, et le prompt reel est affiche par la
        # console elle-meme.
        line = re.sub(r"^Osmo[A-Za-z0-9_-]*[#>]\s?", "", line)
        line = line.replace("\x00", "")
        out.append(line.rstrip())
    while out and not out[0].strip():
        out.pop(0)
    while out and not out[-1].strip():
        out.pop()
    return "\n".join(out)


class VtyError(Exception):
    pass


class VtySession(object):
    """Session VTY persistante vers un (noeud, port)."""

    def __init__(self, node, port, docker=True, timeout=12.0):
        self.node = node
        self.port = int(port)
        self.docker = docker
        self.timeout = timeout
        self.proc = None
        self.sock = None
        self.banner = ""
        self.lock = threading.Lock()
        self.alive = False

    # -- ouverture / fermeture ---------------------------------------------
    def open(self):
        if self.alive:
```

```
            return self
        if self.docker and not _container_up(self.node):
            raise VtyError("le conteneur %s ne tourne pas" % self.node)
        if self.docker:
            self.proc = subprocess.Popen(
                ["docker", "exec", "-i", self.node, "python3", "-", str(self.port)],
                stdin=subprocess.PIPE, stdout=subprocess.PIPE,
                stderr=subprocess.DEVNULL,
            )
            self.proc.stdin.write(_RELAY.encode())
            self.proc.stdin.flush()
            self.proc.stdin.close()
            # Le relais lit ensuite ses commandes sur... stdin, que l'on vient
            # de fermer. On repasse donc par un second canal : docker exec ne
            # sait pas rouvrir stdin, alors le relais est relance en mode
            # "script sur la ligne de commande" (voir _open_docker).
            self.proc.kill()
            self.proc = None
            return self._open_docker()
        try:
            self.sock = socket.create_connection(("127.0.0.1", self.port), 5)
        except OSError as exc:
            raise VtyError("VTY 127.0.0.1:%s injoignable (%s)"
                           % (self.port, exc.strerror or exc))
        self.sock.settimeout(0.35)
        self.banner = clean(self._pump_socket())
        self.alive = True
        return self

    def _open_docker(self):
        # -c : le relais arrive en argument, stdin reste libre pour les
        # commandes. C'est ce qui donne la session persistante.
        self.proc = subprocess.Popen(
            ["docker", "exec", "-i", self.node, "python3", "-c", _RELAY, str(self.port)],
            stdin=subprocess.PIPE, stdout=subprocess.PIPE,
            stderr=subprocess.DEVNULL, bufsize=0,
        )
        try:
            self.banner = clean(self._read_until_eoc())
        except Exception as exc:
            self.close()
            raise VtyError("VTY %s:%s injoignable (%s)" % (self.node, self.port, exc))
        if "__ERR__" in self.banner:
            msg = self.banner.strip()
            self.close()
            raise VtyError(msg.replace("__ERR__", "").strip())
        self.alive = True
        return self

    def close(self):
        self.alive = False
        if self.proc:
            try:
                self.proc.stdin.close()
            except Exception:
                pass
            try:
                self.proc.terminate()
                self.proc.wait(timeout=2)
            except Exception:
                try:
                    self.proc.kill()
                except Exception:
                    pass
            self.proc = None
        if self.sock:
```

```
            try:
                self.sock.close()
            except Exception:
                pass
            self.sock = None

    # ── dialogue ──────────────────────────────────────────────────────
    def _read_until_eoc(self):
        buf = b""
        deadline = time.time() + self.timeout
        while EOC not in buf:
            if time.time() > deadline:
                raise VtyError("delai depasse en lisant le VTY")
            chunk = self.proc.stdout.read(1)
            if not chunk:
                raise VtyError("relais VTY termine")
            buf += chunk
        return buf.split(EOC)[0]

    def _pump_socket(self, idle=0.35, hard=8.0):
        out = b""
        t0 = last = time.time()
        while True:
            try:
                d = self.sock.recv(65536)
                if not d:
                    break
                out += d
                last = time.time()
            except socket.timeout:
                if out and (time.time() - last) >= idle:
                    break
                if (time.time() - t0) > hard:
                    break
        return out

    def cmd(self, command):
        """Envoie une commande, rend la sortie nettoyee."""
        with self.lock:
            if not self.alive:
                self.open()
            if self.docker:
                try:
                    self.proc.stdin.write(command.encode() + b"\n")
                    self.proc.stdin.flush()
                    raw = self._read_until_eoc()
                except (BrokenPipeError, VtyError):
                    self.close()
                    raise VtyError("session VTY perdue (%s:%s)" % (self.node, self.port))
            else:
                self.sock.sendall(command.encode() + b"\r\n")
                raw = self._pump_socket()
            return clean(raw)


class VtyPool(object):
    """Sessions ouvertes, une par (noeud, port). Ferme a la demande - c'est
    ce que fait Echap dans la console : on quitte CE VTY, les autres restent
    disponibles."""

    def __init__(self, docker=True):
        self.docker = docker
        self.sessions = {}

    def get(self, node, port):
        key = (node, int(port))
```

```
        sess = self.sessions.get(key)
        if sess is None or not sess.alive:
            sess = VtySession(node, port, docker=self.docker)
            sess.open()
            self.sessions[key] = sess
        return sess

    def cmd(self, node, port, command):
        return self.get(node, port).cmd(command)

    def drop(self, node, port):
        key = (node, int(port))
        sess = self.sessions.pop(key, None)
        if sess:
            sess.close()
            return True
        return False

    def close_all(self):
        for sess in list(self.sessions.values()):
            sess.close()
        self.sessions.clear()


def _container_up(name):
    try:
        out = subprocess.run(["docker", "ps", "--format", "{{.Names}}"],
                             stdout=subprocess.PIPE, stderr=subprocess.DEVNULL,
                             timeout=10).stdout.decode()
    except Exception:
        return False
    return name in out.split()


def docker_available():
    return shutil.which("docker") is not None and os.system(
        "docker ps >/dev/null 2>&1") == 0
```

3.245 /opt/GSM/osmo-operator/network/apn-nat.sh

2437 octets, 55 lignes → 55 lignes

```
#!/bin/bash
# ============================================================================
# network/apn-nat.sh - sortie Internet des telephones (APN internet, apn0)
#
# Le GGSN range les telephones dans 176.16.32.0/24 sur l'interface apn0. Sans
# NAT ni forward, leurs paquets (DNS compris) partent dans le vide : la policy
# FORWARD de l'hote est en DROP. Ce script pose, de facon idempotente :
#   - MASQUERADE 176.16.32.0/24 vers l'interface de sortie (carte qui route vers
#     Internet, detectee par la route par defaut) ;
#   - FORWARD apn0 <-> sortie, et le retour des connexions etablies.
#
#   network/apn-nat.sh          applique
#   network/apn-nat.sh --status etat, sans rien changer
# ============================================================================
set -u

SUBNET="${APN_SUBNET:-176.16.32.0/24}"
APN_IF="${APN_IF:-apn0}"
MODE=apply
[ "${1:-}" = "--status" ] && MODE=status

OUT_IF="$(ip route get 1.1.1.1 2>/dev/null | sed -n 's/.* dev \([^ ]*\).*/\1/p' | head -1)"
```

```
if [ -z "$OUT_IF" ]; then
    echo "apn-nat : pas de route vers Internet - rien a faire"; exit 1
fi

ok() { printf '  %-58s %s\n' "$1" "$2"; }

check_or_add() {   # $1 = table (nat|filter), $2 = chaine, reste = regle
    local table="$1" chain="$2"; shift 2
    if iptables -w -t "$table" -C "$chain" "$@" 2>/dev/null; then
        ok "$table $chain $*" "deja present"
    elif [ "$MODE" = status ]; then
        ok "$table $chain $*" "ABSENT"
    else
        if iptables -w -t "$table" -I "$chain" "$@" 2>/dev/null; then
            ok "$table $chain $*" "ajoute"
        else
            ok "$table $chain $*" "ECHEC"; return 1
        fi
    fi
}

echo "apn-nat : $SUBNET via $APN_IF -> $OUT_IF"
rc=0
# [2026-10-16] sans forward, MASQUERADE et FORWARD ne servent a rien (repris du bloc
# « NAT du pool GGSN » de start-native.sh, retire au profit de ce script).
if [ "$(sysctl -n net.ipv4.ip_forward 2>/dev/null)" = 1 ]; then ok "net.ipv4.ip_forward" "deja 1"
elif [ "$MODE" = status ]; then ok "net.ipv4.ip_forward" "0 (ABSENT)"
elif sysctl -qw net.ipv4.ip_forward=1 2>/dev/null; then ok "net.ipv4.ip_forward" "mis a 1"
else ok "net.ipv4.ip_forward" "ECHEC"; rc=1; fi
check_or_add nat POSTROUTING -s "$SUBNET" -o "$OUT_IF" -j MASQUERADE || rc=1
check_or_add filter FORWARD -i "$APN_IF" -o "$OUT_IF" -j ACCEPT || rc=1
check_or_add filter FORWARD -i "$OUT_IF" -o "$APN_IF" -m state --state RELATED,ESTABLISHED -j ACCEPT || rc=1
exit $rc
```

3.246 /opt/GSM/osmo-operator/network/firewall-wan.sh

3085 octets, 76 lignes → 76 lignes

```
#!/bin/bash
# ═══════════════════════════════════════════════════════════════════════
# network/firewall-wan.sh - Ouvre les ports necessaires pour l'interop WAN
#
# Usage : sudo ./network/firewall-wan.sh <remote_ip> [n_operators]
# ═══════════════════════════════════════════════════════════════════════

set -euo pipefail

REMOTE_IP="${1:?Usage: $0 <remote_ip> [n_operators]}"
N_OPS="${2:-2}"
SIP_WAN_BASE=5080
RTP_WAN_BASE=20000
RTP_PER_OP=500
SMS_RELAY_PORT=7890

echo "=== Firewall WAN Interop - Autorisation ${REMOTE_IP} ==="
echo ""

# Detection du firewall
if command -v ufw &>/dev/null && ufw status | grep -q "active"; then
    FW="ufw"
elif command -v firewall-cmd &>/dev/null; then
    FW="firewalld"
else
    FW="iptables"
```

```
fi

echo "Firewall detecte : ${FW}"
echo ""

for i in $(seq 1 "$N_OPS"); do
    sip_port=$(( SIP_WAN_BASE + (i - 1) * 2 ))
    rtp_start=$(( RTP_WAN_BASE + (i - 1) * RTP_PER_OP ))
    rtp_end=$(( RTP_WAN_BASE + i * RTP_PER_OP - 1 ))

    echo "Op${i}: SIP ${sip_port}/udp  RTP ${rtp_start}:${rtp_end}/udp"

    case "$FW" in
        ufw)
            ufw allow from "$REMOTE_IP" to any port "$sip_port" proto udp
            ufw allow from "$REMOTE_IP" to any port "${rtp_start}:${rtp_end}" proto udp
            ;;
        firewalld)
            firewall-cmd --permanent --add-rich-rule="rule family=ipv4 source address=${REMOTE_IP} port
            ↪  port=${sip_port} protocol=udp accept"
            firewall-cmd --permanent --add-rich-rule="rule family=ipv4 source address=${REMOTE_IP} port
            ↪  port=${rtp_start}-${rtp_end} protocol=udp accept"
            ;;
        iptables)
            # -C ... || -A : idempotent, pas de doublon si relance a chaque WAN.
            iptables -C INPUT -s "$REMOTE_IP" -p udp --dport "$sip_port" -j ACCEPT 2>/dev/null \
                || iptables -A INPUT -s "$REMOTE_IP" -p udp --dport "$sip_port" -j ACCEPT
            iptables -C INPUT -s "$REMOTE_IP" -p udp --dport "${rtp_start}:${rtp_end}" -j ACCEPT 2>/dev/null \
                || iptables -A INPUT -s "$REMOTE_IP" -p udp --dport "${rtp_start}:${rtp_end}" -j ACCEPT
            ;;
    esac
done

# SMS relay port
echo ""
echo "SMS Relay: ${SMS_RELAY_PORT}/tcp"
case "$FW" in
    ufw)
        ufw allow from "$REMOTE_IP" to any port "$SMS_RELAY_PORT" proto tcp
        ;;
    firewalld)
        firewall-cmd --permanent --add-rich-rule="rule family=ipv4 source address=${REMOTE_IP} port
        ↪  port=${SMS_RELAY_PORT} protocol=tcp accept"
        firewall-cmd --reload
        ;;
    iptables)
        iptables -C INPUT -s "$REMOTE_IP" -p tcp --dport "$SMS_RELAY_PORT" -j ACCEPT 2>/dev/null \
            || iptables -A INPUT -s "$REMOTE_IP" -p tcp --dport "$SMS_RELAY_PORT" -j ACCEPT
        ;;
esac

echo ""
echo "✓ Ports ouverts pour ${REMOTE_IP}"
```

3.247 /opt/GSM/osmo-operator/network/loopback.sh

1856 octets, 87 lignes → 87 lignes

```
#!/usr/bin/env bash
set -euo pipefail

SOURCE="gsm_audio.monitor"
ACTION="${1:-status}"
SINK="${2:-$(pactl get-default-sink)}"
```

```
find_loopback_ids() {
    pactl list short modules | \
        grep -F "module-loopback" | \
        grep -F "source=${SOURCE}" | \
        grep -F "sink=${SINK}" | \
        awk '{print $1}'
}

check_source() {
    pactl list short sources | grep -F "${SOURCE}" >/dev/null || {
        echo "[FAIL] source ${SOURCE} introuvable"
        exit 1
    }
}

enable_loopback() {
    check_source
    local ids
    ids="$(find_loopback_ids || true)"

    echo "[*] Source : $SOURCE"
    echo "[*] Sink   : $SINK"

    if [[ -n "${ids}" ]]; then
        echo "[OK] loopback deja present : ${ids}"
    else
        local mid
        mid="$(pactl load-module module-loopback source="${SOURCE}" sink="${SINK}" latency_msec=20)"
        echo "[OK] loopback charge, module id=${mid}"
    fi
}

disable_loopback() {
    local ids
    ids="$(find_loopback_ids || true)"

    echo "[*] Source : $SOURCE"
    echo "[*] Sink   : $SINK"

    if [[ -z "${ids}" ]]; then
        echo "[OK] aucun loopback a retirer"
        return 0
    fi

    while read -r id; do
        [[ -n "${id}" ]] || continue
        pactl unload-module "${id}"
        echo "[OK] loopback retire, module id=${id}"
    done <<< "${ids}"
}

status_loopback() {
    local ids
    ids="$(find_loopback_ids || true)"

    echo "[*] Source : $SOURCE"
    echo "[*] Sink   : $SINK"

    if [[ -n "${ids}" ]]; then
        echo "[OK] loopback actif : ${ids}"
    else
        echo "[INFO] aucun loopback actif"
    fi
}
```

```
case "${ACTION}" in
    enable)
        enable_loopback
        ;;
    disable)
        disable_loopback
        ;;
    status)
        status_loopback
        ;;
    *)
        echo "Usage: $0 {enable|disable|status} [sink]"
        exit 1
        ;;
esac
```

3.248 /opt/GSM/osmo-operator/network/node-conf.sh

7938 octets, 152 lignes → 152 lignes

```
#!/bin/bash
# network/node-conf.sh - LA FICHE D'UN NOEUD, produite par le run, lue par les
# autres ISO : « mynode<N>.conf ».
#
#   noeud 1 :  ./start-native.sh              -> ecrit /etc/osmocom/mynode1.conf
#              ./start-native.sh --gen-conf   -> la meme fiche, sans lancer
#   noeud 2 :  ./start-native.sh --wan mynode1.conf [mynode3.conf ...]
#              -> table WAN = ce noeud + les fiches, puis le maillage habituel
#                 (network/setup-wan-mesh.sh : trunks SIP, dialplan, routes SMS)
#
# Un « noeud » est une ISO : son operateur natif, ses conteneurs multi-operateur
# et son inter-STP. Le WAN relie plusieurs ISO. La fiche dit tout ce qu'un autre
# noeud doit savoir pour le joindre : identite SS7, adresse publique, indicatif,
# nombre d'operateurs (donc ports SIP/SMS), plan de numerotation. Elle est
# ecrite au format shell KEY=valeur, comme /etc/osmo-wan.conf, pour se lire
# d'un `.` et se relire a l'oeil.
#
# Cette bibliotheque se SOURCE (start-native.sh, start-multi.sh). Elle suppose
# network/wan-nodes.sh deja chargee (WAN_IP/WAN_IND/WAN_NOPS/WAN_NODE_LIST,
# wan_default_ind, wan_detect_local_ip).
: "${OSMOCOM_CFG:=/etc/osmocom}"
NODE_CONF_DIR="${NODE_CONF_DIR:-$OSMOCOM_CFG}"

node_conf_path() { printf '%s/mynode%s.conf' "$NODE_CONF_DIR" "${1:-1}"; }

# Combien d'operateurs porte CE noeud : le natif + les conteneurs osmo-operator-N
# EN MARCHE (start-multi.sh). La topologie ecrite par addition.sh existe meme
# sans multi lance ; ce qui compte pour les ports SIP/SMS du maillage, c'est ce
# qui repond. OSMO_NODE_NOPS force la valeur.
node_conf_nops() {
    local n=1 c
    if [ -n "${OSMO_NODE_NOPS:-}" ]; then printf '%s' "$OSMO_NODE_NOPS"; return 0; fi
    if command -v docker >/dev/null 2>&1; then
        c="$(docker ps --format '{{.Names}}' 2>/dev/null | grep -c '^osmo-operator-[0-9]*$')"
        n=$(( 1 + ${c:-0} ))
    fi
    printf '%s' "$n"
}

# La liste des operateurs du noeud : "idx:mode:pc_stp:rctx_inter[:ip]" - le
# natif d'abord, puis les conteneurs de /etc/osmocom/osmo-multi.conf qui
# tournent. Le plan de point codes est celui du WAN : 1.<noeud><op>.2.
node_conf_ops() {
    local node="$1" op="$2" out="" spec idx mode ip running=""
```

```
    out="${op}:native:1.${node}${op}.2:$(( node * 1000 + op * 100 + 50 ))"
    if command -v docker >/dev/null 2>&1; then
        running="$(docker ps --format '{{.Names}}' 2>/dev/null)"
    fi
    if [ -r "$OSMOCOM_CFG/osmo-multi.conf" ]; then
        for spec in $(sed -n 's/^MULTI_OPS="\{0,1\}\([^"]*\)"\{0,1\}/\1/p' "$OSMOCOM_CFG/osmo-multi.conf"); do
            IFS=: read -r idx mode ip _ _ <<< "$spec"
            [ "$mode" = docker ] || continue
            printf '%s\n' "$running" | grep -qx "osmo-operator-$idx" || continue
            out="$out ${idx}:docker:1.${node}${idx}.2:$(( node * 1000 + idx * 100 + 50 )):${ip}"
        done
    fi
    printf '%s' "$out"
}

# node_conf_write FICHIER NOEUD OP [HUB_IP]
# Ecrit la fiche. IP publique et indicatif : la table WAN si elle connait ce
# noeud (wan_nodes_load a ete fait), sinon detection locale + indicatif par
# defaut du numero de noeud (11, 22, 33...).
node_conf_write() {
    local out="$1" node="$2" op="${3:-1}" hub="${4:-}"
    local ip ind nops ops mcc mnc arfcn nms pfx sip sms
    [[ "$node" =~ ^[1-9]$ ]] || { echo "node_conf_write : noeud '$node' invalide (1-9)" >&2; return 1; }
    ip="${WAN_IP[$node]:-}"
    [ -n "$ip" ] || ip="$(wan_detect_local_ip 2>/dev/null)"
    [ -n "$ip" ] || ip="127.0.0.1"
    ind="${WAN_IND[$node]:-}"
    [ -n "$ind" ] || ind="$(wan_default_ind "$node")"
    nops="$(node_conf_nops)"
    ops="$(node_conf_ops "$node" "$op")"
    mcc="$(sed -n 's/^PLAN_MCC=//p' "$OSMOCOM_CFG/radio-plan.env" 2>/dev/null | tail -1)"
    mnc="$(sed -n 's/^PLAN_MNC=//p' "$OSMOCOM_CFG/radio-plan.env" 2>/dev/null | tail -1)"
    arfcn="$(sed -n 's/^PLAN_ARFCN=//p' "$OSMOCOM_CFG/radio-plan.env" 2>/dev/null | tail -1)"
    [ -n "$mcc" ] || mcc="$(awk '/^ *network country code /{print $4; exit}' "$OSMOCOM_CFG/osmo-msc.cfg"
    ↪  2>/dev/null)"
    [ -n "$mnc" ] || mnc="$(awk '/^ *mobile network code /{print $4; exit}' "$OSMOCOM_CFG/osmo-msc.cfg"
    ↪  2>/dev/null)"
    [ -n "$arfcn" ] || arfcn="$(awk '/^ *arfcn /{print $2; exit}' "$OSMOCOM_CFG/osmo-bsc.cfg" 2>/dev/null)"
    nms="${N_MS:-$(sed -n 's/^: "\${N_MS:=\([0-9]*\)}"/\1/p' "$OSMOCOM_CFG/coeur.env" 2>/dev/null | tail -1)}"
    nms="${nms:-2}"
    pfx="${node}00"
    # Ports vus depuis l'exterieur, memes formules que setup-wan-mesh.sh :
    # un seul operateur -> 5060 / 7890 ; sinon 5080+(j-1)*2 / 7890+j-1.
    if [ "$nops" = 1 ]; then sip=5060; sms=7890; else sip="5080+(j-1)*2"; sms="7890+j-1"; fi
    mkdir -p "$(dirname "$out")" || return 1
    cat > "$out.tmp" <<EOF
# mynode${node}.conf - fiche du noeud ${node}, ecrite par start-native.sh le $(date '+%F %T')
# A copier sur un autre noeud et a lui donner :  ./start-native.sh --wan $(basename "$out")
# Format shell (KEY=valeur). Les champs OSMO_NODE_ID/IP/IND/NOPS suffisent au
# maillage (table WAN "id:ip:indicatif:operateurs") ; le reste documente le noeud.
OSMO_NODE_CONF_VERSION=1
OSMO_NODE_ID=${node}
OSMO_NODE_IP=${ip}
OSMO_NODE_IND=${ind}
OSMO_NODE_NOPS=${nops}
OSMO_NODE_HOSTNAME=$(hostname 2>/dev/null)
# operateurs : "idx:mode:pc_stp:rctx_inter[:backbone_ip]" (natif puis conteneurs en marche)
OSMO_NODE_OPS="${ops}"
OSMO_NODE_OP=${op}
OSMO_NODE_PC_MSC=1.${node}${op}.1
OSMO_NODE_PC_STP=1.${node}${op}.2
OSMO_NODE_PC_BSC=1.${node}${op}.3
OSMO_NODE_RCTX_INTER=$(( node * 1000 + op * 100 + 50 ))
OSMO_NODE_HUB_IP=${hub}
OSMO_NODE_HUB_PORT=2908
OSMO_NODE_MCC=${mcc}
```

```
OSMO_NODE_MNC=${mnc}
OSMO_NODE_ARFCN=${arfcn}
OSMO_NODE_N_MS=${nms}
# numerotation : MSISDN = <noeud>00<op><ms> ; depuis un autre noeud : <indicatif><MSISDN>
OSMO_NODE_MSISDN_PFX=${pfx}
OSMO_NODE_SIP_PORT="${sip}"
OSMO_NODE_SMS_PORT="${sms}"
EOF
    mv -f "$out.tmp" "$out" && chmod 644 "$out"
}

# node_conf_import FICHIER : lit une fiche et pose son noeud dans la table WAN
# (WAN_IP/WAN_IND/WAN_NOPS/WAN_NODE_LIST) - remplace l'entree si elle existait.
# Lecture par sed, pas par `.` : une fiche venue d'ailleurs n'execute rien ici.
node_conf_import() {
    local f="$1" id ip ind nops
    [ -r "$f" ] || { _wan_err "fiche introuvable : $f"; return 1; }
    id="$(sed -n 's/^OSMO_NODE_ID=//p' "$f" | tail -1 | tr -d '" ')"
    ip="$(sed -n 's/^OSMO_NODE_IP=//p' "$f" | tail -1 | tr -d '" ')"
    ind="$(sed -n 's/^OSMO_NODE_IND=//p' "$f" | tail -1 | tr -d '" ')"
    nops="$(sed -n 's/^OSMO_NODE_NOPS=//p' "$f" | tail -1 | tr -d '" ')"
    [[ "$id" =~ ^[1-9]$ ]] || { _wan_err "$f : OSMO_NODE_ID='$id' (attendu 1-9)"; return 1; }
    [[ "$ip" =~ ^[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+$ ]] || { _wan_err "$f : OSMO_NODE_IP='$ip'"; return 1; }
    [[ "$ind" =~ ^[0-9]{2,4}$ ]] || ind="$(wan_default_ind "$id")"
    [[ "$nops" =~ ^[1-9][0-9]*$ ]] || nops=1
    if [ -z "${WAN_IP[$id]:-}" ]; then WAN_NODE_LIST+=("$id"); fi
    WAN_IP[$id]="$ip"; WAN_IND[$id]="$ind"; WAN_NOPS[$id]="$nops"
    WAN_NODE_COUNT=${#WAN_NODE_LIST[@]}
    printf '  %-4s noeud %s  %-15s indicatif %-4s %s operateur(s)  (%s)\n' "" "$id" "$ip" "$ind" "$nops"
     ↪  "$(basename "$f")"
    return 0
}

# node_conf_self NOEUD : garantit que CE noeud figure dans la table (adresse
# locale detectee, indicatif par defaut, operateurs comptes), sans ecraser une
# entree deja connue de la table.
node_conf_self() {
    local id="$1" ip
    [ -n "${WAN_IP[$id]:-}" ] && { WAN_NOPS[$id]="$(node_conf_nops)"; return 0; }
    ip="$(wan_detect_local_ip 2>/dev/null)" || ip=""
    [ -n "$ip" ] || { _wan_err "aucune adresse locale detectee pour le noeud $id (--wan-nodes pour la donner)";
     ↪  return 1; }
    WAN_IP[$id]="$ip"; WAN_IND[$id]="$(wan_default_ind "$id")"; WAN_NOPS[$id]="$(node_conf_nops)"
    WAN_NODE_LIST+=("$id"); WAN_NODE_COUNT=${#WAN_NODE_LIST[@]}
    printf '  %-4s noeud %s  %-15s indicatif %-4s %s operateur(s)  (ce noeud)\n' "" "$id" "$ip" "${WAN_IND[$id]}"
     ↪  "${WAN_NOPS[$id]}"
}
```

3.249 /opt/GSM/osmo-operator/network/osmo-ip-plan.sh

10997 octets, 205 lignes → 205 lignes

```
#!/usr/bin/env bash
# ============================================================================
# network/osmo-ip-plan.sh - QUI PORTE LES ADRESSES PRIVEES DU NOEUD, ET OU
# ============================================================================
#
# CE QUI NE MARCHAIT PAS
# ----------------------
# Les adresses privees du noeud (192.168.<noeud+1>.1 et .10) etaient posees par
# systemd-networkd, dans 20-dhcp.network, sous un [Match] Name=en* eth*. Deux
# consequences, et la seconde est la vraie :
#
#   1. TOUTES les cartes qui repondent au motif recoivent les deux adresses.
```

```
#      Une machine a deux cartes se retrouve avec la meme /32 des deux cotes.
#   2. Surtout, le motif ne dit RIEN de la carte qui porte reellement le
#      reseau. Sur une VM a plusieurs interfaces, "en*" designe aussi bien le
#      NAT que le pont ; l'adresse se posait donc regulierement sur une carte
#      qui ne mene nulle part, pendant que celle qui a la route par defaut ne
#      l'avait pas. Un pair qui visait 192.168.2.10 ne trouvait personne, et
#      "ip addr" montrait pourtant l'adresse presente - sur l'autre carte.
#
# systemd-networkd ne sait pas exprimer "la carte qui a la route par defaut" :
# c'est une propriete d'EXECUTION, pas de configuration. D'ou ce script, appele
# au demarrage et a chaque changement de lien.
#
# LE REPLI, ET POURQUOI IL EXISTE
# -------------------------------
# Quand AUCUNE carte ne fournit Internet - VM sans reseau, cable debranche,
# DHCP muet - il n'y a aucun endroit sensé ou poser une adresse de segment. On
# ne la pose alors pas dans le vide : on met 127.0.0.66/32 sur la boucle
# locale. Les configurations qui nomment une adresse privee trouvent ainsi
# TOUJOURS quelque chose de local et de joignable, les demons demarrent, et le
# banc tourne en autarcie au lieu d'echouer au bind sur une adresse absente.
#
# DOCKER OU NATIF
# ---------------
# Le plan d'adressage n'est pas le meme des deux cotes, et prendre l'un pour
# l'autre donne des demons qui se lient a des adresses qui n'existent pas ici :
#   docker  le conteneur a son adresse sur le bridge avant que les demons ne
#           demarrent, et le backbone est 172.20.0.x. On ne touche a rien :
#           c'est docker qui adresse.
#   natif   l'adresse vient du DHCP, elle peut manquer au lancement, et le
#           segment prive du noeud est 192.168.<noeud+1>.x.
# « docker » = une image DE CE DEPOT : OSMO_IMAGE=1 (pose par le Dockerfile), ou
# en repli /.dockerenv ET /etc/docker-entrypoint-cmd. Tout autre conteneur
# (ubuntu:24.04 nu) est adresse comme une machine native (voir detect_runtime_env).
#
# Usage :   osmo-ip-plan.sh [--apply|--show] [--node N]
# ------------------------------------------------------------------------------
set -uo pipefail

ACTION=apply
NODE=""
while [ $# -gt 0 ]; do
    case "$1" in
        --apply)  ACTION=apply ;;
        --show)   ACTION=show ;;
        --node)   NODE="${2:-}"; shift ;;
        --node=*) NODE="${1#*=}" ;;
        -h|--help) sed -n '2,45p' "$0" | sed 's/^# \{0,1\}//'; exit 0 ;;
        *) echo "option inconnue : $1" >&2; exit 2 ;;
    esac
    shift
done

LOOPBACK_FALLBACK="${OSMO_PRIV_FALLBACK:-127.0.0.66}"

# ── docker ou natif ──────────────────────────────────────────────────────
# [2026-10-18] « docker » veut dire ICI : une image de CE depot, dont le
# reseau est pose par start-docker.sh / compose (backbone 172.20.0.x). Le Dockerfile
# pose ENV OSMO_IMAGE=1 ; en repli (variable perdue sous sudo, ou dans une
# unite systemd qui n herite pas de l environnement du conteneur), le couple
# /.dockerenv + /etc/docker-entrypoint-cmd ecrit par scripts/entrypoint.sh.
# Plus de repli sur le seul /.dockerenv ni sur le cgroup : un ubuntu:24.04 nu
# (le conteneur jetable du notebook d installation) passait pour « docker »,
# personne n y posait 192.168.2.10, et osmo-ggsn / osmo-sgsn refusaient de
# demarrer. Tout autre conteneur est donc traite comme une machine native.
detect_runtime_env() {
    if [ "${OSMO_IMAGE:-0}" = 1 ]; then echo docker; return; fi
```

```
    if [ -f /.dockerenv ] && [ -f /etc/docker-entrypoint-cmd ]; then echo docker; return; fi
    echo native
}

# ── le noeud ──────────────────────────────────────────────────────────
resolve_node() {
    local n="${NODE:-${OSMO_WAN_NODE:-${WAN_NODE_ID:-}}}"
    [ -n "$n" ] || n="$(awk -F= '/^OSMO_WAN_NODE=/{gsub(/[ \r\t]/,"",$2);v=$2} END{print v}' \
                        /etc/osmo-role 2>/dev/null)"
    [ -n "$n" ] || n="$(sed -n 's/^PLAN_NODE=//p' /etc/osmocom/radio-plan.env 2>/dev/null | tail -1)"
    case "$n" in [1-9]) ;; *) n=1 ;; esac
    printf '%s' "$n"
}

# ── LA carte qui fournit Internet ─────────────────────────────────────
# "ip route get" repond ce que le NOYAU ferait vraiment : c'est la seule
# reponse qui ne devine pas. On essaie une adresse publique, puis la passerelle
# par defaut - un banc coupe d'Internet garde en general sa route par defaut.
uplink_dev() {
    local d
    d="$(ip route get 1.1.1.1 2>/dev/null | sed -n 's/.*dev \([^ ]*\).*/\1/p' | head -1)"
    [ -n "$d" ] || d="$(ip route show default 2>/dev/null | sed -n 's/.*dev \([^ ]*\).*/\1/p' | head -1)"
    # lo ne fournit rien : si la route par defaut y mene, c'est qu'il n'y en a pas.
    [ "$d" = "lo" ] && d=""
    printf '%s' "$d"
}

# ── CE QUE LES CONFIGS EXIGENT, PAS CE QU'ON SUPPOSE ──────────────────
# Le plan prive (192.168.<noeud+1>.x) est une convention ; ce qui compte, c'est
# l'adresse que chaque demon va REELLEMENT essayer de lier. Elle est ecrite dans
# /etc/osmocom, et elle peut venir d'un autre generateur que le notre :
# qosmo-grgsm/run_modules/08-gabarits.sh passe op_private_ip() a
# apply_config_templates, donc 192.168.2.10, quand la recette native de l'ISO
# passe 127.0.0.2. Deux verites, et le demon suit la sienne :
#     osmo-ggsn : « gtp bind-ip 192.168.2.10 » -> bind sur une adresse absente
#     osmo-sgsn : « listen 192.168.2.10 23000 » -> idem
# On EXTRAIT donc les adresses des fichiers au lieu de les recalculer : la
# machine finit par porter ce que ses configs demandent, quel que soit celui
# qui les a ecrites.
#
# ATTENTION A CE QU ON NE PREND PAS : « ip dns 0/1 » (serveurs annonces a
# l'abonne), « remote-ip », « ggsn 0 remote-ip » designent des adresses
# DISTANTES. Les poser localement ferait repondre la machine a la place du pair.
addrs_des_configs() {
    local d="${OSMOCOM_CFG:-/etc/osmocom}"
    [ -d "$d" ] || return 0
    {
        sed -n 's/^[[:space:]]*gtp[[:space:]]\+bind-ip[[:space:]]\+\([0-9.]*\).*/\1/p'  "$d"/osmo-ggsn.cfg
         ↪  2>/dev/null
        sed -n 's/^[[:space:]]*gtp[[:space:]]\+local-ip[[:space:]]\+\([0-9.]*\).*/\1/p' "$d"/osmo-sgsn.cfg
         ↪  2>/dev/null
        sed -n 's/^[[:space:]]*listen[[:space:]]\+\([0-9.]*\)[[:space:]]\+[0-9]*.*/\1/p' "$d"/osmo-sgsn.cfg
         ↪  2>/dev/null
        sed -n 's/^[[:space:]]*local-addr[[:space:]]\+\([0-9.]*\).*/\1/p'                 "$d"/osmo-upf.cfg
         ↪  2>/dev/null
    } | sort -u | while read -r a; do
        case "$a" in
            ''|0.0.0.0|127.*) continue ;;      # deja la, ou « toutes les cartes »
        esac
        printf '%s\n' "$a"
    done
}

RUNTIME="$(detect_runtime_env)"
NODE="$(resolve_node)"
PRIV_BASE=$(( NODE + 1 ))
```

```
PRIV_GW="192.168.${PRIV_BASE}.1"
PRIV_IP="192.168.${PRIV_BASE}.10"
DEV="$(uplink_dev)"

if [ "$ACTION" = show ] || [ "${OSMO_IP_PLAN_VERBOSE:-1}" = 1 ]; then
    printf 'osmo-ip-plan : runtime=%s noeud=%s prive=%s,%s carte=%s\n' \
        "$RUNTIME" "$NODE" "$PRIV_GW" "$PRIV_IP" "${DEV:-aucune}"
fi
[ "$ACTION" = show ] && exit 0

# ── DOCKER : on ne touche a rien ──────────────────────────────────
if [ "$RUNTIME" = docker ]; then
    echo "osmo-ip-plan : conteneur - l'adressage appartient a docker, rien a poser"
    exit 0
fi

# ── On retire d'abord les adresses de PARTOUT ─────────────────────
# Sans ce nettoyage, changer de carte (cable debranche, Wi-Fi qui prend le
# relais) laissait l'ancienne adresse en place sur l'ancienne carte : deux
# interfaces revendiquaient la meme /32 et le noyau choisissait la source par
# ordre d'apparition, pas par route.
for a in "$PRIV_GW" "$PRIV_IP" "$LOOPBACK_FALLBACK"; do
    while read -r ifn; do
        [ -n "$ifn" ] || continue
        ip addr del "${a}/32" dev "$ifn" 2>/dev/null || true
    done < <(ip -o -4 addr show 2>/dev/null | awk -v a="$a" '$4 ~ "^"a"/" {print $2}')
done

# L'union du plan du noeud et de ce que les configs reclament. Les doublons
# tombent au tri ; « ip addr add » d'une adresse deja posee est sans effet.
BESOINS="$(printf '%s\n%s\n%s\n' "$PRIV_GW" "$PRIV_IP" "$(addrs_des_configs)" | sed '/^$/d' | sort -u)"

if [ -n "$DEV" ]; then
    _pose=""
    for a in $BESOINS; do
        # Deja portee par une carte quelconque : on n'y touche pas. Reposer une
        # adresse ailleurs ferait deux interfaces sur la meme /32.
        if ip -o -4 addr show 2>/dev/null | awk '{print $4}' | grep -qx "${a}/32\|${a}/[0-9]*"; then
            continue
        fi
        ip -o -4 addr show 2>/dev/null | awk -F'[ /]+' '{print $4}' | grep -qx "$a" && continue
        ip addr add "${a}/32" dev "$DEV" 2>/dev/null && _pose="$_pose $a"
    done
    if [ -n "$_pose" ]; then
        echo "osmo-ip-plan :${_pose} pose(s) en /32 sur ${DEV} (carte qui fournit Internet)"
    else
        echo "osmo-ip-plan : rien a poser, tout est deja la (${DEV})"
    fi
else
    # ── REPLI : la boucle locale ──────────────────────────────────
    # Aucune carte ne mene nulle part. Poser 192.168.x sur une interface au
    # hasard donnerait une adresse presente et injoignable - le pire des deux
    # mondes, parce que tout a l'air en place. Une adresse de boucle, elle, est
    # honnete : locale, joignable, et elle ne pretend rien.
    # Sur lo, une adresse absente empeche un demon de demarrer tout autant que
    # sur une carte : on y pose donc AUSSI ce que les configs reclament, plutot
    # que le seul repli. Le banc tourne alors en autarcie, complet.
    ip addr add "${LOOPBACK_FALLBACK}/32" dev lo 2>/dev/null || true
    for a in $BESOINS; do
        ip -o -4 addr show 2>/dev/null | awk -F'[ /]+' '{print $4}' | grep -qx "$a" && continue
        ip addr add "${a}/32" dev lo 2>/dev/null || true
    done
    echo "osmo-ip-plan : aucune carte ne fournit Internet - ${LOOPBACK_FALLBACK}/32 + $(echo $BESOINS) sur lo"
fi
exit 0
```

3.250 /opt/GSM/osmo-operator/network/pcu-socket.sh

2798 octets, 72 lignes → 72 lignes

```
#!/bin/bash
# ============================================================================
# network/pcu-socket.sh - harnais du socket PCU (BTS <-> osmo-pcu)
#
# Le PCU (GPRS/EGPRS) se branche sur UN socket unix par BTS :
#   - le BTS principal (osmo-bts-trx.cfg) : socket par defaut /tmp/pcu_bts,
#     c'est celui que run.sh donne au PCU (PCU_SOCKET) ;
#   - le BTS side-car (osmo-bts-trx-bts1.cfg) : pcu-socket /tmp/pcu_bts2, qu'aucun
#     PCU ne sert. Un socket sans processus derriere reste sur le disque et
#     fait croire a une liaison qui n'existe pas.
#
# Ce script ne change AUCUNE config. Il dit qui ecoute quoi, et supprime les
# sockets orphelins (fichier present, aucun processus en ecoute) pour que le
# lancement suivant parte d'une ardoise propre.
#
#   network/pcu-socket.sh          etat (defaut)
#   network/pcu-socket.sh --fix    supprime les sockets orphelins /tmp/pcu_bts*
#
# Appele par start-native.sh juste avant le lancement de run.sh (--fix).
# ============================================================================
set -u

PCU_SOCKET="${PCU_SOCKET:-/tmp/pcu_bts}"
PCU_LOG="${PCU_LOG:-/var/log/osmocom/osmo-pcu.log}"
MODE=check
[ "${1:-}" = "--fix" ] && MODE=fix

# Un socket est "en service" si un processus l'ecoute (ss -xlp le montre).
listener_of() {
    ss -xlp 2>/dev/null | grep -F " $1 " | sed -n 's/.*users:((\"\([^\"]*\)\",pid=\([0-9]*\).*/\1 pid \2/p' | head
    ↪  -1
}

rc=0
echo "pcu-socket : socket attendu par le PCU = $PCU_SOCKET"

for s in /tmp/pcu_bts /tmp/pcu_bts2; do
    if [ ! -e "$s" ]; then
        printf '  %-16s absent\n' "$s"
        continue
    fi
    who="$(listener_of "$s")"
    if [ -n "$who" ]; then
        printf '  %-16s en service (%s)\n' "$s" "$who"
    elif [ "$MODE" = fix ]; then
        if rm -f "$s"; then
            printf '  %-16s orphelin supprime\n' "$s"
        else
            printf '  %-16s orphelin, suppression impossible\n' "$s"; rc=1
        fi
    else
        printf '  %-16s orphelin (aucun processus) - lancer avec --fix\n' "$s"
    fi
done

# Qui sert le PCU : la derniere connexion journalisee vers le socket attendu.
if [ -r "$PCU_LOG" ]; then
    last="$(sed 's/\x1b\[[0-9;]*m//g' "$PCU_LOG" | grep -a "PCU socket $PCU_SOCKET has been connected" | tail -1)"
    if [ -n "$last" ]; then
        echo "  PCU          connecte a $PCU_SOCKET  ($(echo "$last" | awk '{print $4, $5, $6}'))"
    else
        echo "  PCU          pas de connexion journalisee a $PCU_SOCKET"
```

```
    fi
fi

# Config : quels BTS declarent un socket PCU, et lequel.
for c in /etc/osmocom/osmo-bts*.cfg; do
    [ -r "$c" ] || continue
    line="$(grep -E '^[[:space:]]*pcu-socket[[:space:]]' "$c" | head -1 | sed 's/^[[:space:]]*//')"
    printf '  %-28s %s\n' "$(basename "$c")" "${line:-(defaut : /tmp/pcu_bts)}"
done

exit $rc
```

3.251 /opt/GSM/osmo-operator/network/set-node-id.sh

16687 octets, 322 lignes → 322 lignes

```
#!/bin/bash
# ═══════════════════════════════════════════════════════════════════════
# network/set-node-id.sh - donne a CETTE machine son identite de noeud SS7
#
# Une seule ISO pour tous les noeuds : le numero se choisit au demarrage, pas a
# la construction. Ce script reecrit ce qui, dans les configs Osmocom, designe
# le noeud - c'est-a-dire ses POINT CODES et ses routing contexts.
#
#   plan local (une machine)   1.<op>.<role>          op×100 + {10,20,30,50}
#   plan WAN   (N machines)    1.<noeud><op>.<role>    noeud×1000 + op×100 + ...
#
# Pourquoi ca ne peut pas rester le plan local : la formule se recalcule a
# l'identique sur chaque machine. Trois noeuds attaches au meme inter-STP y
# presenteraient trois fois 1.11.2. Un point code est une ADRESSE - deux
# equipements qui partagent la leur, ce n'est pas un doublon de nom, c'est du
# routage faux, et le hub n'a aucun moyen de s'en apercevoir.
#
# Trois fichiers portent cette identite, et ils doivent rester d'accord :
#   osmo-stp.cfg   son propre PC, son routing-key, son ASP vers le hub
#   osmo-msc.cfg   son PC, son routing-key, et le PC du BSC qu'il appelle
#   osmo-bsc.cfg   son PC, son routing-key, et le PC du MSC qu'il appelle
# En rater un donne un coeur qui demarre et des appels qui n'aboutissent pas.
#
# Idempotent : il lit l'etat courant et le remplace. Passer --node 2 apres
# --node 1 fonctionne, il n'y a pas d'etat a nettoyer entre les deux.
#
# Usage :
#   sudo network/set-node-id.sh --node 3 [--op 1] [--hub-ip 192.168.1.49]
#   sudo network/set-node-id.sh --show          etat actuel, sans rien modifier
#   sudo network/set-node-id.sh --node 3 --local   plan local (annule le plan WAN)
# ═══════════════════════════════════════════════════════════════════════
set -euo pipefail

RED='\033[0;31m'; GREEN='\033[0;32m'; YELLOW='\033[1;33m'
CYAN='\033[0;36m'; BOLD='\033[1m'; NC='\033[0m'

SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
# wan_detect_local_ip vit ici : c'est la meme regle de choix d'adresse que celle
# qu'appliquent start-docker.sh et start-native.sh. La dupliquer, c'est se garantir
# qu'un jour les deux ne designeront plus la meme carte.
# shellcheck source=network/wan-nodes.sh
. "$SCRIPT_DIR/wan-nodes.sh"

NODE=""
OP=1
HUB_IP=""
MODE=""            # docker | native - l'adressage n'est pas le meme
LOCAL_IP=""        # adresse source de l'ASP ; "auto" = detectee
```

```
CONF_DIR="${CONF_DIR:-/etc/osmocom}"
SHOW=0
LOCAL_PLAN=0
DRY=0

while [ $# -gt 0 ]; do
    case "$1" in
        --node)     NODE="${2:-}"; shift ;;
        --node=*)   NODE="${1#*=}" ;;
        --op)       OP="${2:-1}"; shift ;;
        --op=*)     OP="${1#*=}" ;;
        --local-ip)   LOCAL_IP="${2:-}"; shift ;;
        --local-ip=*) LOCAL_IP="${1#*=}" ;;
        --hub-ip)   HUB_IP="${2:-}"; shift ;;
        --hub-ip=*) HUB_IP="${1#*=}" ;;
        --conf-dir)   CONF_DIR="${2:-}"; shift ;;
        --conf-dir=*) CONF_DIR="${1#*=}" ;;
        --mode)     MODE="${2:-}"; shift ;;
        --mode=*)   MODE="${1#*=}" ;;
        --docker)   MODE="docker" ;;
        --native)   MODE="native" ;;
        --show)     SHOW=1 ;;
        --local)    LOCAL_PLAN=1 ;;
        --dry-run)  DRY=1 ;;
        -h|--help)  sed -n '2,32p' "$0" | sed 's/^# \{0,1\}//'; exit 0 ;;
        *) echo -e "${RED}option inconnue : $1${NC}" >&2; exit 2 ;;
    esac
    shift
done

STP="$CONF_DIR/osmo-stp.cfg"
MSC="$CONF_DIR/osmo-msc.cfg"
BSC="$CONF_DIR/osmo-bsc.cfg"

# ── Lecture de l'identite en place ──────────────────────────────
# Le premier point-code d'un fichier est CELUI DU DEMON ; ceux qui suivent, a
# l'interieur d'un bloc "sccp-address", designent ses pairs. C'est cette
# distinction qui structure toute la reecriture.
own_pc() { awk '/^cs7 instance/{c=1} c && /^ *point-code /{print $2; exit}' "$1" 2>/dev/null; }

show_state() {
    local f n
    printf '  %-16s %-10s %-12s %-16s %s\n' FICHIER "POINT CODE" "ROUTING-KEY" "ASP INTER-STP" "SOURCE"
    for f in "$STP" "$MSC" "$BSC"; do
        [ -r "$f" ] || { printf '  %-16s %s\n' "$(basename "$f")" "absent"; continue; }
        local pc rk asp
        pc="$(own_pc "$f")"
        rk="$(awk '/^ *routing-key /{print $2; exit}' "$f")"
        asp="$(awk '/asp asp-to-inter/{f=1} f&&/remote-ip/{print $2; exit}' "$f")"
        local src
        src="$(awk '/asp asp-to-inter/{f=1} f&&/local-ip/{print $2; exit}' "$f")"
        printf '  %-16s %-10s %-12s %-16s %s\n' "$(basename "$f")" "${pc:--}" "${rk:--}" "${asp:--}" "${src:--}"
    done
}

if [ "$SHOW" = "1" ]; then
    [ -n "$MODE" ] || { [ -f /.dockerenv ] && MODE=docker || MODE=native; }
    echo -e "${BOLD}Identite SS7 de cette machine${NC}  (mode ${CYAN}${MODE}${NC})"
    show_state
    exit 0
fi

[[ "$NODE" =~ ^[1-9]$ ]] || { echo -e "${RED}--node : un chiffre de 1 a 9${NC}" >&2; exit 2; }
[[ "$OP"   =~ ^[1-9]$ ]] || { echo -e "${RED}--op : un chiffre de 1 a 9${NC}" >&2; exit 2; }
[ "$(id -u)" -eq 0 ] || { echo -e "${RED}Root requis (ecriture dans ${CONF_DIR}).${NC}" >&2; exit 1; }
```

```
for f in "$STP" "$MSC" "$BSC"; do
    [ -w "$f" ] || { echo -e "${RED}$f introuvable ou non modifiable${NC}" >&2; exit 1; }
done

# ── Le nouveau plan ──────────────────────────────────────────────────
if [ "$LOCAL_PLAN" = "1" ]; then
    MID="$OP"                       # 1.<op>.<role> - plan d'une machine isolee
    RBASE=$(( OP * 100 ))
else
    MID="${NODE}${OP}"              # 1.<noeud><op>.<role> - plan WAN
    RBASE=$(( NODE * 1000 + OP * 100 ))
fi
PC_MSC="1.${MID}.1"; PC_STP="1.${MID}.2"; PC_BSC="1.${MID}.3"
RCTX_MSC=$(( RBASE + 10 )); RCTX_STP=$(( RBASE + 20 ))
RCTX_BSC=$(( RBASE + 30 )); RCTX_INTER=$(( RBASE + 50 ))

# ── DOCKER OU VM : deux plans d'adressage, et le defaut de l'un est faux dans
# l'autre ────────────────────────────────────────────────────────────
#   docker : le hub est le conteneur osmo-inter-stp, 172.20.0.10, sur le reseau
#            gsm-inter. Le noeud y a deja son IP quand osmo-stp demarre.
#   VM/ISO : le hub est une machine a part, sur le LAN en acces par pont.
#            L'adresse du noeud vient du DHCP et n'est pas forcement montee au
#            lancement.
# Se tromper de defaut donne un ASP qui ne s'attache jamais, et le journal ne
# dit que "connection refused" - sans indiquer que l'adresse visee n'existe
# nulle part sur ce reseau-la.
#
# Le host-only VirtualBox (192.168.56.1) a longtemps servi de repli ici. Il est
# FAUX des que les VM sont en acces par pont : l'adresse n'existe sur aucun
# segment, et l'ASP la vise sans jamais rien atteindre. On DEMANDE donc
# l'adresse au lieu de la deviner, avec le hub du banc comme defaut. Une
# reponse VIDE est acceptee quand on ne la connait pas encore : dans ce cas le
# remote-ip deja present dans le fichier n'est pas touche - mieux vaut ne rien
# ecrire qu'ecrire une adresse fausse.
if [ -z "$MODE" ]; then
    if [ -f /.dockerenv ]; then MODE="docker"; else MODE="native"; fi
fi

: "${HUB_IP_DEFAULT:=172.20.0.10}"    # le hub du banc : osmo-inter-stp, backbone docker

if [ -z "$HUB_IP" ]; then
    if [ -r /etc/osmo-role ]; then
        HUB_IP="$(awk -F= '/^OSMO_HUB_IP=/{print $2}' /etc/osmo-role)"
    fi
fi
if [ -z "$HUB_IP" ]; then
    case "$MODE" in
        docker) HUB_IP="${INTER_STP_IP:-172.20.0.10}" ;;
        *)
            # Interactif : on demande. Sinon (ISO, service, CI) on prend le
            # defaut du banc - un script non interactif ne peut pas repondre,
            # et rester bloque sur une lecture ne ferait qu'echouer plus tard.
            if [ -t 0 ] && [ "$DRY" != "1" ] && command -v _wan_ask >/dev/null 2>&1; then
                HUB_IP="$(_wan_ask "Inter-STP" \
                    "IP de l'inter-STP (vide si inconnue) :" "$HUB_IP_DEFAULT")" || exit 1
            else
                HUB_IP="$HUB_IP_DEFAULT"
            fi ;;
    esac
fi

# Vide assume : on ne reecrira pas le remote-ip. Le reste (point codes,
# routing-keys, local-ip) garde tout son sens et s'applique quand meme.
SET_HUB_IP=1
if [ -z "$HUB_IP" ]; then
    SET_HUB_IP=0
```

```
    echo -e "  ${YELLOW}Inter-STP inconnu : remote-ip laisse tel quel dans les fichiers.${NC}"
    echo -e "  ${YELLOW}Renseignez-le plus tard : --hub-ip ADRESSE, ou set_stp_ip.sh${NC}"
fi

# ── Adresse SOURCE de l'ASP vers le hub ──────────────────────────────
# En docker on n'y touche pas : le conteneur a son adresse avant que le demon ne
# demarre, et une source explicite evite que SCTP annonce plusieurs adresses
# (multi-homing) a un pair qui n'en attend qu'une.
#
# En VM, on INJECTE l'adresse reellement detectee sur le segment VirtualBox
# plutot que 0.0.0.0. La difference n'est pas cosmetique : avec 0.0.0.0, SCTP
# annonce TOUTES les adresses locales dans son INIT - y compris le NAT 10.0.2.15
# et les alias 172.20.x que l'ISO pose. Le hub tente alors de joindre des
# adresses qui, vues de lui, ne menent nulle part. Une adresse unique et juste
# vaut mieux qu'un multi-homing dont trois branches sur quatre sont mortes.
#
# On retombe sur 0.0.0.0 si aucune adresse n'est detectable - s'y lier vaut
# mieux que de se lier a une adresse absente, qui empeche osmo-stp de demarrer.
SET_LOCAL_IP=0
ASP_LOCAL_IP=""
if [ "$MODE" != "docker" ]; then
    SET_LOCAL_IP=1
    case "$LOCAL_IP" in
        "" | auto)
            # ── L'adresse qui ROUTE vers le hub, avant toute heuristique ────
            # Le noyau sait exactement quelle source il utilisera pour joindre
            # le hub : c'est ce que "ip route get" repond. Toute autre methode
            # devine - et devine mal des que la machine porte plusieurs
            # adresses privees. Une VM du banc en a trois (192.168.1.2 le pont,
            # 192.168.2.1 et .2.10 le plan prive de l'ISO) : l'heuristique
            # choisissait 192.168.2.1, une adresse que le hub ne peut pas
            # joindre en retour. L'ASP partait alors dans le vide et le journal
            # ne disait que "connect failed (-110)" - un timeout, sans indice
            # sur la source fautive.
            if [ -n "$HUB_IP" ]; then
                ASP_LOCAL_IP="$(ip route get "$HUB_IP" 2>/dev/null \
                                | sed -n 's/.*src \([0-9.]*\).*/\1/p' | head -1)"
                [ -n "$ASP_LOCAL_IP" ] && LOCAL_IP_SRC="route vers le hub"
            fi
            if [ -n "$ASP_LOCAL_IP" ]; then
                :
            elif ! command -v wan_detect_local_ip >/dev/null 2>&1; then
                echo -e "${RED}wan_detect_local_ip introuvable - network/wan-nodes.sh non charge${NC}" >&2
                exit 1
            else
                ASP_LOCAL_IP="$(wan_detect_local_ip || true)"
            fi
            if [ -n "$ASP_LOCAL_IP" ]; then
                [ -n "${LOCAL_IP_SRC:-}" ] || LOCAL_IP_SRC="detectee sur le segment"
            else
                ASP_LOCAL_IP="0.0.0.0"
                LOCAL_IP_SRC="aucune adresse detectee - repli"
            fi ;;
        *)  ASP_LOCAL_IP="$LOCAL_IP"; LOCAL_IP_SRC="imposee par --local-ip" ;;
    esac
fi

echo -e "${BOLD}Noeud ${NODE} - operateur ${OP}${NC}   (mode ${CYAN}${MODE}${NC})"
echo -e "  STP ${CYAN}${PC_STP}${NC}   MSC ${CYAN}${PC_MSC}${NC}   BSC ${CYAN}${PC_BSC}${NC}"
echo -e "  routing-key : stp ${RCTX_STP} · msc ${RCTX_MSC} · bsc ${RCTX_BSC} · inter ${RCTX_INTER}"
if [ "$LOCAL_PLAN" = "1" ] && [ -z "$HUB_IP" ]; then
    echo -e "  ${YELLOW}plan LOCAL sans hub : point codes 1.<op>.<role>, ASP inchange${NC}"
else
    [ "$LOCAL_PLAN" = "1" ] && echo -e "  ${CYAN}plan LOCAL${NC} (point codes 1.<op>.<role>) avec hub sur cette
     ↪  machine"
```

```
    echo -e "  ASP inter-STP → ${CYAN}${HUB_IP}${NC}$([ "$SET_LOCAL_IP" = "1" ] && echo "  depuis
     ↪  ${CYAN}${ASP_LOCAL_IP}${NC} (${LOCAL_IP_SRC})" || echo "  (source inchangee)")"
fi
echo ""

# ── Reecriture ──────────────────────────────────────────────────────
# `own` = le point code du demon (au niveau cs7 instance).
# `peer` = celui qu'il declare dans ses blocs sccp-address.
# La distinction se fait sur le bloc courant, pas sur le numero de ligne : un
# fichier qui gagne une directive ne decale pas la reecriture.
rewrite() {  # $1=fichier  $2=own  $3=peer  $4=rctx  $5=1 si bloc asp inter-STP
    local f="$1" own="$2" peer="$3" rctx="$4" do_asp="$5"
    local tmp; tmp="$(mktemp)"
    awk -v own="$own" -v peer="$peer" -v rctx="$rctx" -v do_asp="$do_asp" -v hub="$HUB_IP" \
        -v set_hub="$SET_HUB_IP" \
        -v set_local="$SET_LOCAL_IP" -v localip="${ASP_LOCAL_IP:-0.0.0.0}" '
        /^cs7 instance/            { in_cs7=1 }
        /^ *sccp-address /         { sccp=1 }
        /^ *(as|asp|listen|route-table|network-indicator|xua) / { sccp=0 }
        /^!/                       { sccp=0 }
        /^ *asp asp-to-inter/      { in_asp=1; print; next }
        in_asp && /^ *(as|listen|route-table|!)/ { in_asp=0 }

        in_asp && do_asp && set_hub && /^ *remote-ip /   { sub(/remote-ip .*/, "remote-ip " hub); print; next }
        # local-ip de l ASP : l adresse SOURCE de la SCTP vers le hub. Elle vaut
        # 127.0.0.1 dans l image (tout etait local) ; la laisser ainsi ferait
        # partir la connexion de la boucle locale vers une adresse de segment -
        # elle n aboutirait jamais, et le seul symptome serait un ASP DOWN.
        # 0.0.0.0 : le noyau choisit la source selon la route.
        in_asp && do_asp && set_local && /^ *local-ip / { sub(/local-ip.*/, "local-ip " localip); print; next }
        in_asp && do_asp && /^ *shutdown[ \t]*$/ { sub(/shutdown/, "no shutdown"); print; next }

        /^ *point-code / {
            if (sccp && peer != "") { sub(/point-code .*/, "point-code " peer) }
            else if (in_cs7)        { sub(/point-code .*/, "point-code " own) }
            print; next
        }
        /^ *routing-key / { sub(/routing-key .*/, "routing-key " rctx " " own); print; next }
        { print }
    ' "$f" > "$tmp"

    if [ "$DRY" = "1" ]; then
        # diff renvoie 1 quand les fichiers different : c'est le cas normal ici,
        # et pipefail ferait tomber le script des la premiere reecriture.
        echo "  [dry-run] $f"; { diff -u "$f" "$tmp" || true; } | sed -n '4,30p' | sed 's/^/    /'; rm -f "$tmp"
    else
        # cat plutot que mv : sur l'ISO comme dans un conteneur, ces fichiers
        # peuvent etre des points de montage, et rename() y echoue.
        cat "$tmp" > "$f"; rm -f "$tmp"
        echo -e "  ${GREEN}✓${NC} $(basename "$f")"
    fi
}

# ── LE PLAN DE POINT CODES ET LE HUB SONT DEUX CHOSES ──────────────
# [2026-08-31] do_asp valait 0 des que LOCAL_PLAN=1 : le plan local coupait
# TOUT traitement du bloc asp-to-inter. C etait vrai tant que "plan local"
# voulait dire "machine isolee, aucun hub". Ce n est plus le cas : le banc
# multi-operateur porte son inter-STP dans un CONTENEUR, sur la meme machine,
# joignable en 127.0.0.1:2908 (port SCTP publie par start-docker.sh). Le plan des
# point codes y est bien local - 1.<op>.<role> - ET il y a un hub.
# Consequence du couplage : `--local --hub-ip 127.0.0.1` affichait
# "plan LOCAL : cette machine ne peut pas partager un hub avec une autre" et
# laissait le local-ip du gabarit, 172.20.0.11 - l adresse du PREMIER
# CONTENEUR, inexistante sur l hote. L ASP restait ASP_DOWN sans qu une seule
# ligne ne parle d adresse : le journal ne montre qu un timeout.
# On traite donc l ASP des qu un hub est CONFIGURE, quel que soit le plan.
```

```
_DO_ASP=0
[ -n "$HUB_IP" ] && _DO_ASP=1
[ "$LOCAL_PLAN" = "1" ] || _DO_ASP=1
rewrite "$STP" "$PC_STP" ""        "$RCTX_INTER" "$_DO_ASP"
rewrite "$MSC" "$PC_MSC" "$PC_BSC" "$RCTX_MSC"   0
rewrite "$BSC" "$PC_BSC" "$PC_MSC" "$RCTX_BSC"   0

if [ "$DRY" != "1" ]; then
    echo ""
    show_state
    echo ""
    echo -e "  ${YELLOW}Les demons deja lances gardent l'ancienne identite :${NC} relancez la pile."
fi
```

3.252 /opt/GSM/osmo-operator/network/set_ip_vm.sh

23105 octets, 438 lignes → 438 lignes

```
#!/bin/bash
# ============================================================================
# network/set_ip_vm.sh - rend une VM capable de joindre les conteneurs docker
#
# Le probleme, en une ligne : l'ISO pose sur le NIC des adresses du plan docker
# (172.20.0.1/24, 172.20.0.11/24, 172.20.1.10/24) heritees du temps ou tout
# tournait sur une seule machine. Sur un banc ou les conteneurs vivent sur un
# HOTE a part, ces adresses mentent :
#
#   1. le /24 fait croire que 172.20.0.0/24 est sur le lien local. La VM l'ARP
#      donc sur le LAN au lieu de le router, et
#          ip route add 172.20.0.0/24 via <hote>
#      est refuse d'un "RTNETLINK answers: File exists".
#   2. 172.20.0.11 est l'adresse du PREMIER CONTENEUR operateur. La VM se
#      repond a elle-meme : la panne la plus deroutante du lot, puisque tout
#      "fonctionne" en local et que rien n'arrive jamais au conteneur.
#
# Ce script retire ce qui ment, garde ce qui sert, et pose la route. Tout est
# deduit a l'execution - interface, adresses presentes, reseaux - rien n'est
# ecrit en dur : la meme commande vaut sur les trois VM du banc.
#
# CE QU'IL NE FAIT PAS : casser une pile qui tourne. Une adresse sur laquelle un
# demon ecoute est signalee et CONSERVEE, sauf --force. Mieux vaut un routage
# incomplet qu'un osmo-stp qui ne redemarre plus.
#
# Usage :
#   sudo network/set_ip_vm.sh --via 192.168.1.101      l'hote qui porte docker
#   sudo network/set_ip_vm.sh                          --via demande, ou relu
#   sudo network/set_ip_vm.sh --status                 etat, sans rien changer
#   sudo network/set_ip_vm.sh --undo                   retire la route posee
#
#   --via ADRESSE      hote docker (routeur vers le backbone)
#   --docker-net CIDR  backbone des conteneurs (defaut 172.20.0.0/24)
#   --iface NOM        interface (defaut : celle de la route par defaut)
#   --keep ADRESSE     adresse a NE PAS retirer (repetable)
#   --force            retire meme les adresses sur lesquelles un demon ecoute
#   --hub-ip ADRESSE   ecrit l'inter-STP a viser dans osmo-stp.cfg, leve le
#                      shutdown de l'ASP, puis aligne la source sur la route
#                      qui y mene. Sans elle, on ne fait qu'aligner la source.
#   --no-stp           ne touche pas a osmo-stp.cfg (source de l'ASP)
#   --defaut           applique la configuration du banc sans rien demander.
#                      L'hote docker n'est PAS suppose : il est cherche sur le
#                      reseau lu sur l'interface - la seule machine qui sache
#                      joindre le backbone des conteneurs. --via le fixe.
#   --persist          rejoue au boot (unite systemd)
#   --dry-run          montre, n'applique pas
```

```
# ============================================================================
set -uo pipefail

RED='\033[0;31m'; GREEN='\033[0;32m'; YELLOW='\033[1;33m'
CYAN='\033[0;36m'; BOLD='\033[1m'; NC='\033[0m'

DOCKER_NET="172.20.0.0/24"
VIA=""
IFACE=""
FORCE=0
NO_STP=0
# --hub-ip : l'inter-STP que cet ASP doit viser. Sans elle, on ne fait
# qu'ALIGNER la source sur le remote-ip deja ecrit - ce qui ne vaut rien quand
# ce remote-ip est reste le defaut du gabarit (127.0.0.1) : on validerait une
# configuration qui ne joindra jamais personne. Avec elle, on ECRIT la
# destination, puis la source qui y mene.
HUB_IP=""
USE_DEFAULTS=0
# ── La configuration du banc, en un mot ──────────────────────────────────
# --defaut applique ce qui est vrai sur toutes nos VM sans rien demander :
# l'hote qui porte docker, le backbone des conteneurs, et la persistance au
# boot (le live n'en a aucune, la route serait perdue au redemarrage).
# Chaque valeur reste surchargeable : --defaut --via X gagne sur le defaut.
DEFAULT_VIA=""                  # decouvert : voir discover_docker_host()
DEFAULT_DOCKER_NET="172.20.0.0/24"
DRY=0
PERSIST=0
ACTION="apply"
KEEP=()
STATE=/etc/osmo-docker-host          # memoire du --via, pour ne le taper qu'une fois

while [ $# -gt 0 ]; do
    case "$1" in
        --via)          VIA="${2:-}"; shift ;;
        --via=*)        VIA="${1#*=}" ;;
        --docker-net)   DOCKER_NET="${2:-}"; shift ;;
        --docker-net=*) DOCKER_NET="${1#*=}" ;;
        --iface)        IFACE="${2:-}"; shift ;;
        --iface=*)      IFACE="${1#*=}" ;;
        --keep)         KEEP+=("${2:-}"); shift ;;
        --keep=*)       KEEP+=("${1#*=}") ;;
        --force)        FORCE=1 ;;
        --hub-ip)       HUB_IP="${2:-}"; shift ;;
        --hub-ip=*)     HUB_IP="${1#*=}" ;;
        --no-stp)       NO_STP=1 ;;
        --defaut|--default) USE_DEFAULTS=1 ;;
        --persist)      PERSIST=1 ;;
        --dry-run)      DRY=1 ;;
        --status)       ACTION="status" ;;
        --undo)         ACTION="undo" ;;
        -h|--help)      sed -n '2,40p' "$0" | sed 's/^# \{0,1\}//'; exit 0 ;;
        *) echo -e "${RED}option inconnue : $1${NC}" >&2; exit 2 ;;
    esac
    shift
done

run() { if [ "$DRY" = 1 ]; then echo "    [dry-run] $*"; else "$@"; fi; }

# ── Trouver l'hote docker, sans le nommer ────────────────────────────────
# Une adresse en dur ne survit pas au banc : il a suffi que l'hote passe de
# l'ethernet au wifi pour qu'il change d'adresse, et toutes les VM se sont mises
# a router vers une machine qui n'existait plus - aller sans retour, "connect
# failed" partout, sans qu'aucune configuration n'ait bouge.
#
# On le cherche donc par ce qu'il FAIT, pas par ce qu'il s'appelle : c'est la
# seule machine du LAN qui sait joindre le backbone des conteneurs. Pour chaque
```

```
# machine vivante, on pose la route a l'essai et on tente d'atteindre le premier
# conteneur ; celle qui repond est l'hote. La route est retiree si l'essai
# echoue - rien ne subsiste d'une tentative infructueuse.
discover_docker_host() {
    local prefix probe live c saved
    # Cette fonction POSE des routes pour trouver la bonne : elle ne passe pas
    # par run(), donc --dry-run modifiait quand meme la table de routage de la
    # machine. Un mode qui annonce "montre, n'applique pas" doit tenir parole.
    if [ "$DRY" = 1 ]; then
        echo -e "    [dry-run] decouverte sautee : elle pose des routes a l'essai - donnez --via" >&2
        return 1
    fi
    # Et sans root, "ip route replace" echoue silencieusement a chaque candidat :
    # la boucle allait au bout, VIA ressortait vide, et --defaut sortait en
    # "--via manquant" - un message qui designe la mauvaise cause. Le controle
    # de root n'arrive que cinquante lignes plus bas, trop tard pour celui-ci.
    if [ "$(id -u)" -ne 0 ]; then
        echo -e "    decouverte impossible sans root (elle pose des routes a l'essai) - donnez --via" >&2
        return 1
    fi
    # Le reseau se lit sur l'interface, il n'est jamais suppose.
    [ -n "$LAN_IP" ] || return 1
    prefix="${LAN_IP%.*}"
    probe="$(echo "$DOCKER_NET" | cut -d. -f1-3).11"   # le premier conteneur

    echo -e "  ${CYAN}Recherche de l'hote docker sur ${prefix}.0/24...${NC}" >&2
    # 1. qui est vivant ? (en parallele, une seconde au total)
    live="$(for c in $(seq 1 254); do
                [ "${prefix}.${c}" = "$LAN_IP" ] && continue
                ping -c1 -W1 "${prefix}.${c}" >/dev/null 2>&1 && echo "${prefix}.${c}" &
            done; wait)"
    [ -n "$live" ] || return 1

    # 2. lequel route vers les conteneurs ?
    saved="$(ip -4 route show "$DOCKER_NET" 2>/dev/null | head -1)"
    for c in $live; do
        ip route replace "$DOCKER_NET" via "$c" 2>/dev/null || continue
        if ping -c1 -W1 "$probe" >/dev/null 2>&1; then
            echo -e "  ${GREEN}✓${NC} hote docker : ${CYAN}${c}${NC} (il joint ${probe})" >&2
            printf '%s' "$c"; return 0
        fi
    done
    # rien trouve : on remet ce qui etait la
    ip route del "$DOCKER_NET" 2>/dev/null || true
    [ -n "$saved" ] && ip route add $saved 2>/dev/null || true
    return 1
}

# ── L'interface : celle qui porte la route par defaut ──────────────────
[ -n "$IFACE" ] || IFACE="$(ip route show default 2>/dev/null | awk '/^default/{print $5; exit}')"
[ -n "$IFACE" ] || { echo -e "${RED}Aucune interface par defaut - precisez --iface${NC}" >&2; exit 1; }

# ── L'adresse de la machine SUR LE LAN ─────────────────────────────────
# Prise sur la route par defaut, pas dans la liste des adresses : depuis que
# l'ISO pose aussi le plan prive (192.168.<n>.1/32 et .10/32), un simple "la
# premiere qui n'est pas 172.20" tombait sur 192.168.2.1 - une adresse locale,
# sur aucun segment. La decouverte de l'hote balayait alors le mauvais reseau.
# La source de la route par defaut, elle, est par construction celle qui sort.
LAN_IP="$(ip route show default 2>/dev/null | awk '/^default/{print $9; exit}')"
[ -n "$LAN_IP" ] || LAN_IP="$(ip -4 -o addr show dev "$IFACE" scope global 2>/dev/null \
          | awk '$0 !~ /\/32/ {print $4}' | cut -d/ -f1 | grep -vE '^172\.20\.' | head -1)"

# ── --defaut : la table du banc, sans rien demander ────────────────────
# Ce bloc vient APRES la lecture de l'interface et de l'adresse LAN, et il doit
# y rester : discover_docker_host balaye le reseau lu sur l'interface. Tant
# qu'il etait place avant, LAN_IP n'existait pas encore et set -u tuait le
```

```
# sous-shell de la substitution - la recherche ne demarrait donc jamais, VIA
# restait vide, et --defaut ressortait sur "--via manquant" (code 2) alors que
# son role est justement de ne rien avoir a demander.
if [ "$USE_DEFAULTS" = "1" ]; then
    [ -n "$VIA" ] || VIA="$(awk -F= '/^OSMO_DOCKER_HOST=/{print $2}' "$STATE" 2>/dev/null | tr -d ' \r')"
    [ -n "$VIA" ] || VIA="$(discover_docker_host || true)"
    [ "$DOCKER_NET" = "172.20.0.0/24" ] && DOCKER_NET="$DEFAULT_DOCKER_NET"
    PERSIST=1
    echo -e "${CYAN}  --defaut : hote docker ${VIA}, backbone ${DOCKER_NET}, rejoue au boot${NC}"
fi

# Le prefixe du plan docker, pour reconnaitre les adresses a retirer :
# 172.20.0.0/24 -> "172.20." (on ratisse le /16, car l'ISO pose aussi 172.20.1.x)
DOCKER_PREFIX="$(echo "$DOCKER_NET" | cut -d. -f1-2)."

echo -e "${CYAN}${BOLD}== Adressage VM vers les conteneurs docker ==${NC}"
echo -e "  interface : ${CYAN}${IFACE}${NC}   adresse LAN : ${CYAN}${LAN_IP:-?}${NC}"
echo -e "  backbone  : ${CYAN}${DOCKER_NET}${NC}"

# -- Ce qui traine ------------------------------------------------------
mapfile -t STALE_ADDRS < <(ip -4 -o addr show dev "$IFACE" 2>/dev/null \
    | awk '{print $4}' | grep "^${DOCKER_PREFIX}" || true)
mapfile -t STALE_ROUTES < <(ip -4 route show dev "$IFACE" 2>/dev/null \
    | awk '$1 ~ /^'"${DOCKER_PREFIX//./\\.}"'/ && /scope link/ {print $1}' || true)

# -- --status -----------------------------------------------------------
if [ "$ACTION" = "status" ]; then
    echo ""
    if [ "${#STALE_ADDRS[@]}" -eq 0 ]; then
        echo -e "  ${GREEN}✓${NC} aucune adresse ${DOCKER_PREFIX}x sur ${IFACE}"
    else
        echo -e "  ${YELLOW}adresses du plan docker encore posees :${NC}"
        for a in "${STALE_ADDRS[@]}"; do echo "    $a"; done
    fi
    cur="$(ip -4 route show "$DOCKER_NET" 2>/dev/null | head -1)"
    if echo "$cur" | grep -q " via "; then
        echo -e "  ${GREEN}✓${NC} route : ${cur}"
    elif [ -n "$cur" ]; then
        echo -e "  ${RED}✗${NC} route CONNECTEE (pas de via) : ${cur}"
        echo -e "      tant qu'elle existe, les conteneurs sont injoignables."
    else
        echo -e "  ${YELLOW}—${NC} aucune route vers ${DOCKER_NET}"
    fi
    exit 0
fi

[ "$DRY" = 1 ] || [ "$(id -u)" -eq 0 ] \
    || { echo -e "${RED}Root requis.${NC}" >&2; exit 1; }

# -- --undo -------------------------------------------------------------
if [ "$ACTION" = "undo" ]; then
    run ip route del "$DOCKER_NET" 2>/dev/null \
        && echo -e "  ${GREEN}✓${NC} route retiree" \
        || echo -e "  ${YELLOW}—${NC} pas de route a retirer"
    rm -f /etc/systemd/system/osmo-docker-route.service 2>/dev/null && \
        systemctl disable osmo-docker-route.service >/dev/null 2>&1
    exit 0
fi

# -- L'hote docker ------------------------------------------------------
# Memorise : on ne le tape qu'une fois par machine. Sans terminal et sans
# memoire, on s'arrete plutot que d'inventer une passerelle.
[ -n "$VIA" ] || VIA="$(awk -F= '/^OSMO_DOCKER_HOST=/{print $2}' "$STATE" 2>/dev/null | tr -d ' \r')"
if [ -z "$VIA" ] && [ -t 0 ]; then
    def=""
    [ -n "$LAN_IP" ] && def="${LAN_IP%.*}.101"
```

```
    printf '  Adresse de l hote qui porte docker [%s] : ' "${def:-?}"
    read -r VIA || VIA=""
    [ -n "$VIA" ] || VIA="$def"
fi
if [ -z "$VIA" ]; then
    echo -e "  ${RED}✗ --via manquant : j'ignore par ou joindre ${DOCKER_NET}${NC}" >&2
    exit 2
fi
if ! [[ "$VIA" =~ ^[0-9]{1,3}(\.[0-9]{1,3}){3}$ ]]; then
    echo -e "  ${RED}✗ '${VIA}' n'est pas une IPv4${NC}" >&2; exit 2
fi
echo -e "  hote docker : ${CYAN}${VIA}${NC}"

# ── Ce script est pour les VM. Pas pour l'hote qui porte docker ──────────
# Lance sur l'hote, il pose une route vers LUI-MEME et prend le pas sur la route
# connectee du bridge docker. Resultat immediat : "Destination Host Unreachable"
# vers ses propres conteneurs - alors qu'ils tournent tres bien.
#
# Deux signes le trahissent, et un seul suffit :
#   - --via est une adresse locale ;
#   - un bridge porte deja le reseau des conteneurs.
#
# Attention a ne pas confondre les deux facons de "porter" ce reseau :
#   un BRIDGE docker (br-xxxx, docker0)  -> c'est l'hote, on s'arrete ;
#   une carte ordinaire (enp0s3...)      -> ce sont justement les alias perimes
#                                           de l'ISO, et c'est notre travail.
# Un premier jet refusait les deux et bloquait donc sur la VM qu'il devait
# reparer.
_self=0
ip -4 -o addr show scope global 2>/dev/null | awk '{print $4}' | cut -d/ -f1 \
    | grep -qx "$VIA" && _self=1
_dev="$(ip -4 -o route show "$DOCKER_NET" 2>/dev/null | awk '/proto kernel/{print $3; exit}')"
_bridge=""
case "$_dev" in br-*|docker*|virbr*) _bridge="$_dev" ;; esac
if [ "$_self" = 1 ] || [ -n "$_bridge" ]; then
    echo ""
    echo -e "  ${RED}✗ Cette machine EST l'hote docker - rien a faire ici.${NC}" >&2
    [ "$_self" = 1 ] && echo -e "     ${VIA} est une de ses propres adresses." >&2
    [ -n "$_bridge" ] && echo -e "     ${DOCKER_NET} est deja porte par ${_bridge}." >&2
    echo -e "  ${YELLOW}Ce script se lance SUR LES VM${NC}, pour leur apprendre le chemin" >&2
    echo -e "  vers les conteneurs. Cote hote, c'est l'autre script :" >&2
    echo -e "     ${CYAN}sudo network/setup-docker-lan-route.sh --push \"<ip des VM>\"${NC}" >&2
    exit 2
fi
echo ""

# ── 1. Retirer ce qui ment ──────────────────────────────────────────────
# Une adresse sur laquelle un demon ECOUTE n'est pas retiree : la supprimer
# ferait tomber le service qui s'y est lie, et le probleme deviendrait pire que
# celui qu'on corrige.
for a in ${STALE_ADDRS[@]+"${STALE_ADDRS[@]}"}; do
    ip_only="${a%%/*}"
    skip=0
    for k in ${KEEP[@]+"${KEEP[@]}"}; do [ "$k" = "$ip_only" ] && skip=1; done
    if [ "$skip" = 1 ]; then
        echo -e "  ${CYAN}—${NC} ${a} conservee (--keep)"; continue
    fi
    if [ "$FORCE" != 1 ] && ss -Hlnp 2>/dev/null | grep -q "[^0-9]${ip_only}:"; then
        echo -e "  ${YELLOW}⚠${NC} ${a} CONSERVEE : un demon y ecoute"
        echo -e "     (relancez la pile puis --force, ou --keep pour l'assumer)"
        continue
    fi
    # Une adresse qu'une CONFIG nomme, meme si rien n'ecoute encore. C'est le
    # cas au repos : la pile est arretee, aucun socket ouvert, et pourtant
    # osmo-ggsn.cfg dit "gtp bind-ip 172.20.1.10". La retirer donne, au
    # demarrage suivant, un service qui refuse de partir sur une adresse
```

```
    # "introuvable localement" - et le lien avec ce script est loin.
    if [ "$FORCE" != 1 ]; then
        _ref="$(grep -rlF "$ip_only" /etc/osmocom /etc/asterisk 2>/dev/null | head -3)"
        if [ -n "$_ref" ]; then
            echo -e "  ${YELLOW}⚠${NC} ${a} CONSERVEE : nommee dans une configuration"
            printf '     %s\n' $_ref
            echo -e "     (corrigez la config - 127.0.0.1 convient en natif - puis --force)"
            continue
        fi
    fi
    run ip addr del "$a" dev "$IFACE" 2>/dev/null \
        && echo -e "  ${GREEN}✓${NC} ${a} retiree" \
        || echo -e "  ${YELLOW}–${NC} ${a} deja absente"
done

for r in ${STALE_ROUTES[@]+"${STALE_ROUTES[@]}"}; do
    run ip route del "$r" dev "$IFACE" 2>/dev/null \
        && echo -e "  ${GREEN}✓${NC} route connectee ${r} retiree"
done

# ── 2. Poser la route ──────────────────────────────────────────────
# 'replace' plutot que 'add' : idempotent, et il ne bute pas sur un reste.
if run ip route replace "$DOCKER_NET" via "$VIA" dev "$IFACE"; then
    echo -e "  ${GREEN}✓${NC} ${DOCKER_NET} via ${CYAN}${VIA}${NC}"
else
    echo -e "  ${RED}✗${NC} route refusee - une adresse du plan docker subsiste ?"
    echo -e "     ${CYAN}ip -4 addr show dev ${IFACE} | grep ${DOCKER_PREFIX}${NC}"
    exit 1
fi

# ── 2bis. L'ASP doit partir de l'adresse qui ROUTE ─────────────────
# Poser la route ne suffit pas si osmo-stp se lie a une autre adresse. Une VM du
# banc en porte plusieurs - le pont, et le plan prive de l'ISO (192.168.<n>.1 et
# .10) - et l'auto-detection choisissait volontiers une adresse privee. Le hub
# ne peut alors pas repondre : l'association n'aboutit jamais et le journal ne
# dit que "connect failed (-110)", un timeout, sans nommer la source fautive.
#
# On aligne donc la source de l'ASP sur ce que le noyau utilisera reellement
# pour joindre le hub - la meme regle que network/set-node-id.sh.
STP_CFG="${STP_CFG:-/etc/osmocom/osmo-stp.cfg}"
if [ "${NO_STP:-0}" != "1" ] && [ -w "$STP_CFG" ]; then
    # --hub-ip fait autorite ; sinon on lit ce que le fichier declare deja.
    if [ -n "$HUB_IP" ]; then
        _hub="$HUB_IP"
        _cur_hub="$(awk '/asp asp-to-inter/{f=1} f&&/remote-ip/{print $2; exit}' "$STP_CFG")"
        if [ "$_cur_hub" != "$_hub" ]; then
            if [ "$DRY" = 1 ]; then
                echo -e "  [dry-run] ASP remote-ip ${_cur_hub:-?} -> ${_hub}"
            else
                sed -i "/asp asp-to-inter/,/^ *as / s/^\( *remote-ip \).*/\1${_hub}/" "$STP_CFG"
                # Un ASP en shutdown ne s'attache pas, meme bien adresse : on le
                # leve, sinon la correction reste sans effet visible.
                sed -i "/asp asp-to-inter/,/^ *as / s/^\( *\)shutdown[[:space:]]*$/\1no shutdown/" "$STP_CFG"
                echo -e "  ${GREEN}✓${NC} ASP : hub ${CYAN}${_hub}${NC} (etait ${_cur_hub:-aucun})"
            fi
        fi
    else
        _hub="$(awk '/asp asp-to-inter/{f=1} f&&/remote-ip/{print $2; exit}' "$STP_CFG")"
    fi
    # Un remote-ip en boucle locale n'est pas un hub : c'est le defaut du
    # gabarit, laisse par une regeneration qui a efface l'identite du noeud.
    # Aligner la source dessus validerait une configuration morte - "source
    # deja correcte (127.0.0.1)" sur un ASP qui ne joindra jamais personne.
    case "$_hub" in
        127.*|0.0.0.0|"")
            echo -e "  ${YELLOW}⚠${NC} ASP : remote-ip=${_hub:-vide} - ce n'est pas un hub."
```

```
            echo -e "     L'identite SS7 n'a pas ete appliquee : ${CYAN}./start-native.sh --node N --hub-ip
            ↪  <hub>${NC}"
            _hub="" ;;
    esac
    if [ -n "$_hub" ]; then
        _src="$(ip route get "$_hub" 2>/dev/null | sed -n 's/.*src \([0-9.]*\).*/\1/p' | head -1)"
        _cur="$(awk '/asp asp-to-inter/{f=1} f&&/local-ip/{print $2; exit}' "$STP_CFG")"
        if [ -n "$_src" ] && [ "$_src" != "$_cur" ]; then
            if [ "$DRY" = 1 ]; then
                echo -e "  [dry-run] ASP local-ip ${_cur:-?} -> ${_src}"
            else
                sed -i "/asp asp-to-inter/,/^ *as / s/^\( *local-ip \).*/\1${_src}/" "$STP_CFG"
                echo -e "  ${GREEN}✓${NC} ASP : source ${CYAN}${_src}${NC} (etait ${_cur:-aucune}) vers le hub
                ↪  ${_hub}"
                if systemctl is-active osmo-stp >/dev/null 2>&1; then
                    systemctl restart osmo-stp \
                        && echo -e "  ${GREEN}✓${NC} osmo-stp relance pour prendre la nouvelle source"
                fi
            fi
        else
            echo -e "  ${GREEN}✓${NC} ASP : source deja correcte (${_cur:-aucune})"
        fi
    fi
fi

[ "$DRY" = 1 ] || printf 'OSMO_DOCKER_HOST=%s\n' "$VIA" > "$STATE" 2>/dev/null || true

# ── 3. Au boot, si demande ──────────────────────────────────────────
# Un live sans persistance repart de l'ISO a chaque demarrage : sans unite, tout
# est a refaire. Elle rejoue simplement ce script, qui est idempotent.
if [ "$PERSIST" = 1 ]; then
    here="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
    if [ "$DRY" = 1 ]; then
        echo "    [dry-run] unite osmo-docker-route.service"
    else
        cat > /etc/systemd/system/osmo-docker-route.service <<EOF
[Unit]
Description=osmo-operator - route vers les conteneurs docker (${DOCKER_NET})
After=network-online.target
Wants=network-online.target
[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=${here}/set_ip_vm.sh --via ${VIA} --docker-net ${DOCKER_NET}
[Install]
WantedBy=multi-user.target
EOF
        systemctl enable osmo-docker-route.service >/dev/null 2>&1 \
            && echo -e "  ${GREEN}✓${NC} rejoue au boot (osmo-docker-route.service)"
    fi
fi

# ── 4. Le controle qui compte ───────────────────────────────────────
echo ""
first="$(echo "$DOCKER_NET" | cut -d. -f1-3).11"
echo -e "  ${BOLD}Verifier :${NC} ${CYAN}ip route get ${first}${NC}   doit indiquer via ${VIA}"
if [ "$DRY" != 1 ]; then
    got="$(ip route get "$first" 2>/dev/null | head -1)"
    if echo "$got" | grep -q "via ${VIA}"; then
        echo -e "  ${GREEN}✓${NC} ${got}"
    else
        echo -e "  ${YELLOW}⚠${NC} ${got:-aucune reponse}"
    fi
fi
```

3.253 /opt/GSM/osmo-operator/network/setup-docker-lan-route.sh

10146 octets, 195 lignes → 195 lignes

```
#!/bin/bash
# ============================================================================
# network/setup-docker-lan-route.sh - les conteneurs docker vus du LAN, ROUTES
#
# Les operateurs docker vivent sur le backbone 172.20.0.0/24 (osmo-operator-1 en
# .11, -2 en .12, -3 en .13 ; l'inter-STP en .10). Les VM VirtualBox, elles, sont
# en acces par pont sur 192.168.1.0/24 avec le hub SS7 en .49. Les deux mondes
# s'ignorent : rien ne dit au LAN ou trouver 172.20.0.0/24.
#
# CE SCRIPT EN FAIT UN SEUL RESEAU, PAR ROUTAGE
# L'hote porte les deux (192.168.1.101 cote LAN, 172.20.0.1 cote docker) : il lui
# suffit de router. Pas de macvlan, pas de renumerotation, pas de publication de
# ports - les conteneurs gardent leurs adresses, et les VM les joignent telles
# quelles.
#
# POURQUOI PAS SIMPLEMENT -p (publication de ports)
# Une publication NAT presente TOUS les conteneurs sous l'IP de l'hote. Or
# l'inter-STP identifie ses ASP par leur adresse source ("match=" dans les blocs
# type=identify) : trois operateurs derriere une seule IP sont indiscernables, et
# le hub rattache leurs associations au mauvais AS - ou les refuse. Le routage,
# lui, preserve l'adresse d'origine.
#
# LES TROIS CHOSES A FAIRE, ET POURQUOI CHACUNE
#   1. ip_forward       sans lui l'hote ne route rien, il ne fait que recevoir.
#   2. pas de masquage vers le LAN
#                       docker MASQUERADE tout ce qui sort du bridge vers
#                       l'exterieur. Le hub verrait alors 192.168.1.101 partout,
#                       au lieu de 172.20.0.11/12/13 - exactement ce que le
#                       paragraphe ci-dessus dit d'eviter. On insere donc un
#                       RETURN avant la regle de docker, pour la seule
#                       destination 192.168.1.0/24.
#   3. FORWARD autorise docker pose un FORWARD par defaut a DROP. La chaine
#                       DOCKER-USER est faite pour ca : elle est consultee AVANT
#                       les regles de docker, et il ne la reecrit pas.
#
# COTE VM, une route a ajouter (le script l'affiche, et la pose si --push) :
#     ip route add 172.20.0.0/24 via <IP de l hote>
#
# Usage :
#   sudo network/setup-docker-lan-route.sh              applique sur l'hote
#   sudo network/setup-docker-lan-route.sh --push "192.168.1.123 192.168.1.49"
#                                                       + pose la route sur ces VM
#   sudo network/setup-docker-lan-route.sh --status     etat, sans rien changer
#   sudo network/setup-docker-lan-route.sh --undo       retire ce que le script a pose
#
#   --docker-net CIDR   reseau des conteneurs (defaut 172.20.0.0/24)
#   --lan CIDR          reseau du LAN         (defaut : deduit de l'interface)
#   --iface NOM         interface LAN         (defaut : celle de la route par defaut)
# ============================================================================
set -uo pipefail

RED='\033[0;31m'; GREEN='\033[0;32m'; YELLOW='\033[1;33m'
CYAN='\033[0;36m'; BOLD='\033[1m'; NC='\033[0m'

DOCKER_NET="172.20.0.0/24"
LAN_NET=""
IFACE=""
PUSH=""
ACTION="apply"

while [ $# -gt 0 ]; do
    case "$1" in
```

```
        --docker-net)   DOCKER_NET="${2:-}"; shift ;;
        --docker-net=*) DOCKER_NET="${1#*=}" ;;
        --lan)          LAN_NET="${2:-}"; shift ;;
        --lan=*)        LAN_NET="${1#*=}" ;;
        --iface)        IFACE="${2:-}"; shift ;;
        --iface=*)      IFACE="${1#*=}" ;;
        --push)         PUSH="${2:-}"; shift ;;
        --push=*)       PUSH="${1#*=}" ;;
        --status)       ACTION="status" ;;
        --undo)         ACTION="undo" ;;
        -h|--help)      sed -n '2,46p' "$0" | sed 's/^# \{0,1\}//'; exit 0 ;;
        *) echo -e "${RED}option inconnue : $1${NC}" >&2; exit 2 ;;
    esac
    shift
done

# ── Ou sommes-nous ? ──────────────────────────────────────────────────
[ -n "$IFACE" ] || IFACE="$(ip route show default 2>/dev/null | awk '/^default/{print $5; exit}')"
[ -n "$IFACE" ] || { echo -e "${RED}Aucune interface par defaut - precisez --iface${NC}" >&2; exit 1; }

HOST_IP="$(ip -4 -o addr show dev "$IFACE" scope global 2>/dev/null | awk '{print $4}' | cut -d/ -f1 | head -1)"
[ -n "$HOST_IP" ] || { echo -e "${RED}Pas d'adresse sur ${IFACE}${NC}" >&2; exit 1; }

if [ -z "$LAN_NET" ]; then
    CIDR="$(ip -4 -o addr show dev "$IFACE" scope global 2>/dev/null | awk '{print $4}' | head -1)"
    # 192.168.1.101/24 -> 192.168.1.0/24
    LAN_NET="$(python3 - "$CIDR" <<'PY' 2>/dev/null || true
import ipaddress, sys
print(ipaddress.ip_interface(sys.argv[1]).network)
PY
)"
    [ -n "$LAN_NET" ] || LAN_NET="${HOST_IP%.*}.0/24"
fi

echo -e "${CYAN}${BOLD}== Routage docker <-> LAN ==${NC}"
echo -e "  hote      : ${CYAN}${HOST_IP}${NC} sur ${CYAN}${IFACE}${NC}"
echo -e "  LAN       : ${CYAN}${LAN_NET}${NC}"
echo -e "  conteneurs: ${CYAN}${DOCKER_NET}${NC}"
echo ""

# ── --status ──────────────────────────────────────────────────────────
if [ "$ACTION" = "status" ]; then
    fwd="$(sysctl -n net.ipv4.ip_forward 2>/dev/null)"
    printf '  %-34s %s\n' "ip_forward" "$([ "$fwd" = 1 ] && echo "1 (ok)" || echo "${fwd:-?} - le routage est
    ↪  inactif")"
    if iptables -t nat -C POSTROUTING -s "$DOCKER_NET" -d "$LAN_NET" -j RETURN 2>/dev/null; then
        printf '  %-34s %s\n' "pas de masquage vers le LAN" "pose"
    else
        printf '  %-34s %s\n' "pas de masquage vers le LAN" "ABSENT - le hub verrait ${HOST_IP} partout"
    fi
    for spec in "-s ${LAN_NET} -d ${DOCKER_NET}" "-s ${DOCKER_NET} -d ${LAN_NET}"; do
        # shellcheck disable=SC2086
        if iptables -C DOCKER-USER $spec -j ACCEPT 2>/dev/null; then
            printf '  %-34s %s\n' "FORWARD ${spec}" "autorise"
        else
            printf '  %-34s %s\n' "FORWARD ${spec}" "ABSENT"
        fi
    done
    echo ""
    echo -e "  ${BOLD}Sur chaque VM :${NC} ip route add ${DOCKER_NET} via ${HOST_IP}"
    exit 0
fi

[ "$(id -u)" -eq 0 ] || { echo -e "${RED}Root requis.${NC}" >&2; exit 1; }

# ── --undo ────────────────────────────────────────────────────────────
```

```
if [ "$ACTION" = "undo" ]; then
    iptables -t nat -D POSTROUTING -s "$DOCKER_NET" -d "$LAN_NET" -j RETURN 2>/dev/null \
        && echo -e "  ${GREEN}✓${NC} regle de non-masquage retiree"
    iptables -D DOCKER-USER -s "$LAN_NET" -d "$DOCKER_NET" -j ACCEPT 2>/dev/null \
        && echo -e "  ${GREEN}✓${NC} FORWARD LAN -> docker retire"
    iptables -D DOCKER-USER -s "$DOCKER_NET" -d "$LAN_NET" -j ACCEPT 2>/dev/null \
        && echo -e "  ${GREEN}✓${NC} FORWARD docker -> LAN retire"
    echo -e "  ${YELLOW}ip_forward laisse tel quel : d'autres services peuvent en dependre.${NC}"
    exit 0
fi

# ── 1. Le routage ──────────────────────────────────────────────────
if [ "$(sysctl -n net.ipv4.ip_forward 2>/dev/null)" != "1" ]; then
    sysctl -qw net.ipv4.ip_forward=1 && echo -e "  ${GREEN}✓${NC} ip_forward active"
else
    echo -e "  ${GREEN}✓${NC} ip_forward deja actif"
fi

# ── 2. Ne PAS masquer vers le LAN ──────────────────────────────────
# -I : en TETE de POSTROUTING, donc avant le MASQUERADE que docker y a mis. Un
# RETURN rend la main a la chaine appelante sans masquer : le paquet garde son
# adresse source 172.20.0.x, et le hub SS7 reconnait l'operateur qui l'emet.
if iptables -t nat -C POSTROUTING -s "$DOCKER_NET" -d "$LAN_NET" -j RETURN 2>/dev/null; then
    echo -e "  ${GREEN}✓${NC} non-masquage deja en place"
else
    iptables -t nat -I POSTROUTING 1 -s "$DOCKER_NET" -d "$LAN_NET" -j RETURN \
        && echo -e "  ${GREEN}✓${NC} ${DOCKER_NET} -> ${LAN_NET} : adresse source preservee"
fi

# ── 3. Autoriser le FORWARD dans les deux sens ─────────────────────
# DOCKER-USER est consultee avant les regles de docker, et docker ne la reecrit
# pas : c'est le seul endroit ou une regle survit a un redemarrage du daemon.
iptables -N DOCKER-USER 2>/dev/null || true
for spec in "-s ${LAN_NET} -d ${DOCKER_NET}" "-s ${DOCKER_NET} -d ${LAN_NET}"; do
    # shellcheck disable=SC2086
    if iptables -C DOCKER-USER $spec -j ACCEPT 2>/dev/null; then
        echo -e "  ${GREEN}✓${NC} FORWARD deja autorise (${spec})"
    else
        # shellcheck disable=SC2086
        iptables -I DOCKER-USER 1 $spec -j ACCEPT \
            && echo -e "  ${GREEN}✓${NC} FORWARD autorise (${spec})"
    fi
done

# ── 4. La route, cote VM ───────────────────────────────────────────
echo ""
echo -e "  ${BOLD}Il reste UNE chose, sur chaque VM :${NC}"
echo -e "    ${CYAN}ip route add ${DOCKER_NET} via ${HOST_IP}${NC}"
echo -e "  Sans elle, la VM ne sait pas ou envoyer 172.20.0.x et le trafic part"
echo -e "  vers sa passerelle par defaut, qui le jette."

if [ -n "$PUSH" ]; then
    echo ""
    echo -e "  ${BOLD}Pose de la route sur les machines indiquees :${NC}"
    for vm in $PUSH; do
        if ssh -o StrictHostKeyChecking=no -o ConnectTimeout=6 -o BatchMode=yes \
               "root@${vm}" "ip route replace ${DOCKER_NET} via ${HOST_IP}" 2>/dev/null; then
            echo -e "    ${GREEN}✓${NC} ${vm}"
        else
            echo -e "    ${YELLOW}⚠${NC} ${vm} injoignable - a poser a la main"
        fi
    done
fi

echo ""
echo -e "  ${BOLD}Verifier, depuis une VM :${NC} ${CYAN}ping 172.20.0.11${NC}  puis"
```

```
echo -e "  ${CYAN}checks/diag-interstp.sh${NC} cote hub - les ASP doivent apparaitre"
echo -e "  avec leur VRAIE adresse (172.20.0.11/.12/.13), pas celle de l'hote."
```

3.254 /opt/GSM/osmo-operator/network/setup-payload-ism.sh

11123 octets, 221 lignes → 221 lignes

```
#!/bin/bash
# ============================================================================
# network/setup-payload-ism.sh — le PAYLOAD des SMS par radio ISM, le reste par IP
#
# Complement de --wan : la signalisation (MSISDN, IMSI, SC-address, SS7, SIP)
# passe deja par IP entre les deux noeuds ; ce script branche la liaison RADIO
# en bande ISM qui portera le SEUL texte des SMS, et ecrit la section [payload]
# de sms-routing.conf que lit scripts/sms-interop-relay.py (via ism_payload.py).
#
# A lancer sur CHAQUE noeud (avec --wan en place). Choisis le backend selon ta
# radio :
#
#   serial  — un module ISM en UART (LoRa, RFM69, CC1101, HC-12...) sur /dev/tty*.
#             Le plus simple materiellement. Le debit est pose ici par stty.
#               sudo ./network/setup-payload-ism.sh --serial /dev/ttyUSB0 --baud 9600
#
#   exec    — un pipeline SDR que TU fournis (HackRF/LimeSDR/USRP + GNU Radio,
#             rtl_433 en reception, etc.). Deux commandes : TX lit une trame sur
#             son stdin et l'emet ; RX ecrit les trames recues sur son stdout.
#               sudo ./network/setup-payload-ism.sh --exec \
#                    --tx-cmd '/opt/GSM/ism/ism_tx.sh' --rx-cmd '/opt/GSM/ism/ism_rx.sh'
#             Des gabarits de ism_tx.sh / ism_rx.sh sont deposes dans /opt/GSM/ism
#             avec --exemples (ne transmettent rien tant que tu ne les remplis pas).
#
#   fifo    — deux tubes nommes, si tu branches ta radio autrement (socat...).
#               sudo ./network/setup-payload-ism.sh --fifo --tx /tmp/ism_tx --rx /tmp/ism_rx
#
#   loopback— tout en memoire, pour ESSAYER la chaine SMS sans radio (un noeud).
#               sudo ./network/setup-payload-ism.sh --loopback
#
# Options communes : --conf FICHIER (defaut : les sms-routing.conf trouves),
# --recv-timeout SECONDES (attente du payload radio avant echec, defaut 30),
# --selftest (lance l'autotest de ism_payload.py et sort), --off (retire [payload]).
#
# RAPPEL REGLEMENTAIRE : n'emets que sur une bande et une puissance autorisees
# chez toi (ISM 433/868 MHz en Europe, 902-928 MHz en region 2, 2,4 GHz), ou en
# cage/charge. C'est ta responsabilite d'operateur du banc.
# ============================================================================
set -euo pipefail

GREEN='\033[0;32m'; CYAN='\033[0;36m'; YELLOW='\033[1;33m'; RED='\033[0;31m'; NC='\033[0m'
HERE="$(cd "$(dirname "${BASH_SOURCE[0]}")/.." && pwd)"

LINK="" DEV="" BAUD="9600" TX="" RX="" TXCMD="" RXCMD=""
RECV_TIMEOUT="30" CONF="" DO_SELFTEST=0 DO_OFF=0 DO_EXAMPLES=0

while [ $# -gt 0 ]; do
    case "$1" in
        --serial)   LINK=serial; DEV="${2:?--serial attend un /dev/tty*}"; shift ;;
        --baud)     BAUD="${2:?}"; shift ;;
        --exec)     LINK=exec ;;
        --fifo)     LINK=fifo ;;
        --loopback) LINK=loopback ;;
        --tx)       TX="${2:?}"; shift ;;
        --rx)       RX="${2:?}"; shift ;;
        --tx-cmd)   TXCMD="${2:?}"; shift ;;
        --rx-cmd)   RXCMD="${2:?}"; shift ;;
```

```
        --recv-timeout) RECV_TIMEOUT="${2:?}"; shift ;;
        --conf)     CONF="${2:?}"; shift ;;
        --selftest) DO_SELFTEST=1 ;;
        --exemples|--examples) DO_EXAMPLES=1 ;;
        --off)      DO_OFF=1 ;;
        -h|--help)  sed -n '2,/^set -euo/p' "$0" | sed 's/^# \{0,1\}//; $d'; exit 0 ;;
        *) echo "option inconnue : $1" >&2; exit 2 ;;
    esac
    shift
done

say() { echo -e "  ${GREEN}✓${NC} $*"; }
warn() { echo -e "  ${YELLOW}!${NC} $*" >&2; }

# ── les sms-routing.conf a modifier ──────────────────────────────────
# Natif : /etc/osmocom/sms-routing.conf. Docker : celui de chaque container.
confs() {
    if [ -n "$CONF" ]; then printf '%s\n' "$CONF"; return; fi
    local found=0
    if [ -f /etc/osmocom/sms-routing.conf ]; then printf '%s\n' /etc/osmocom/sms-routing.conf; found=1; fi
    if command -v docker >/dev/null 2>&1; then
        for c in $(docker ps --format '{{.Names}}' 2>/dev/null | grep '^osmo-operator-' || true); do
            printf 'docker:%s\n' "$c"; found=1
        done
    fi
    [ "$found" = 1 ] || warn "aucun sms-routing.conf trouve (passe --conf)"
}

# Ecrit la section [payload] dans un fichier (chemin local ou docker:<container>).
write_block() {
    local target="$1" block="$2"
    local script="
        CONF=/etc/osmocom/sms-routing.conf
        sed -i '/# ── PAYLOAD ISM/,/# ── FIN PAYLOAD ISM/d' \"\$CONF\" 2>/dev/null || true
        sed -i '/^\\[payload\\]/,/^\\[/{/^\\[payload\\]/d; /^\\[/!d}' \"\$CONF\" 2>/dev/null || true
        printf '%s\n' \"\$BLOCK\" >> \"\$CONF\"
    "
    if [ "$DO_OFF" = 1 ]; then
        block=""
        script="
            CONF=/etc/osmocom/sms-routing.conf
            sed -i '/# ── PAYLOAD ISM/,/# ── FIN PAYLOAD ISM/d' \"\$CONF\" 2>/dev/null || true
        "
    fi
    case "$target" in
        docker:*)
            local c="${target#docker:}"
            docker exec -e BLOCK="$block" "$c" bash -c "$script" \
                && say "[$c] section [payload] ${DO_OFF:+retiree}${DO_OFF:-ecrite}" \
                || warn "[$c] echec d'ecriture"
            ;;
        *)
            local CONFP="$target"
            if [ "$DO_OFF" = 1 ]; then
                sed -i '/# ── PAYLOAD ISM/,/# ── FIN PAYLOAD ISM/d' "$CONFP" 2>/dev/null || true
                say "[$CONFP] section [payload] retiree"
            else
                sed -i '/# ── PAYLOAD ISM/,/# ── FIN PAYLOAD ISM/d' "$CONFP" 2>/dev/null || true
                printf '%s\n' "$block" >> "$CONFP"
                say "[$CONFP] section [payload] ecrite"
            fi
            ;;
    esac
}

# ── autotest ─────────────────────────────────────────────────────────
```

```
if [ "$DO_SELFTEST" = 1 ]; then
    echo -e "${CYAN}Autotest de la liaison payload ISM${NC}"
    case "$LINK" in
        fifo)   python3 -I "$HERE/scripts/ism_payload.py" --selftest --link fifo \
                    --tx "${TX:-/tmp/osmo_ism_tx}" --rx "${RX:-/tmp/osmo_ism_rx}" ;;
        serial) python3 -I "$HERE/scripts/ism_payload.py" --selftest --link serial --serial "$DEV" ;;
        *)      python3 -I "$HERE/scripts/ism_payload.py" --selftest ;;
    esac
    exit $?
fi

# ── retrait ──────────────────────────────────────────────────────────
if [ "$DO_OFF" = 1 ]; then
    echo -e "${CYAN}Retrait du payload ISM (retour au texte par IP)${NC}"
    while read -r t; do [ -n "$t" ] && write_block "$t" ""; done < <(confs)
    exit 0
fi

[ -n "$LINK" ] || { echo -e "${RED}choisis un backend : --serial /dev/tty* | --exec | --fifo | --loopback${NC}"
 ↪  >&2; exit 2; }

# ── gabarits SDR pour le backend exec ────────────────────────────────
if [ "$DO_EXAMPLES" = 1 ] || { [ "$LINK" = exec ] && [ -z "$TXCMD" ] && [ -z "$RXCMD" ]; }; then
    mkdir -p /opt/GSM/ism
    if [ ! -f /opt/GSM/ism/ism_tx.sh ]; then
        cat > /opt/GSM/ism/ism_tx.sh <<'TX'
#!/bin/bash
# ism_tx.sh — EMET une trame (lue sur stdin) en bande ISM. GABARIT : a remplir.
# La trame est une ligne JSON {"t":jeton,"p":hex,"c":crc}. Emets-la telle quelle
# (modulation FSK/LoRa/OOK au choix) sur ta frequence ISM autorisee.
# Exemples de chaine possible :
#   - module LoRa/UART : cat > /dev/ttyUSB0
#   - SDR GNU Radio    : passer la ligne a un flowgraph (gr-paquet) via un FIFO
# Tant que ce fichier n'emet rien, le payload ISM ne traverse pas : remplis-le.
cat >/dev/null   # <- REMPLACER : ici on jette la trame
TX
        chmod +x /opt/GSM/ism/ism_tx.sh
    fi
    if [ ! -f /opt/GSM/ism/ism_rx.sh ]; then
        cat > /opt/GSM/ism/ism_rx.sh <<'RX'
#!/bin/bash
# ism_rx.sh — RECOIT en bande ISM et ecrit chaque trame recue (une ligne JSON)
# sur stdout. GABARIT : a remplir. Demodule ta radio et recrache les lignes.
# Exemples :
#   - module LoRa/UART : exec cat /dev/ttyUSB0
#   - SDR              : lire la sortie d'un flowgraph (gr-paquet)
# Tant que ce fichier ne lit rien, aucun payload n'arrive : remplis-le.
exec sleep infinity   # <- REMPLACER : ici on ne recoit jamais
RX
        chmod +x /opt/GSM/ism/ism_rx.sh
    fi
    say "gabarits SDR : /opt/GSM/ism/ism_tx.sh et ism_rx.sh (a remplir avant d'emettre)"
    [ -z "$TXCMD" ] && TXCMD="/opt/GSM/ism/ism_tx.sh"
    [ -z "$RXCMD" ] && RXCMD="/opt/GSM/ism/ism_rx.sh"
fi

# ── serie : poser le debit (le backend lit/ecrit ensuite en brut) ────
if [ "$LINK" = serial ]; then
    [ -e "$DEV" ] || warn "$DEV absent pour l'instant (branche le module ISM)"
    if [ -e "$DEV" ]; then
        stty -F "$DEV" "$BAUD" cs8 -cstopb -parenb -echo raw 2>/dev/null \
            && say "serie $DEV a $BAUD bauds, mode brut" \
            || warn "stty sur $DEV a echoue (droits ? module absent ?)"
    fi
fi
```

```
# ── composer la section [payload] ──────────────────────────────────────
BLOCK="# ── PAYLOAD ISM ──────────────────────────────────────────────────────
# Ecrit par network/setup-payload-ism.sh. Le texte des SMS part par radio ISM,
# la signalisation reste sur IP (cf. wiki/Payload-ISM.md).
[payload]
mode = ism
link = $LINK
recv_timeout = $RECV_TIMEOUT"
case "$LINK" in
    serial) BLOCK="$BLOCK
serial = $DEV" ;;
    fifo)   BLOCK="$BLOCK
tx = ${TX:-/tmp/osmo_ism_tx}
rx = ${RX:-/tmp/osmo_ism_rx}" ;;
    exec)   BLOCK="$BLOCK
tx_cmd = $TXCMD
rx_cmd = $RXCMD" ;;
esac
BLOCK="$BLOCK
# ── FIN PAYLOAD ISM ──────────────────────────────────────────────────────"

echo -e "${CYAN}Payload ISM : backend ${LINK}${NC}"
while read -r t; do [ -n "$t" ] && write_block "$t" "$BLOCK"; done < <(confs)

echo
say "Fait. Relance le relais SMS pour qu'il prenne la liaison :"
echo "     - natif  : sudo systemctl restart osmo-smsc  (ou scripts/smsc-start.sh)"
echo "     - docker : sudo docker restart \$(docker ps --format '{{.Names}}' | grep osmo-operator-)"
echo "   ou simplement : ./start-native.sh --wan --payload-ism (il fait tout)."
if [ "$LINK" = exec ]; then
    warn "backend exec : remplis /opt/GSM/ism/ism_tx.sh et ism_rx.sh (ta radio)."
fi
warn "N'emets que sur une bande/puissance ISM autorisees chez toi, ou en charge/cage."
```

3.255 /opt/GSM/osmo-operator/network/setup-vbox-interco.sh

9667 octets, 177 lignes → 177 lignes

```
#!/bin/bash
# ═══════════════════════════════════════════════════════════════════════════
# network/setup-vbox-interco.sh - interconnexion WAN entre CETTE machine et des
# VM VirtualBox, sur un seul PC
#
# Le WAN d'osmo-operator relie des machines par leur IP publique. Pour l'eprouver
# sans louer N serveurs, on remplace les pairs distants par des VM et le lien
# public par un segment host-only : la maquette est la meme, les adresses sont
# locales.
#
#   ┌─ cette machine ──────────┐        ┌─ VM osmo-wan-2 ─┐  ┌─ VM osmo-wan-3 ─┐
#   │ lab docker ou natif      │        │ ISO osmo-operator │  │ ISO osmo-operator  │
#   │ noeud 1 · indicatif 11   │        │ noeud 2 · 22    │  │ noeud 3 · 33    │
#   └──────────┬───────────────┘        └────────┬────────┘  └────────┬────────┘
#              │ vboxnet0 192.168.56.11          │ .12                │ .13
#              └──────────────── segment host-only ──────────────────┘
#
# POURQUOI HOST-ONLY ET PAS "RESEAU INTERNE"
# Un reseau interne relie les VM ENTRE ELLES et laisse l'hote dehors. Or ici
# l'hote est lui-meme un noeud : sans patte sur le segment, son lab ne peut pas
# joindre ses pairs, et rien ne le signale - les trunks restent simplement
# "Unavailable". Le host-only est le seul mode ou l'hote participe.
#
# LA CONTRAINTE A CONNAITRE
# VirtualBox refuse toute plage host-only hors 192.168.56.0/21 tant que
# /etc/vbox/networks.conf ne l'autorise pas. D'ou ce prefixe par defaut.
```

```
#
# Usage :
#   sudo network/setup-vbox-interco.sh [--nodes N] [--host-node 1]
#                                      [--subnet 192.168.56] [--iso f.iso]
#                                      [--build-iso] [--no-vms] [--indicatifs "11 22"]
#
# Ecrit la table dans /etc/osmo-wan.conf, prete pour --wan.
# Appele automatiquement par : start-docker.sh --virtualbox  /  start-native.sh --virtualbox
# ═══════════════════════════════════════════════════════════════════════
set -euo pipefail

RED='\033[0;31m'; GREEN='\033[0;32m'; YELLOW='\033[1;33m'
CYAN='\033[0;36m'; BOLD='\033[1m'; NC='\033[0m'

DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")/.." && pwd)"
LAB="$DIR/tools/vbox-wan-lab.sh"
# shellcheck source=network/wan-nodes.sh
. "$DIR/network/wan-nodes.sh"

NODES=""
HOST_NODE=1
SUBNET="192.168.56"
ISO=""
BUILD_ISO=0
NO_VMS=0
INDS=""
PASSTHRU=()

while [ $# -gt 0 ]; do
    case "$1" in
        --nodes)       NODES="${2:-}"; shift ;;
        --nodes=*)     NODES="${1#*=}" ;;
        --host-node)   HOST_NODE="${2:-}"; shift ;;
        --host-node=*) HOST_NODE="${1#*=}" ;;
        --subnet)      SUBNET="${2:-}"; shift ;;
        --subnet=*)    SUBNET="${1#*=}" ;;
        --iso)         ISO="${2:-}"; shift ;;
        --iso=*)       ISO="${1#*=}" ;;
        --build-iso)   BUILD_ISO=1 ;;
        --no-vms)      NO_VMS=1 ;;
        --indicatifs)  INDS="${2:-}"; shift ;;
        --indicatifs=*) INDS="${1#*=}" ;;
        --conf)        WAN_CONF_FILE="${2:-}"; shift ;;
        --conf=*)      WAN_CONF_FILE="${1#*=}" ;;
        -h|--help)     sed -n '2,40p' "$0" | sed 's/^# \{0,1\}//'; exit 0 ;;
        *) PASSTHRU+=("$1") ;;   # transmis tel quel au banc (--ram, --usb-filter...)
    esac
    shift
done

echo -e "${CYAN}${BOLD}"
echo "╔═══════════════════════════════════════════════════════════════╗"
echo "║   Interconnexion WAN - cette machine + VM VirtualBox          ║"
echo "╚═══════════════════════════════════════════════════════════════╝"
echo -e "${NC}"

# ── Garde-fous ──────────────────────────────────────────────────────
# Lance DANS une VM, ce script ne peut rien faire d'utile et ses messages
# d'erreur ne le diraient pas : VBoxManage y est simplement absent.
if command -v systemd-detect-virt >/dev/null 2>&1; then
    # `|| echo none` produirait deux lignes quand la commande sort en 1 (cas
    # "aucune virtualisation") : on ne garde que la premiere.
    _virt="$(systemd-detect-virt 2>/dev/null | head -1)" || true
    _virt="${_virt:-none}"
    case "$_virt" in
        oracle|kvm|qemu|vmware|microsoft)
```

```
            echo -e "${RED}Vous etes DANS une machine virtuelle (${_virt}).${NC}" >&2
            echo -e "  L'interconnexion VirtualBox se monte depuis l'HOTE." >&2
            echo -e "  Ici, le WAN se configure normalement : ${CYAN}./start-native.sh --wan${NC}" >&2
            exit 1 ;;
    esac
fi

VBM="$(command -v VBoxManage || command -v vboxmanage || true)"
if [ -z "$VBM" ]; then
    echo -e "${RED}VBoxManage introuvable - VirtualBox n'est pas installe.${NC}" >&2
    exit 1
fi
if ! grep -q '^vboxdrv ' /proc/modules 2>/dev/null; then
    echo -e "${RED}Le module vboxdrv n'est pas charge : aucune VM ne demarrera.${NC}" >&2
    echo -e "  ${CYAN}sudo /sbin/vboxconfig${NC}" >&2
    if command -v mokutil >/dev/null 2>&1 && mokutil --sb-state 2>/dev/null | grep -i enabled >/dev/null; then
        echo -e "  ${YELLOW}Secure Boot actif : le module doit etre signe et sa cle enrolee (MOK).${NC}" >&2
    fi
    exit 1
fi
[ "$(id -u)" -eq 0 ] || { echo -e "${RED}Root requis (interface reseau + /etc/osmo-wan.conf).${NC}" >&2; exit 1; }

# ── Combien de noeuds ──────────────────────────────────────────────
if [ -z "$NODES" ]; then
    if [ -t 0 ]; then
        # Meme question que ./launch/start-docker.sh --wan, meme bornes.
       NODES=$(_wan_ask "WAN VirtualBox" "Nombre de noeuds au total, cette machine comprise (1-9) :" "3") || exit 1
    else
        NODES=3
    fi
fi
[[ "$NODES" =~ ^[1-9]$ ]] || { echo -e "${RED}--nodes doit valoir 1 a 9${NC}" >&2; exit 2; }
[[ "$HOST_NODE" =~ ^[1-9]$ ]] && [ "$HOST_NODE" -le "$NODES" ] \
    || { echo -e "${RED}--host-node doit designer un noeud de 1 a ${NODES}${NC}" >&2; exit 2; }

# ── Plage autorisee ? ──────────────────────────────────────────────
if [ "$SUBNET" != "192.168.56" ] && ! grep -q "${SUBNET}" /etc/vbox/networks.conf 2>/dev/null; then
    echo -e "${YELLOW}Attention : VirtualBox n'autorise que 192.168.56.0/21 en host-only.${NC}"
    echo -e "  Pour ouvrir ${SUBNET}.0/24 : ${CYAN}echo '* ${SUBNET}.0/24' | sudo tee -a
     ↪  /etc/vbox/networks.conf${NC}"
    echo ""
fi

# ── Table des noeuds ───────────────────────────────────────────────
declare -a ind_arr=()
[ -n "$INDS" ] && read -r -a ind_arr <<< "$INDS"
WAN_IP=(); WAN_IND=(); WAN_NODE_LIST=()
for i in $(seq 1 "$NODES"); do
    # Noeud N → .1N, sans exception pour l'hote : la table se lit d'un coup
    # d'oeil et le numero de noeud se retrouve dans l'adresse.
    WAN_IP[$i]="${SUBNET}.$((10 + i))"
    WAN_IND[$i]="${ind_arr[$((i-1))]:-$(wan_default_ind "$i")}"
    WAN_NODE_LIST+=("$i")
done
WAN_NODE_COUNT="$NODES"
WAN_NODE_ID="$HOST_NODE"
WAN_OPS="${WAN_OPS:-1}"
wan_nodes_validate || exit 1

echo -e "  Cette machine porte le ${BOLD}noeud ${HOST_NODE}${NC} ; les autres sont des VM."
wan_nodes_summary
echo ""

# ── Le banc de VM fait le reste : reseau host-only, DHCP fixe, materiel ──────
if [ "$NO_VMS" -eq 1 ]; then
    echo -e "  ${YELLOW}--no-vms : ni reseau ni VM crees, seule la table est ecrite.${NC}"
```

```
else
    [ -x "$LAB" ] || { echo -e "${RED}tools/vbox-wan-lab.sh introuvable${NC}" >&2; exit 1; }
    lab_args=(create --nodes "$NODES" --host-node "$HOST_NODE" --subnet "$SUBNET"
              --indicatifs "$(for i in "${WAN_NODE_LIST[@]}"; do printf '%s ' "${WAN_IND[$i]}"; done)")
    [ -n "$ISO" ]          && lab_args+=(--iso "$ISO")
    [ "$BUILD_ISO" -eq 1 ] && lab_args+=(--build-iso)
    bash "$LAB" "${lab_args[@]}" ${PASSTHRU[@]+"${PASSTHRU[@]}"} || {
        echo -e "${RED}La creation du banc a echoue.${NC}" >&2; exit 1; }
fi

# ── Table persistee ──────────────────────────────────────────────────
# WAN_AUTO=0 cote hote : ici c'est start-docker.sh / start-native.sh qui decide quand
# monter le WAN. Les ISO des VM, elles, sont construites avec WAN_AUTO=1.
WAN_AUTO=0 wan_nodes_save "$WAN_CONF_FILE"
echo -e "  ${GREEN}✓${NC} table ecrite dans ${CYAN}${WAN_CONF_FILE}${NC}"
echo ""
echo -e "  ${BOLD}Ensuite, sur cette machine :${NC}"
echo -e "    ${CYAN}sudo ./launch/start-docker.sh --wan${NC}         (le lab docker devient le noeud ${HOST_NODE})"
echo -e "    ${CYAN}tools/vbox-wan-lab.sh start${NC}    (demarre les VM)"
echo ""
```

3.256 /opt/GSM/osmo-operator/network/setup-wan-interop.sh

28222 octets, 590 lignes → 590 lignes

```
#!/bin/bash
# ═══════════════════════════════════════════════════════════════════════
# network/setup-wan-interop.sh - Interconnexion WAN entre deux instances osmo-operator
#
# Permet aux MS d'un serveur d'appeler les MS de l'autre serveur via le
# prefixe +66. Exemple : composer 6610001 depuis le serveur A appelle
# le MS 10001 sur le serveur B.
#
# Architecture :
#
#   Serveur A (82.66.231.141)              Serveur B (37.59.111.84)
#   ┌──────────────────────────┐           ┌──────────────────────────┐
#   │ Op1 Asterisk :5060       │           │ Op1 Asterisk :5060       │
#   │   172.20.0.11            │           │   172.20.0.11            │
#   │ Op2 Asterisk :5060       │           │ Op2 Asterisk :5060       │
#   │   172.20.0.12            │           │   172.20.0.12            │
#   └──────────┬───────────────┘           └──────────┬───────────────┘
#              │                                      │
#   Host iptables DNAT                     Host iptables DNAT
#   Op1: :5080 → 172.20.0.11:5060          Op1: :5080 → 172.20.0.11:5060
#   Op2: :5082 → 172.20.0.12:5060          Op2: :5082 → 172.20.0.12:5060
#              │                                      │
#              └──────── SIP/RTP over WAN ────────────┘
#                   82.66.231.141 ↔ 37.59.111.84
#
# Ports WAN par operateur :
#   OpN SIP  : 5080 + (N-1)*2
#   OpN RTP  : 20000 + (N-1)*500  →  20000 + N*500 - 1
#
# Prefixe d'appel :
#   66 + numero_normal  →  route vers le serveur distant
#   Ex: 6610001 = appeler MS 10001 sur le serveur distant
#   Ex: 6620001 = appeler MS 20001 (op2) sur le serveur distant
#
# Usage :
#   sudo ./network/setup-wan-interop.sh <local_public_ip> <remote_public_ip> [n_operators]
#
# Executer sur CHAQUE serveur avec les IPs inversees :
#   Serveur A : sudo ./network/setup-wan-interop.sh 82.66.231.141 37.59.111.84 2
```

```
#   Serveur B : sudo ./network/setup-wan-interop.sh 37.59.111.84 82.66.231.141 2
# ═══════════════════════════════════════════════════════════════════════════

set -euo pipefail

RED='\033[0;31m'; GREEN='\033[0;32m'; YELLOW='\033[1;33m'
CYAN='\033[0;36m'; BOLD='\033[1m'; NC='\033[0m'

# ── Parametres ──────────────────────────────────────────────────────────────
LOCAL_IP="${1:-}"
REMOTE_IP="${2:-}"
N_OPS="${3:-2}"
WAN_PREFIX="66"                    # Prefixe pour les appels inter-serveur
SIP_WAN_BASE=5080                  # Port SIP WAN de base
RTP_WAN_BASE=20000                 # Port RTP WAN de base
RTP_PER_OP=500                     # Ports RTP par operateur
CONTAINER_PREFIX="osmo-operator-"
SCRIPT_DIR="$(cd "$(dirname "$0")" && pwd)"   # pour appeler network/firewall-wan.sh

if [ -z "$LOCAL_IP" ] || [ -z "$REMOTE_IP" ]; then
    echo -e "${RED}Usage: sudo $0 <local_public_ip> <remote_public_ip> [n_operators]${NC}"
    echo ""
    echo "  Serveur A : sudo $0 82.66.231.142 37.59.111.84 2"
    echo "  Serveur B : sudo $0 37.59.111.84 82.66.231.142 2"
    exit 1
fi

if [ "$(id -u)" -ne 0 ]; then
    echo -e "${RED}Doit etre lance en root (sudo).${NC}"; exit 1
fi

# ── Fonctions helper ────────────────────────────────────────────────────────
op_backbone_ip() { echo "172.20.0.$((10 + $1))"; }
op_sip_port()    { echo $(( SIP_WAN_BASE + ($1 - 1) * 2 )); }
op_rtp_start()   { echo $(( RTP_WAN_BASE + ($1 - 1) * RTP_PER_OP )); }
op_rtp_end()     { echo $(( RTP_WAN_BASE + $1 * RTP_PER_OP - 1 )); }
op_container()   { echo "${CONTAINER_PREFIX}$1"; }

# NOTE : sed -i echoue sur les bind mounts Docker (rename() impossible).
# Toutes les editions sed utilisent le pattern : cp file /tmp → sed -i /tmp → cp back

# ═══════════════════════════════════════════════════════════════════════════
# Banner
# ═══════════════════════════════════════════════════════════════════════════
echo -e "${CYAN}${BOLD}"
echo "╔═══════════════════════════════════════════════════════════════════════╗"
echo "║        WAN Interop - Interconnexion osmo-operator distante            ║"
echo "╚═══════════════════════════════════════════════════════════════════════╝"
echo -e "${NC}"
echo -e "  Local       : ${CYAN}${LOCAL_IP}${NC}"
echo -e "  Remote      : ${CYAN}${REMOTE_IP}${NC}"
echo -e "  Operateurs  : ${CYAN}${N_OPS}${NC}"
echo -e "  Prefixe WAN : ${CYAN}${WAN_PREFIX}${NC}"
echo ""

# ── Verification des containers ─────────────────────────────────────────────
echo -e "${GREEN}[1/5] Verification des containers...${NC}"
for i in $(seq 1 "$N_OPS"); do
    cname=$(op_container "$i")
    if ! docker ps --format '{{.Names}}' | grep -q "^${cname}$"; then
        echo -e "  ${RED}✗ ${cname} non trouve${NC}"
        echo -e "  ${YELLOW}Lancez start-docker.sh d'abord.${NC}"
        exit 1
    fi
    echo -e "  ${GREEN}✓${NC} ${cname} actif"
done
```

```
echo ""

# ═══════════════════════════════════════════════════════════════════════
# [2/5] iptables - DNAT pour le trafic entrant depuis le serveur distant
# ═══════════════════════════════════════════════════════════════════════
echo -e "${GREEN}[2/5] Configuration iptables (DNAT entrant)...${NC}"

# Activer le forwarding IP
sysctl -w net.ipv4.ip_forward=1 > /dev/null 2>&1

# Nettoyer les anciennes regles WAN interop (chaine personnalisee)
iptables -t nat -D PREROUTING -j OSMO_WAN_INTEROP 2>/dev/null || true
iptables -t nat -F OSMO_WAN_INTEROP 2>/dev/null || true
iptables -t nat -X OSMO_WAN_INTEROP 2>/dev/null || true

# Creer la chaine
iptables -t nat -N OSMO_WAN_INTEROP

for i in $(seq 1 "$N_OPS"); do
    bb_ip=$(op_backbone_ip "$i")
    sip_port=$(op_sip_port "$i")
    rtp_start=$(op_rtp_start "$i")
    rtp_end=$(op_rtp_end "$i")

    # SIP signaling : port externe → container:5060
    iptables -t nat -A OSMO_WAN_INTEROP \
        -s "$REMOTE_IP" -p udp --dport "$sip_port" \
        -j DNAT --to-destination "${bb_ip}:5060"

    # SIP TCP (au cas ou)
    iptables -t nat -A OSMO_WAN_INTEROP \
        -s "$REMOTE_IP" -p tcp --dport "$sip_port" \
        -j DNAT --to-destination "${bb_ip}:5060"

    # RTP : plage de ports → container (meme ports en interne)
    iptables -t nat -A OSMO_WAN_INTEROP \
        -s "$REMOTE_IP" -p udp --dport "${rtp_start}:${rtp_end}" \
        -j DNAT --to-destination "${bb_ip}"

    echo -e "  ${CYAN}Op${i}${NC} SIP :${sip_port} → ${bb_ip}:5060  RTP ${rtp_start}-${rtp_end} → ${bb_ip}"
done

# Inserer la chaine dans PREROUTING
iptables -t nat -I PREROUTING -j OSMO_WAN_INTEROP

# MASQUERADE pour le retour (le container voit le trafic comme venant du host)
# --ctstate DNAT : sans restriction, ce masquage prenait AUSSI les flux du LAN
# simplement routes vers un conteneur ; celui-ci repondait alors a l'hote et non
# au client, qui n'a jamais vu la reponse. On ne masque que ce que le DNAT
# ci-dessus a redirige. Restreindre plutot sur l'adresse de l'hote ne matcherait
# rien : apres DNAT la source est celle du serveur distant, pas la notre.
# L'ancienne regle large est retiree d'abord - le garde -C teste la spec exacte,
# sans cela les deux resteraient en place et la large gagnerait.
iptables -t nat -D POSTROUTING -d 172.20.0.0/24 -j MASQUERADE 2>/dev/null || true
if ! iptables -t nat -C POSTROUTING -d 172.20.0.0/24 -m conntrack --ctstate DNAT -j MASQUERADE 2>/dev/null; then
    # Repli si le module conntrack d'iptables manque : sans lui le -A echoue et
    # le script s'arretait la, la regle large ayant deja ete retiree - plus
    # aucun masquage, donc plus aucun retour vers les conteneurs.
    iptables -t nat -A POSTROUTING -d 172.20.0.0/24 -m conntrack --ctstate DNAT -j MASQUERADE 2>/dev/null || {
        echo -e "  ${YELLOW}! -m conntrack indisponible : masquage large 172.20.0.0/24 (moins precis)${NC}"
        iptables -t nat -C POSTROUTING -d 172.20.0.0/24 -j MASQUERADE 2>/dev/null \
            || iptables -t nat -A POSTROUTING -d 172.20.0.0/24 -j MASQUERADE || true
    }
fi

echo -e "  ${GREEN}✓ iptables configure${NC}"
```

```
echo ""

# ═══════════════════════════════════════════════════════════════════════
# [2bis/5] Firewall - ouverture automatique des ports entrants depuis le distant
# Le DNAT ci-dessus redirige le trafic, mais si un firewall (ufw/firewalld/
# iptables INPUT) filtre l'INPUT, SIP/RTP WAN sont droppes -> pas d'audio WAN.
# On applique donc les regles a CHAQUE lancement du WAN (network/firewall-wan.sh est
# idempotent : pas de doublon si relance).
# ═══════════════════════════════════════════════════════════════════════
echo -e "${GREEN}[2bis/5] Firewall - ouverture des ports WAN (entrant ${REMOTE_IP})...${NC}"
if [ -x "$SCRIPT_DIR/firewall-wan.sh" ] || [ -f "$SCRIPT_DIR/firewall-wan.sh" ]; then
    if bash "$SCRIPT_DIR/firewall-wan.sh" "$REMOTE_IP" "$N_OPS"; then
        echo -e "  ${GREEN}✓ Firewall ouvert pour ${REMOTE_IP}${NC}"
    else
        echo -e "  ${YELLOW}⚠ network/firewall-wan.sh a echoue - ouvre les ports a la main (voir resume).${NC}"
    fi
else
    echo -e "  ${YELLOW}⚠ network/firewall-wan.sh introuvable dans ${SCRIPT_DIR} - ports a ouvrir a la main.${NC}"
fi
echo ""

# ═══════════════════════════════════════════════════════════════════════
# [3/5] Configuration RTP - plage de ports WAN par operateur
# ═══════════════════════════════════════════════════════════════════════
echo -e "${GREEN}[3/5] Configuration RTP WAN dans chaque container...${NC}"

for i in $(seq 1 "$N_OPS"); do
    cname=$(op_container "$i")
    rtp_start=$(op_rtp_start "$i")
    rtp_end=$(op_rtp_end "$i")

    # Creer/modifier rtp.conf dans Asterisk pour utiliser la plage WAN
    # On garde la plage originale pour les appels locaux, mais on ajoute
    # la config dans le transport PJSIP (external_media_address)
    docker exec "$cname" bash -c "cat > /etc/asterisk/rtp.conf << 'RTPEOF'
[general]
rtpstart=${rtp_start}
rtpend=${rtp_end}
strictrtp=no
icesupport=no
RTPEOF"

    echo -e "  ${CYAN}Op${i}${NC} RTP range: ${rtp_start}-${rtp_end}"
done

echo -e "  ${GREEN}✓ RTP configure${NC}"
echo ""

# ═══════════════════════════════════════════════════════════════════════
# [4/5] PJSIP + Dialplan - Trunks WAN et contexte interop distant
# ═══════════════════════════════════════════════════════════════════════
echo -e "${GREEN}[4/5] Injection PJSIP trunks + dialplan WAN...${NC}"

for i in $(seq 1 "$N_OPS"); do
    cname=$(op_container "$i")
    local_sip_port=$(op_sip_port "$i")
    rtp_start=$(op_rtp_start "$i")
    rtp_end=$(op_rtp_end "$i")

    # ── PJSIP : patch transport-udp + trunks vers chaque operateur distant ──

    # Generer les blocs PJSIP pour les trunks WAN
    pjsip_wan=""

    # Pas de transport-wan separe - on patche transport-udp existant
    # pour ajouter external_media_address (necessaire pour SDP corrects)
```

```
    docker exec "$cname" bash -c "
        if ! grep -q 'external_media_address' /etc/asterisk/pjsip.conf; then
            cp /etc/asterisk/pjsip.conf /tmp/_pj.work
            sed -i '/^\[transport-udp\]/,/^\$/ {
                /^\$/i\\
external_media_address=${LOCAL_IP}\\
external_signaling_address=${LOCAL_IP}\\
local_net=172.20.0.0/24\\
local_net=127.0.0.0/8
            }' /tmp/_pj.work
            cp /tmp/_pj.work /etc/asterisk/pjsip.conf
            rm -f /tmp/_pj.work
        fi
    "

    pjsip_wan+="
; ═══════════════════════════════════════════════════════════════════
; WAN INTEROP - Trunks vers serveur distant ${REMOTE_IP}
; Genere par network/setup-wan-interop.sh le $(date '+%Y-%m-%d %H:%M:%S')
; ═══════════════════════════════════════════════════════════════════
"

    # Un identify par ADRESSE, hors de la boucle : identify ne regarde que
    # l'adresse source, jamais le port. Un bloc par operateur distant en posait
    # autant pour la meme adresse, et l'appel entrant etait attribue a l'un
    # d'eux au hasard. Deux adresses seulement : la passerelle du bridge, par
    # laquelle arrive le trafic WAN masque par l'hote, et le serveur distant
    # lui-meme quand il n'est pas masque.
    pjsip_wan+="
; ── Identification des appels entrants du WAN ──────────────────────
[wan-identify-remote]
type=identify
endpoint=wan_trunk_op1
match=${REMOTE_IP}

[wan-identify-gw]
type=identify
endpoint=wan_trunk_op1
match=172.20.0.1
"

    # Un trunk par operateur distant
    for j in $(seq 1 "$N_OPS"); do
        remote_sip_port=$(op_sip_port "$j")

        pjsip_wan+="
; ── Trunk WAN → Serveur distant Op${j} (${REMOTE_IP}:${remote_sip_port}) ─────
[wan_trunk_op${j}]
type=endpoint
transport=transport-udp
context=wan_in
disallow=all
allow=gsm
allow=ulaw
aors=wan_trunk_op${j}
media_encryption=no
direct_media=no
rtp_symmetric=yes
force_rport=yes
rewrite_contact=yes
ice_support=no

[wan_trunk_op${j}]
type=aor
contact=sip:${REMOTE_IP}:${remote_sip_port}
qualify_frequency=30
```

```
qualify_timeout=5.0
"
    done

    # Injecter dans pjsip.conf
    # D'abord retirer l'ancien bloc WAN s'il existe
    docker exec "$cname" bash -c "
        cp /etc/asterisk/pjsip.conf /tmp/_pj.work
        sed -i '/; ==.*WAN INTEROP/,\$d' /tmp/_pj.work 2>/dev/null || true
        cp /tmp/_pj.work /etc/asterisk/pjsip.conf
        rm -f /tmp/_pj.work
    "

    # Ajouter le nouveau bloc
    docker exec "$cname" bash -c "cat >> /etc/asterisk/pjsip.conf << 'PJSIPEOF'
${pjsip_wan}
PJSIPEOF"

    echo -e "  ${CYAN}Op${i}${NC} PJSIP: ${N_OPS} trunk(s) WAN ajoutes"

    # ── Dialplan : contextes [wan_out] et [wan_in] ──────────────────────

    # ── Softphones SIP : numero → endpoint, extraits DYNAMIQUEMENT de pjsip.conf ──
    # Tout endpoint portant un "callerid=... <NNN>" (ex: linphone_A <100>) devient
    # routable SIP→SIP over WAN dans les deux sens, sans hardcoder les numeros.
    softphones=$(docker exec "$cname" cat /etc/asterisk/pjsip.conf | awk '
        /^\[/                 { s=$0; gsub(/[][]/,"",s); ep=s }
        /callerid=.*<[0-9]+>/ { n=$0; sub(/.*</,"",n); sub(/>.*/,"",n); print n":"ep }
    ')
    echo -e "  ${CYAN}Op${i}${NC} Softphones SIP detectes: $(echo "$softphones" | tr '\n' ' ')"

    # Generer le contexte wan_out (appels sortants vers le distant)
    wan_dialplan="
; ═══════════════════════════════════════════════════════════════════
; WAN INTEROP - Routage appels inter-serveur (prefixe ${WAN_PREFIX})
; Genere par network/setup-wan-interop.sh le $(date '+%Y-%m-%d %H:%M:%S')
;
; Composer ${WAN_PREFIX}NXXXX = appeler NXXXX sur le serveur distant
; Composer ${WAN_PREFIX}NXXXXX = appeler NXXXXX sur le serveur distant
; Ex: ${WAN_PREFIX}10001 → MS 10001 op1 distant
;     ${WAN_PREFIX}20001 → MS 20001 op2 distant
; ═══════════════════════════════════════════════════════════════════

[wan_out]
"
    for j in $(seq 1 "$N_OPS"); do
        # Pattern pour prefixe 1 chiffre operateur (1-9)
        if [ "$j" -lt 10 ]; then
            wan_dialplan+="
; → Operateur distant ${j} (4 chiffres)
exten => _${j}XXXX,1,NoOp(=== WAN OUT Op${j}: \${EXTEN} → ${REMOTE_IP} ===)
 same => n,Dial(PJSIP/\${EXTEN}@wan_trunk_op${j},,rT)
 same => n,NoOp(WAN: \${DIALSTATUS})
 same => n,Congestion()
 same => n,Hangup()

; → Operateur distant ${j} (5 chiffres)
exten => _${j}XXXXX,1,NoOp(=== WAN OUT Op${j}: \${EXTEN} → ${REMOTE_IP} ===)
 same => n,Dial(PJSIP/\${EXTEN}@wan_trunk_op${j},,rT)
 same => n,NoOp(WAN: \${DIALSTATUS})
 same => n,Congestion()
 same => n,Hangup()
"
        else
            wan_dialplan+="
; → Operateur distant ${j} (prefixe 2 chiffres)
```

```
exten => _${j}XXXX,1,NoOp(=== WAN OUT Op${j}: \${EXTEN} → ${REMOTE_IP} ===)
 same => n,Dial(PJSIP/\${EXTEN}@wan_trunk_op${j},,rT)
 same => n,Congestion()
 same => n,Hangup()
"
        fi
    done

    # → Softphones SIP distants (SIP→SIP over WAN) - generes dynamiquement.
    #   Un Linphone local compose ${WAN_PREFIX}<num> ; le ${WAN_PREFIX} est strippe
    #   en amont (Goto wan_out,\${EXTEN:2}), on route donc <num> vers le softphone
    #   homologue du meme operateur sur le serveur distant.
    for sp in $softphones; do
        sp_num="${sp%%:*}"
        wan_dialplan+="
; → Softphone SIP distant ${sp_num} (Op${i})
exten => ${sp_num},1,NoOp(=== WAN OUT SIP Op${i}: ${sp_num} → ${REMOTE_IP} ===)
 same => n,Dial(PJSIP/${sp_num}@wan_trunk_op${i},,rT)
 same => n,NoOp(WAN: \${DIALSTATUS})
 same => n,Congestion()
 same => n,Hangup()
"
    done

    wan_dialplan+="
; Fallback WAN
exten => _X.,1,NoOp(=== WAN OUT: destination inconnue \${EXTEN} ===)
 same => n,Congestion()
 same => n,Hangup()

; ═══════════════════════════════════════════════════════════════════
; [wan_in] - Appels entrants DEPUIS le serveur distant
; ═══════════════════════════════════════════════════════════════════
[wan_in]

; Appel distant → MS local (meme operateur)
exten => _${i}XXXX,1,NoOp(=== WAN IN → GSM local Op${i}: \${EXTEN} ===)
 same => n,Set(CALLERID(all)=<\${CALLERID(num)}>)
 same => n,Gosub(sub-record,s,1(\${EXTEN}))
 same => n,Dial(PJSIP/\${EXTEN}@gsm_msc,,rT)
 same => n,Congestion()
 same => n,Hangup()

exten => _${i}XXXXX,1,NoOp(=== WAN IN → GSM local Op${i}: \${EXTEN} ===)
 same => n,Set(CALLERID(all)=<\${CALLERID(num)}>)
 same => n,Gosub(sub-record,s,1(\${EXTEN}))
 same => n,Dial(PJSIP/\${EXTEN}@gsm_msc,,rT)
 same => n,Congestion()
 same => n,Hangup()
"

    # Softphones SIP locaux (appels entrants depuis le serveur distant) - dynamiques
    for sp in $softphones; do
        sp_num="${sp%%:*}"
        sp_ep="${sp#*:}"
        wan_dialplan+="
; Softphone SIP local ${sp_num} → ${sp_ep}
exten => ${sp_num},1,NoOp(=== WAN IN → ${sp_ep} local (${sp_num}) ===)
 same => n,Gosub(sub-record,s,1(${sp_num}))
 same => n,Dial(PJSIP/${sp_ep},,rT)
 same => n,Congestion()
 same => n,Hangup()
"
    done

    wan_dialplan+="
```

```
; Appel distant → autre operateur local (re-route via interop_out)
exten => _X.,1,NoOp(=== WAN IN → routage local: \${EXTEN} ===)
 same => n,Gosub(sub-record,s,1(\${EXTEN}))
 same => n,Goto(interop_out,\${EXTEN},1)

; Echo test depuis le distant
exten => 600,1,NoOp(=== WAN ECHO TEST ===)
 same => n,Answer()
 same => n,Playback(demo-echotest)
 same => n,Echo()
 same => n,Playback(demo-echodone)
 same => n,Hangup()
"

    # Retirer l'ancien bloc WAN du dialplan
    docker exec "$cname" bash -c '
        cp /etc/asterisk/extensions.conf /tmp/_ext.work
        sed -i "/; ==.*WAN INTEROP.*Routage/,\$d" /tmp/_ext.work 2>/dev/null || true
        cp /tmp/_ext.work /etc/asterisk/extensions.conf
        rm -f /tmp/_ext.work
    '

    # Ajouter au dialplan
    docker exec "$cname" bash -c "cat >> /etc/asterisk/extensions.conf << 'EXTEOF'
${wan_dialplan}
EXTEOF"

    # ── Injecter le pattern _66. directement dans [gsm_in] et [internal] ──
    docker exec "$cname" bash -c '
        EXT="/etc/asterisk/extensions.conf"
        if ! grep -q "_'"${WAN_PREFIX}"'\." "$EXT"; then
            cp "$EXT" /tmp/_ext.work

            sed -i "/^\[gsm_in\]$/a\\
\\
; -- WAN : prefixe '"${WAN_PREFIX}"' -> serveur distant --\\
exten => _'"${WAN_PREFIX}"'.,1,NoOp(=== GSM -> WAN: strip '"${WAN_PREFIX}"' ===)\\
 same => n,Gosub(sub-record,s,1(\${EXTEN:2}))\\
 same => n,Goto(wan_out,\${EXTEN:2},1)" /tmp/_ext.work

            sed -i "/^\[internal\]$/a\\
\\
; -- WAN : prefixe '"${WAN_PREFIX}"' -> serveur distant --\\
exten => _'"${WAN_PREFIX}"'.,1,NoOp(=== SIP -> WAN: strip '"${WAN_PREFIX}"' ===)\\
 same => n,Gosub(sub-record,s,1(\${EXTEN:2}))\\
 same => n,Goto(wan_out,\${EXTEN:2},1)" /tmp/_ext.work

            cp /tmp/_ext.work "$EXT"
            rm -f /tmp/_ext.work
        fi
    '

    echo -e "  ${CYAN}Op${i}${NC} Dialplan: wan_out + wan_in + routage ${WAN_PREFIX}"
done

echo -e "  ${GREEN}✓ PJSIP + Dialplan configures${NC}"
echo ""

# ═══════════════════════════════════════════════════════════════
# [5/5] Restart Asterisk (transport modifie → restart complet requis)
# ═══════════════════════════════════════════════════════════════
echo -e "${GREEN}[5/5] Restart Asterisk...${NC}"

for i in $(seq 1 "$N_OPS"); do
    cname=$(op_container "$i")
```

```
    docker exec "$cname" bash -c "
        asterisk -rx 'core stop now' 2>/dev/null || pkill asterisk 2>/dev/null || true
        sleep 2
        pkill -9 asterisk 2>/dev/null || true
        sleep 1
        rm -f /var/lib/asterisk/astdb.sqlite3
        asterisk -f &
        disown
    " 2>/dev/null || true

    echo -e "  ${CYAN}Op${i}${NC} Asterisk redemarre"
done

echo ""

# ═══════════════════════════════════════════════════════════════════════
# Resume
# ═══════════════════════════════════════════════════════════════════════
echo -e "${GREEN}${BOLD}"
echo "╔═══════════════════════════════════════════════════════════════════╗"
echo "║              WAN Interop configure avec succes !                  ║"
echo "╚═══════════════════════════════════════════════════════════════════╝"
echo -e "${NC}"

echo -e "  ${BOLD}Serveur local  :${NC} ${CYAN}${LOCAL_IP}${NC}"
echo -e "  ${BOLD}Serveur distant:${NC} ${CYAN}${REMOTE_IP}${NC}"
echo ""

echo -e "  ${BOLD}Ports WAN par operateur :${NC}"
for i in $(seq 1 "$N_OPS"); do
    sip_port=$(op_sip_port "$i")
    rtp_start=$(op_rtp_start "$i")
    rtp_end=$(op_rtp_end "$i")
    echo -e "    Op${i}: SIP ${CYAN}:${sip_port}${NC}  RTP ${CYAN}${rtp_start}-${rtp_end}${NC}"
done

echo ""
echo -e "  ${BOLD}Comment appeler :${NC}"
echo -e "    Depuis un MS local, composer ${CYAN}${WAN_PREFIX}${NC} + numero distant"
echo -e "    Ex: ${CYAN}${WAN_PREFIX}10001${NC} = appeler MS 10001 op1 sur ${REMOTE_IP}"
echo -e "    Ex: ${CYAN}${WAN_PREFIX}20001${NC} = appeler MS 20001 op2 sur ${REMOTE_IP}"
echo ""

echo -e "  ${BOLD}Firewall :${NC} ${GREEN}deja applique automatiquement sur CE serveur${NC} (etape 2bis)."
echo -e "    Pense a lancer le WAN aussi sur ${CYAN}${REMOTE_IP}${NC} (il ouvrira son propre INPUT)."
echo -e "    Pour re-ouvrir manuellement : ${CYAN}sudo ${SCRIPT_DIR}/network/firewall-wan.sh ${REMOTE_IP}
 ↪  ${N_OPS}${NC}"
echo -e "    Equivalent ufw :"
echo ""
for i in $(seq 1 "$N_OPS"); do
    sip_port=$(op_sip_port "$i")
    rtp_start=$(op_rtp_start "$i")
    rtp_end=$(op_rtp_end "$i")
    echo -e "    ${YELLOW}# Op${i}${NC}"
    echo -e "    ufw allow from ${REMOTE_IP} to any port ${sip_port} proto udp"
    echo -e "    ufw allow from ${REMOTE_IP} to any port ${rtp_start}:${rtp_end} proto udp"
done

echo ""
echo -e "  ${BOLD}Test rapide :${NC}"
echo -e "    # Depuis le container op1 sur ce serveur :"
echo -e "    docker exec -it osmo-operator-1 bash"
echo -e "    # Dans le VTY mobile (Ctrl-b → fenetre ue_g1) :"
echo -e "    call 1 ${WAN_PREFIX}10001"
echo ""
echo -e "  ${BOLD}Diagnostic :${NC}"
```

```
echo -e "    # Verifier la connectivite SIP :"
echo -e "    docker exec osmo-operator-1 asterisk -rx 'pjsip show endpoints'"
echo -e "    docker exec osmo-operator-1 asterisk -rx 'pjsip show aors'"
echo -e "    # Logs Asterisk :"
echo -e "    docker exec osmo-operator-1 asterisk -rx 'core set verbose 5'"
echo ""

# ── Sauvegarder la config pour persistence ──────────────────────────
CONFIG_SAVE="/etc/osmo-wan-interop.conf"
cat > "$CONFIG_SAVE" << EOF
# osmo-wan-interop.conf - sauvegarde config WAN
# Genere le $(date)
LOCAL_IP=${LOCAL_IP}
REMOTE_IP=${REMOTE_IP}
N_OPS=${N_OPS}
WAN_PREFIX=${WAN_PREFIX}
SIP_WAN_BASE=${SIP_WAN_BASE}
RTP_WAN_BASE=${RTP_WAN_BASE}
RTP_PER_OP=${RTP_PER_OP}
EOF

echo -e "  Config sauvegardee dans ${CYAN}${CONFIG_SAVE}${NC}"
echo -e "  Relancer apres reboot : ${CYAN}sudo $0 ${LOCAL_IP} ${REMOTE_IP} ${N_OPS}${NC}"
echo ""
```

3.257 /opt/GSM/osmo-operator/network/setup-wan-mesh.sh

42761 octets, 837 lignes → 837 lignes

```
#!/bin/bash
# ═══════════════════════════════════════════════════════════════════════
# network/setup-wan-mesh.sh - WAN a N noeuds, route par INDICATIF
#
# Generalise network/setup-wan-interop.sh (qui ne connait que 2 serveurs) :
# ici la topologie est une TABLE de noeuds (network/wan-nodes.sh), chacun avec
# son indicatif, et TOUT est genere en un seul passage - donc relancable et
# valable pour 2 noeuds comme pour 12.
#
#   Composer <indicatif_cible><numero>  →  ce numero, sur le noeud cible.
#   L'indicatif est retire a la SORTIE : le noeud d'arrivee recoit le numero nu,
#   celui que son HLR et son dialplan connaissent deja.
#
# Ce que ce script configure, sur CE noeud :
#   • iptables : entree SIP/RTP/SMS depuis chaque pair (DNAT docker, REDIRECT natif)
#   • firewall : ouverture des memes ports (network/firewall-wan.sh, idempotent)
#   • Asterisk : rtp.conf, trunks PJSIP vers chaque (noeud, operateur) distant,
#                dialplan [wan_out]/[wan_in] + entrees par indicatif
#   • SMS      : sms-routing.conf - un operateur distant par (noeud, operateur),
#                avec strip de l'indicatif (cf. scripts/sms-interop-relay.py)
#
# Usage :
#   sudo network/setup-wan-mesh.sh --nodes "1:1.2.3.4:33 2:5.6.7.8:44" --id 1 [--ops 2]
#   sudo network/setup-wan-mesh.sh --config /etc/osmo-wan.conf
#   options : --native | --docker (defaut : auto)  --no-restart  --dry-run
#
# A lancer sur CHAQUE noeud, avec la MEME table et son propre --id.
# ═══════════════════════════════════════════════════════════════════════
set -euo pipefail

RED='\033[0;31m'; GREEN='\033[0;32m'; YELLOW='\033[1;33m'
CYAN='\033[0;36m'; BOLD='\033[1m'; NC='\033[0m'

SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
# shellcheck source=network/wan-nodes.sh
```

```
. "$SCRIPT_DIR/wan-nodes.sh"

MODE=""
# --op-is-node : l'operateur local i EST le noeud i, et non un operateur du
# noeud --id. C'est le cas quand chaque conteneur porte sa propre adresse et
# son propre indicatif : sans cette notion, un seul jeu de pairs etait calcule
# pour toute la machine, et tous les conteneurs recevaient la meme vue - l'un
# d'eux se retrouvait meme a se router vers lui-meme.
OP_IS_NODE=0            # docker | native
NO_RESTART=0
CONFIG_ONLY=0
DRY=0
CONF=""
CHAIN="OSMO_WAN_MESH"

SIP_WAN_BASE=5080
RTP_WAN_BASE=20000
RTP_PER_OP=500
SMS_RELAY_BASE=7890
CONTAINER_PREFIX="osmo-operator-"
# Prefixe de racine, mode natif seulement : permet de viser un arbre de test
# (NATIVE_ROOT=/tmp/essai) au lieu du /etc de la machine. Vide en production.
NATIVE_ROOT="${NATIVE_ROOT:-}"

wan_sip_port()  { echo $(( SIP_WAN_BASE + ($1 - 1) * 2 )); }
wan_rtp_start() { echo $(( RTP_WAN_BASE + ($1 - 1) * RTP_PER_OP )); }
wan_rtp_end()   { echo $(( RTP_WAN_BASE + $1 * RTP_PER_OP - 1 )); }
wan_sms_port()  { echo $(( SMS_RELAY_BASE + $1 - 1 )); }
op_backbone_ip(){ echo "172.20.0.$((10 + $1))"; }
op_container()  { echo "${CONTAINER_PREFIX}$1"; }

# Un pair dont l'adresse est celle du backbone est joint DANS le bridge : le
# trafic n'y traverse pas la table nat de l'hote (br_netfilter n'est pas
# charge), donc les ports PUBLIES sur l'hote - 5080/5082 pour SIP, 7891 pour le
# relais SMS du second operateur - ne repondent a personne. Les contacts et les
# routes visaient ces ports-la : le qualify restait KO et les SMS partaient dans
# le vide. Vers un tel pair il faut viser ce que le conteneur ecoute vraiment,
# 5060 et le port de base du relais.
peer_is_backbone() { case "$1" in 172.20.0.*) return 0 ;; *) return 1 ;; esac; }

# Sur quel port joindre l'operateur j du noeud r.
#
# Le plan 5080/5082/... n'existe QUE pour multiplexer plusieurs operateurs
# derriere une seule adresse : c'est un hote docker qui republie les ports de
# ses conteneurs, ou un noeud natif qui pose des REDIRECT vers 5060. Un noeud
# qui ne porte QU'UN operateur n'a rien a multiplexer - son Asterisk ecoute
# directement sur 5060.
#
# Le discriminant etait "l'adresse est-elle sur le backbone docker". C'etait
# trop etroit : le noeud 3 du banc est une VM NATIVE, hors backbone, avec un
# seul operateur. On visait donc 192.168.1.2:5080, ou rien n'ecoute, et le
# trunk restait "Unavail" pendant que la VM repondait tranquillement sur 5060 :
#     192.168.1.2:5060  -> SIP/2.0 401 Unauthorized
#     192.168.1.2:5080  -> aucune reponse
# (Elle repondrait sur 5080 si le maillage y avait tourne - il pose les
#  REDIRECT - mais on ne peut pas en faire une hypothese depuis ici.)
peer_sip_port() {          # $1 = id du noeud distant, $2 = operateur
    if [ "$(node_nops "$1")" -eq 1 ]; then echo 5060; else wan_sip_port "$2"; fi
}
peer_sms_port() {          # $1 = id du noeud distant, $2 = operateur
    if [ "$(node_nops "$1")" -eq 1 ]; then echo "$SMS_RELAY_BASE"; else wan_sms_port "$2"; fi
}

# L'adresse que la TABLE annonce pour nous n'est pas toujours celle que porte le
# paquet a l'arrivee : derriere un routeur, la table donne l'adresse publique et
# l'hote ne voit jamais que sa privee. Un "-d" sur une adresse absente de la
```

```
# machine ne matche alors PLUS RIEN, et le WAN meurt en silence - le cas de
# l'en-tete de ce fichier, deux serveurs derriere leur box. On ne restreint donc
# la destination que si l'adresse est bien a nous.
addr_is_local() {
    ip -o addr show 2>/dev/null | awk '{print $4}' | cut -d/ -f1 | grep -qx "$1"
}

# Nombre d'operateurs portes par un noeud DISTANT. Le compteur local ne le dit
# pas : avec un noeud par conteneur il vaut 2 et fabriquait un trunk vers un
# operateur inexistant, un pattern _<indicatif>2XXXX et une route SMS que
# personne ne sert. WAN_NOPS vient de la table.
#
# Aucun producteur n'ecrit encore le 4e champ : le repli est donc, aujourd'hui,
# le cas general. Un repli fixe a 1 aurait supprime l'operateur 2 de tout
# montage CLASSIQUE ("--wan-nodes 1:a:11 2:b:22 --ops 2", ou chaque noeud porte
# les deux operateurs) - trunk absent, appels inter-noeuds morts pour lui. On
# ne replie sur 1 que la ou c'est vrai par construction : --op-is-node, un
# operateur par noeud. Sinon on garde l'hypothese d'origine, les pairs sont
# montes comme nous.
node_nops() {
    local n="${WAN_NOPS[$1]:-}"
    if ! [[ "$n" =~ ^[1-9][0-9]*$ ]]; then
        if [ "${OP_IS_NODE:-0}" = "1" ]; then n=1; else n="${N_OPS:-1}"; fi
    fi
    echo "$n"
}

# ── Piege `pipefail` + `grep -q` ──────────────────────────────────────────
# `grep -q` sort des la PREMIERE correspondance ; le producteur du pipeline
# recoit alors SIGPIPE et meurt en 141, et sous `set -o pipefail` c'est 141 que
# le pipeline renvoie. Resultat : "trouve" se lit "pas trouve", de facon
# intermittente - selon que la sortie tenait ou non dans le tampon du tube.
# C'est exactement ce qui faisait dire a `check` que vboxdrv n'etait pas charge
# alors qu'il l'etait. On lit donc toute l'entree, et on jette la sortie.
qgrep() { grep "$@" >/dev/null; }

usage() { sed -n '2,26p' "${BASH_SOURCE[0]}" | sed 's/^# \{0,1\}//'; }

while [ $# -gt 0 ]; do
    case "$1" in
        --nodes)   WAN_NODES="${2:-}"; shift ;;
        --nodes=*) WAN_NODES="${1#*=}" ;;
        --id)      WAN_NODE_ID="${2:-}"; shift ;;
        --id=*)    WAN_NODE_ID="${1#*=}" ;;
        --op-is-node) OP_IS_NODE=1 ;;
        --ops)     WAN_OPS="${2:-}"; shift ;;
        --ops=*)   WAN_OPS="${1#*=}" ;;
        --config)  CONF="${2:-}"; shift ;;
        --config=*) CONF="${1#*=}" ;;
        --native)  MODE="native" ;;
        --docker)  MODE="docker" ;;
        --no-restart) NO_RESTART=1 ;;
        --config-only) CONFIG_ONLY=1; NO_RESTART=1 ;;
        --dry-run) DRY=1 ;;
        -h|--help) usage; exit 0 ;;
        *) echo -e "${RED}option inconnue : $1${NC}" >&2; usage >&2; exit 2 ;;
    esac
    shift
done

# ── Table des noeuds ──────────────────────────────────────────────────────
# --config sert a la fois de source et de destination : on ne le lit que s'il
# existe deja, et --nodes reste prioritaire (on decrit une nouvelle topologie).
if [ -n "${WAN_NODES:-}" ]; then
    wan_nodes_parse "$WAN_NODES" || exit 1
elif [ -n "$CONF" ]; then
```

```
    wan_nodes_load "$CONF" || { echo -e "${RED}config WAN illisible : $CONF${NC}" >&2; exit 1; }
elif [ -n "${WAN_NODES:-}" ]; then
    wan_nodes_parse "$WAN_NODES" || exit 1
elif wan_nodes_load "$WAN_CONF_FILE" 2>/dev/null; then
    :
else
    echo -e "${RED}Aucune table de noeuds : --nodes, --config ou $WAN_CONF_FILE${NC}" >&2
    exit 1
fi

# --id explicite > config > auto-detection par IP locale.
if [ "${WAN_NODE_ID:-0}" = "0" ] || [ -z "${WAN_NODE_ID:-}" ]; then
    wan_nodes_detect_self || {
        echo -e "${RED}Impossible de determiner le noeud local : passez --id N${NC}" >&2; exit 1; }
    echo -e "  ${CYAN}Noeud local detecte par son IP : ${WAN_NODE_ID}${NC}"
fi
wan_nodes_validate || exit 1

# ── Mode ──────────────────────────────────────────────────────────────
# L'ISO n'a pas de docker ; l'hote du lab en a un ET des containers osmo-operator-*.
# On ne se fie donc pas a la presence du binaire mais a celle des containers.
if [ -z "$MODE" ]; then
    if command -v docker >/dev/null 2>&1 && \
       docker ps --format '{{.Names}}' 2>/dev/null | qgrep "^${CONTAINER_PREFIX}1$"; then
        MODE="docker"
    else
        MODE="native"
    fi
fi

if [ "$MODE" = "native" ]; then
    OP_IDS=( "${NATIVE_OP_ID:-${OPERATOR_ID:-1}}" )
else
    [ "${WAN_OPS:-1}" -ge 1 ] || WAN_OPS=1
    OP_IDS=(); for i in $(seq 1 "$WAN_OPS"); do OP_IDS+=("$i"); done
fi
N_OPS="${#OP_IDS[@]}"

LOCAL_IP="$(wan_local_ip)"
LOCAL_IND="$(wan_local_ind)"

echo -e "${CYAN}${BOLD}"
echo "╔══════════════════════════════════════════════════════════════════╗"
echo "║   WAN mesh - routage par indicatif entre N noeuds osmo-operator  ║"
echo "╚══════════════════════════════════════════════════════════════════╝"
echo -e "${NC}"
echo -e "  Mode        : ${CYAN}${MODE}${NC}   Operateurs locaux : ${CYAN}${N_OPS}${NC} (${OP_IDS[*]})"
wan_nodes_summary
echo ""

[ "$(id -u)" -ne 0 ] && { echo -e "${RED}Root requis (iptables + configs).${NC}" >&2; exit 1; }

REMOTES=()
for id in "${WAN_NODE_LIST[@]}"; do [ "$id" = "$WAN_NODE_ID" ] || REMOTES+=("$id"); done

# Le contexte d'UN operateur : son noeud, son indicatif, son adresse et ses
# pairs. Sans --op-is-node il ne change pas d'un operateur a l'autre (ils
# partagent le noeud de la machine) ; avec, chacun a le sien.
mesh_set_context() {
    local op="$1" node id
    if [ "${OP_IS_NODE:-0}" = "1" ]; then node="$op"; else node="$WAN_NODE_ID"; fi
    MESH_NODE="$node"
    [ -n "${WAN_IND[$node]:-}" ] && LOCAL_IND="${WAN_IND[$node]}"
    [ -n "${WAN_IP[$node]:-}"  ] && LOCAL_IP="${WAN_IP[$node]}"
    REMOTES=()
    for id in "${WAN_NODE_LIST[@]}"; do [ "$id" = "$node" ] || REMOTES+=("$id"); done
```

```
}
mesh_set_context "${OP_IDS[0]}"
[ "${#REMOTES[@]}" -ge 1 ] || { echo -e "${RED}Aucun pair distant.${NC}" >&2; exit 1; }

# Union des pairs de TOUS les operateurs locaux. mesh_set_context recalcule
# REMOTES a chaque appel : ce qui ne depend pas d'un operateur - l'ouverture du
# pare-feu, les compteurs affiches - lisait la liste du DERNIER operateur et
# oubliait les pairs des autres.
ALL_REMOTES=(); _seen=""
for _i in "${OP_IDS[@]}"; do
    mesh_set_context "$_i"
    for _r in "${REMOTES[@]}"; do
        case " $_seen " in
            *" $_r "*) ;;
            *) _seen="$_seen $_r"; ALL_REMOTES+=("$_r") ;;
        esac
    done
done
mesh_set_context "${OP_IDS[0]}"

run() { if [ "$DRY" -eq 1 ]; then echo "  [dry-run] $*"; else "$@"; fi; }

# -- Acces aux fichiers de conf, docker ou natif ------------------------------
# On tire le fichier sur l'hote, on le transforme avec awk/sed LOCALEMENT, on le
# repousse. C'est ce qui permet au meme code de servir les deux modes - et ca
# evite les sed multi-niveaux echappes dans un `docker exec bash -c` (l'endroit
# ou network/setup-wan-interop.sh est le plus fragile).
ast_pull() {  # $1=op  $2=chemin  → stdout
    if [ "$MODE" = "docker" ]; then docker exec "$(op_container "$1")" cat "$2" 2>/dev/null || true
    else cat "${NATIVE_ROOT}$2" 2>/dev/null || true; fi
}
ast_push() {  # $1=op  $2=chemin  ← stdin
    if [ "$DRY" -eq 1 ]; then cat >/dev/null; echo "  [dry-run] ecriture $2 (op $1)"; return 0; fi
    if [ "$MODE" = "docker" ]; then docker exec -i "$(op_container "$1")" bash -c "cat > '$2'"
    else mkdir -p "$(dirname "${NATIVE_ROOT}$2")"; cat > "${NATIVE_ROOT}$2"; fi
}
ast_cli() {   # $1=op  $2=commande CLI asterisk
    if [ "$MODE" = "docker" ]; then docker exec "$(op_container "$1")" asterisk -rx "$2" 2>/dev/null || true
    else asterisk -rx "$2" 2>/dev/null || true; fi
}

BEGIN_MARK="; ---- OSMO WAN MESH BEGIN ----"
END_MARK="; ---- OSMO WAN MESH END ----"
SMS_BEGIN="# ---- OSMO WAN MESH BEGIN ----"
SMS_END="# ---- OSMO WAN MESH END ----"

# Retire nos blocs ET l'ancien bloc "WAN INTEROP" (genere par
# setup-wan-interop.sh, toujours ajoute en fin de fichier) : les deux decrivent
# le meme trafic, les laisser cohabiter ferait dependre le routage de l'ordre
# des patterns.
strip_generated() { sed -e '/OSMO WAN MESH BEGIN/,/OSMO WAN MESH END/d' -e '/; ==.*WAN INTEROP/,$d'; }

# ============================================================================
# [1/6] Verification des cibles
# ============================================================================
echo -e "${GREEN}[1/6] Verification des cibles Asterisk...${NC}"
for i in "${OP_IDS[@]}"; do
    mesh_set_context "$i"
    if [ "$MODE" = "docker" ]; then
        docker ps --format '{{.Names}}' | qgrep "^$(op_container "$i")$" \
            || { echo -e "  ${RED}✗ $(op_container "$i") absent - lancez start-docker.sh d'abord${NC}"; exit 1; }
        echo -e "  ${GREEN}✓${NC} $(op_container "$i")"
    else
        [ -d "${NATIVE_ROOT}/etc/asterisk" ] || { echo -e "  ${RED}✗ ${NATIVE_ROOT}/etc/asterisk absent${NC}";
        ↪  exit 1; }
        echo -e "  ${GREEN}✓${NC} /etc/asterisk (natif, operateur ${i})"
```

```
    fi
done
echo ""

# ═══════════════════════════════════════════════════════════════════════
# [2/6] iptables - entree depuis chaque pair
# ═══════════════════════════════════════════════════════════════════════
echo -e "${GREEN}[2/6] iptables - ouverture entrante depuis ${#ALL_REMOTES[@]} pair(s)...${NC}"
if [ "$CONFIG_ONLY" -eq 1 ]; then
    echo -e "  ${YELLOW}--config-only : ni iptables ni firewall. Les ports WAN resteront fermes${NC}"
    echo -e "  ${YELLOW}tant que ce script n'aura pas ete relance sans --config-only.${NC}"
else
run sysctl -w net.ipv4.ip_forward=1 >/dev/null 2>&1 || true

if [ "$DRY" -eq 0 ]; then
    iptables -t nat -D PREROUTING -j "$CHAIN" 2>/dev/null || true
    iptables -t nat -F "$CHAIN" 2>/dev/null || true
    iptables -t nat -X "$CHAIN" 2>/dev/null || true
    iptables -t nat -N "$CHAIN"
fi

# L'operateur local est la boucle EXTERIEURE : mesh_set_context recalcule
# REMOTES, et prise dans l'autre sens la boucle des pairs lisait la liste laissee
# par le dernier appel - une seule vue servait tous les operateurs, si bien que
# les pairs des autres n'avaient aucune regle et que les seules sources presentes
# dans la chaine etaient celles d'un seul operateur.
for i in "${OP_IDS[@]}"; do
    mesh_set_context "$i"
    lip="$LOCAL_IP"
    sip=$(wan_sip_port "$i"); rs=$(wan_rtp_start "$i"); re=$(wan_rtp_end "$i")
    sms=$(wan_sms_port "$i")
    # Le filtre de destination, calcule une fois par operateur local.
    _dfil=()
    if addr_is_local "$lip"; then _dfil=(-d "$lip")
    else echo -e "  ${YELLOW}! ${lip} (table) n'est pas une adresse de cette machine : DNAT non restreint en
     ↪  destination${NC}"; fi
    # ── LE SENS SORTANT : un port source PAR CONTENEUR ──────────────────
    # Tous les conteneurs font tourner Asterisk sur le meme port, 5060. Vers un
    # pair du LAN ils sont masques derriere l'unique adresse de l'hote, et le
    # masquage leur a donne LE MEME port traduit :
    #     192.168.1.31.48057 > 192.168.1.2.5080   (venant de 172.20.0.11)
    #     192.168.1.31.48057 > 192.168.1.2.5080   (venant de 172.20.0.12)
    # Deux flux distincts, un seul tuple : les reponses du pair ne reviennent
    # qu'a l'un des deux, et le trunk de l'autre reste "Unavail" pour toujours.
    # Constate sur le banc, et invisible autrement - les deux conteneurs
    # emettent bien, seul le retour manque.
    #
    # On epingle donc un port source stable et distinct par operateur. Stable
    # aussi pour le pair : son identify voit toujours le meme couple
    # adresse/port pour un noeud donne, au lieu d'un port tire au hasard a
    # chaque expiration de conntrack.
    if [ "$MODE" = "docker" ] && [ "${#REMOTES[@]}" -gt 0 ]; then
        _snat_port=$(( 5060 + i ))
        _lan_net="$(printf '%s' "$lip" | cut -d. -f1-3).0/24"
        for _pr in udp tcp; do
            run iptables -t nat -D POSTROUTING -s "$(op_backbone_ip "$i")" -p "$_pr" --sport 5060 \
                -d "$_lan_net" -j SNAT --to-source "${lip}:${_snat_port}" 2>/dev/null || true
            run iptables -t nat -I POSTROUTING 1 -s "$(op_backbone_ip "$i")" -p "$_pr" --sport 5060 \
                -d "$_lan_net" -j SNAT --to-source "${lip}:${_snat_port}"
        done
        echo -e "  ${CYAN}Op${i}${NC} sort en SIP depuis ${CYAN}${lip}:${_snat_port}${NC} (port source epingle)"
        # Les entrees conntrack deja etablies ignorent la regle neuve : Asterisk
        # rafraichit son qualify toutes les 30 s, elles n'expirent jamais toutes
        # seules. On les efface, sinon le correctif ne prend qu'au prochain
        # redemarrage d'Asterisk - et l'on croit qu'il ne marche pas.
        if [ "$DRY" -eq 0 ] && command -v conntrack >/dev/null 2>&1; then
```

```
            conntrack -D -s "$(op_backbone_ip "$i")" -p udp >/dev/null 2>&1 || true
        fi
    fi

    for r in "${REMOTES[@]}"; do
        rip="${WAN_IP[$r]}"
        if [ "$MODE" = "docker" ]; then
            bb=$(op_backbone_ip "$i")
            # -d "$lip" : filtrer la seule source attrapait aussi ce que CE noeud
            # EMET vers le pair (meme source, meme port de destination). Le paquet
            # ne quittait jamais l'hote, il repartait vers notre propre conteneur
            # qui se repondait a lui-meme - le trunk affichait "Avail" alors que le
            # pair n'avait rien recu.
            run iptables -t nat -A "$CHAIN" -s "$rip" ${_dfil[@]+"${_dfil[@]}"} -p udp --dport "$sip" -j DNAT
             ↪  --to-destination "${bb}:5060"
            run iptables -t nat -A "$CHAIN" -s "$rip" ${_dfil[@]+"${_dfil[@]}"} -p tcp --dport "$sip" -j DNAT
             ↪  --to-destination "${bb}:5060"
            run iptables -t nat -A "$CHAIN" -s "$rip" ${_dfil[@]+"${_dfil[@]}"} -p udp --dport "${rs}:${re}" -j
             ↪  DNAT --to-destination "$bb"
            run iptables -t nat -A "$CHAIN" -s "$rip" ${_dfil[@]+"${_dfil[@]}"} -p tcp --dport "$sms" -j DNAT
             ↪  --to-destination "${bb}:${SMS_RELAY_BASE}"
        else
            # Natif : Asterisk et le relais ecoutent deja sur l'hote. Seule la
            # translation de port WAN → port de service est necessaire ; le RTP
            # arrive directement dans la plage posee par rtp.conf.
            run iptables -t nat -A "$CHAIN" -s "$rip" -p udp --dport "$sip" -j REDIRECT --to-ports 5060
            run iptables -t nat -A "$CHAIN" -s "$rip" -p tcp --dport "$sip" -j REDIRECT --to-ports 5060
            [ "$sms" = "$SMS_RELAY_BASE" ] || \
                run iptables -t nat -A "$CHAIN" -s "$rip" -p tcp --dport "$sms" -j REDIRECT --to-ports
                 ↪  "$SMS_RELAY_BASE"
        fi
        echo -e "  ${CYAN}node ${r}${NC} ${rip} → Op${i} SIP/RTP/SMS acceptes"
    done
done

if [ "$DRY" -eq 0 ]; then
    iptables -t nat -I PREROUTING -j "$CHAIN"
    if [ "$MODE" = "docker" ]; then
        # --ctstate DNAT : sans restriction, ce masquage prenait AUSSI les flux du
        # LAN simplement routes vers un conteneur ; celui-ci repondait alors a
        # l'hote et non au client, qui n'a jamais vu la reponse. On ne masque donc
        # que ce que notre propre DNAT a redirige. Restreindre plutot sur l'adresse
        # de l'hote ne matcherait rien : apres DNAT la source est celle du pair.
        # L'ancienne regle large est retiree d'abord - le garde -C teste la spec
        # exacte, sans cela les deux resteraient en place et la large gagnerait.
        iptables -t nat -D POSTROUTING -d 172.20.0.0/24 -j MASQUERADE 2>/dev/null || true
        if ! iptables -t nat -C POSTROUTING -d 172.20.0.0/24 -m conntrack --ctstate DNAT -j MASQUERADE
         ↪  2>/dev/null; then
            # Sans le module conntrack d'iptables, le -A echoue et, sous set -e,
            # le script mourait ICI : chaine deja inseree dans PREROUTING et
            # regle large deja retiree, mais ni rtp.conf, ni pjsip, ni dialplan,
            # ni routes SMS. On repose alors la regle large en le disant - moins
            # precise, mais un maillage complet vaut mieux qu'un demi-maillage.
            iptables -t nat -A POSTROUTING -d 172.20.0.0/24 -m conntrack --ctstate DNAT -j MASQUERADE 2>/dev/null
             ↪  || {
               echo -e "  ${YELLOW}! -m conntrack indisponible : masquage large 172.20.0.0/24 (moins precis)${NC}"
                iptables -t nat -C POSTROUTING -d 172.20.0.0/24 -j MASQUERADE 2>/dev/null \
                    || iptables -t nat -A POSTROUTING -d 172.20.0.0/24 -j MASQUERADE || true
            }
        fi
    fi
fi

# Firewall INPUT - sans ca le DNAT redirige un trafic que l'INPUT jette.
for r in "${ALL_REMOTES[@]}"; do
    if [ -f "$SCRIPT_DIR/firewall-wan.sh" ]; then
```

```
        run bash "$SCRIPT_DIR/firewall-wan.sh" "${WAN_IP[$r]}" "$N_OPS" >/dev/null 2>&1 \
            && echo -e "  ${GREEN}✓${NC} firewall ouvert pour ${WAN_IP[$r]}" \
            || echo -e "  ${YELLOW}⚠${NC} firewall-wan.sh a echoue pour ${WAN_IP[$r]}"
    fi
done
fi
echo ""

# ═══════════════════════════════════════════════════════════════════
# [3/6] rtp.conf - plage RTP WAN par operateur
# ═══════════════════════════════════════════════════════════════════
echo -e "${GREEN}[3/6] rtp.conf...${NC}"
for i in "${OP_IDS[@]}"; do
    mesh_set_context "$i"
    rs=$(wan_rtp_start "$i"); re=$(wan_rtp_end "$i")
    printf '[general]\nrtpstart=%s\nrtpend=%s\nstrictrtp=no\nicesupport=no\n' "$rs" "$re" \
        | ast_push "$i" /etc/asterisk/rtp.conf
    echo -e "  ${CYAN}Op${i}${NC} RTP ${rs}-${re}"
done
echo ""

# ═══════════════════════════════════════════════════════════════════
# [4/6] PJSIP - un trunk par (noeud distant, operateur distant)
# ═══════════════════════════════════════════════════════════════════
echo -e "${GREEN}[4/6] PJSIP - trunks WAN...${NC}"

for i in "${OP_IDS[@]}"; do
    mesh_set_context "$i"
    pj="$(mktemp)"; ast_pull "$i" /etc/asterisk/pjsip.conf | strip_generated > "$pj"

    # external_media_address : l'IP que le SDP annonce a l'autre bout. Elle DOIT
    # etre l'IP publique de CE noeud. Le gabarit configs/pjsip.conf en pose deja
    # une (l'IP docker/hote) : on l'ECRASE, on ne se contente pas de l'ajouter
    # quand elle manque - sinon le WAN annonce une adresse privee et l'audio
    # part dans le vide alors que la signalisation, elle, passe. C'est ce que
    # fait network/setup-wan-interop.sh ("if ! qgrep external_media_address"),
    # et ca se voit seulement a l'oreille : appel decroche, silence.
    awk -v ip="$LOCAL_IP" '
        /^\[/ { if (intr && !done) { emit(); done=1 } ; intr = ($0 == "[transport-udp]") ; print ; next }
        intr && /^external_(media|signaling)_address=/ { next }
        intr && /^local_net=/ { seen_net=1 ; print ; next }
        intr && /^[[:space:]]*$/ && !done { emit(); done=1; intr=0; print; next }
        { print }
        END { if (intr && !done) emit() }
        function emit() {
            print "external_media_address=" ip
            print "external_signaling_address=" ip
            if (!seen_net) { print "local_net=172.20.0.0/16"; print "local_net=127.0.0.0/8" }
        }
    ' "$pj" > "${pj}.new" && mv "${pj}.new" "$pj"

    {
        echo "$BEGIN_MARK"
        echo "; Trunks WAN - generes par network/setup-wan-mesh.sh"
        echo "; Noeud local ${WAN_NODE_ID} (${LOCAL_IP}, indicatif ${LOCAL_IND})"
        for r in "${REMOTES[@]}"; do
            rip="${WAN_IP[$r]}"; rind="${WAN_IND[$r]}"; rnops="$(node_nops "$r")"
            # UN identify par adresse distante : identify ne regarde que l'adresse
            # source, jamais le port. Un bloc par operateur distant en posait
            # plusieurs pour la MEME adresse, et l'appel entrant etait attribue a
            # l'un d'eux au hasard - donc parfois au mauvais contexte d'arrivee.
            cat <<EOF

; ── node ${r} (indicatif ${rind}) · ${rip} · ${rnops} operateur(s) ──
[wan-id-n${r}]
type=identify
```

```
endpoint=wan_n${r}_op1
match=${rip}
EOF
            for j in $(seq 1 "$rnops"); do
                rsip=$(peer_sip_port "$r" "$j")
                # Pair sur le backbone : le port publie sur l'hote n'est pas
                # traverse dans le bridge, c'est 5060 que le conteneur ecoute.

                cat <<EOF

; ── node ${r} · operateur ${j} · ${rip}:${rsip} ──
[wan_n${r}_op${j}]
type=endpoint
transport=transport-udp
context=wan_in
disallow=all
allow=gsm
allow=ulaw
aors=wan_n${r}_op${j}
media_encryption=no
direct_media=no
rtp_symmetric=yes
force_rport=yes
rewrite_contact=yes
ice_support=no

[wan_n${r}_op${j}]
type=aor
contact=sip:${rip}:${rsip}
qualify_frequency=30
qualify_timeout=5.0
EOF
            done
        done
        # La passerelle du bridge une seule fois, hors de la boucle : le trafic
        # WAN entrant est masque par l'hote et arrive donc avec 172.20.0.1 pour
        # source. Repetee dans chaque bloc, elle donnait autant d'identify que de
        # couples (noeud, operateur) pour cette unique adresse.
        if [ "$MODE" = "docker" ]; then
            # Le trafic qui arrive masque en 172.20.0.1 est celui que l'hote a
            # route depuis le LAN : il ne peut donc venir QUE d'un pair hors du
            # backbone. Prendre REMOTES[0] attribuait ce trafic au premier pair
            # de la liste - souvent un conteneur voisin, qui n'arrive jamais
            # masque : on passait d'une attribution au hasard a une attribution
            # systematiquement fausse.
            _gwr=""
            for _g in "${REMOTES[@]}"; do
                peer_is_backbone "${WAN_IP[$_g]}" || { _gwr="$_g"; break; }
            done
            if [ -n "$_gwr" ]; then
                cat <<EOF

; ── passerelle du bridge - trafic WAN masque par l'hote (noeud ${_gwr}) ──
[wan-id-gw]
type=identify
endpoint=wan_n${_gwr}_op1
match=172.20.0.1
EOF
            fi
        fi
        echo "$END_MARK"
    } >> "$pj"

    ast_push "$i" /etc/asterisk/pjsip.conf < "$pj"
    rm -f "$pj"
    ntrunk=0
```

```
    for r in "${REMOTES[@]}"; do ntrunk=$(( ntrunk + $(node_nops "$r") )); done
    echo -e "  ${CYAN}Op${i}${NC} ${ntrunk} trunk(s) WAN"
done
echo ""

# ═══════════════════════════════════════════════════════════════════════
# [5/6] Dialplan - [wan_out] par indicatif, [wan_in], entrees gsm_in/internal
# ═══════════════════════════════════════════════════════════════════════
echo -e "${GREEN}[5/6] Dialplan - routage par indicatif...${NC}"

for i in "${OP_IDS[@]}"; do
    mesh_set_context "$i"
    ext="$(mktemp)"; ast_pull "$i" /etc/asterisk/extensions.conf | strip_generated > "$ext"

    # Softphones : tout endpoint avec "callerid=... <NNN>" devient joignable
    # par le WAN, sans coder aucun numero en dur.
    softphones=$(ast_pull "$i" /etc/asterisk/pjsip.conf | awk '
        /^\[/                 { s=$0; gsub(/[][]/,"",s); ep=s }
        /callerid=.*<[0-9]+>/ { n=$0; sub(/.*</,"",n); sub(/>.*/,"",n); print n":"ep }')

    body="$(mktemp)"
    {
        echo "$BEGIN_MARK"
        echo "; Dialplan WAN - network/setup-wan-mesh.sh"
        echo "; Composer <indicatif><numero> ; l'indicatif est retire a la sortie."
        for r in "${REMOTES[@]}"; do
            printf ';   %-4s → node %s (%s)\n' "${WAN_IND[$r]}" "$r" "${WAN_IP[$r]}"
        done
        echo ""
        echo "[wan_out]"
        for r in "${REMOTES[@]}"; do
            rind="${WAN_IND[$r]}"; off="${#rind}"
            echo ""
            echo "; ── node ${r} - indicatif ${rind} (strip ${off}) ──────────────────"
            # Les operateurs du noeud DISTANT, pas les notres : compter les notres
            # produisait un _<indicatif>2XXXX vers un operateur que ce noeud-la
            # n'a pas - l'appel sortait et se perdait au lieu d'etre refuse ici.
            # Le numero compose : <indicatif du noeud> + le MSISDN a SIX chiffres.
            # Pour joindre l'operateur 1 du noeud 1 depuis ici : 11 100101.
            # Le MSISDN commence par le numero du noeud DISTANT (<noeud>00<op>
            # <rang>) : c'est "${r}00${j}" et non un "600" commun. Fige a 600,
            # le motif n'accrochait plus aucun numero du pair - l'appel tombait
            # dans le fourre-tout, Asterisk repondait "successful" a l'envoi SMS
            # et le message n'allait nulle part.
            # Un seul motif suffit, la longueur etant fixe.
            for j in $(seq 1 "$(node_nops "$r")"); do
                cat <<EOF
exten => _${rind}${r}00${j}XX,1,NoOp(=== WAN OUT node${r} op${j}: \${EXTEN} → ${WAN_IP[$r]} ===)
 same => n,Dial(PJSIP/\${EXTEN:${off}}@wan_n${r}_op${j},,rT)
 same => n,NoOp(WAN \${DIALSTATUS})
 same => n,Congestion()
 same => n,Hangup()
EOF
            done
            # L'operateur vise chez le pair, borne a ce qu'il porte reellement :
            # cette ligne prenait l'indice de l'operateur LOCAL. Vers un noeud
            # qui n'a qu'un operateur, elle appelait wan_n<r>_op2 - un endpoint
            # qui n'est plus declare depuis que les trunks suivent node_nops :
            # Asterisk journalise "endpoint not found" au lieu d'essayer.
            _spop="$i"; [ "$_spop" -le "$(node_nops "$r")" ] || _spop=1
            for sp in $softphones; do
                sp_num="${sp%%:*}"
                cat <<EOF
exten => ${rind}${sp_num},1,NoOp(=== WAN OUT node${r} softphone ${sp_num} ===)
 same => n,Dial(PJSIP/${sp_num}@wan_n${r}_op${_spop},,rT)
 same => n,Congestion()
```

```
 same => n,Hangup()
EOF
            done
        done
        cat <<EOF

; Indicatif local (${LOCAL_IND}) compose depuis ce noeud : c'est chez nous.
; On retire l'indicatif et on repart dans le routage local plutot que de
; sortir sur le WAN pour revenir - un aller-retour qui, lui, ne raccroche pas.
exten => _${LOCAL_IND}X.,1,NoOp(=== WAN: indicatif local, routage interne \${EXTEN} ===)
 same => n,Goto(gsm_in,\${EXTEN:${#LOCAL_IND}},1)

exten => _X.,1,NoOp(=== WAN OUT: indicatif inconnu \${EXTEN} ===)
 same => n,Congestion()
 same => n,Hangup()

; ── [wan_in] - ce qui ARRIVE des autres noeuds ──────────────────────────
; Le noeud emetteur a deja retire l'indicatif : on recoit le numero nu.
[wan_in]
EOF
        # L'indicatif a ete retire par le noeud appelant : ce qui arrive ici est
        # le MSISDN nu, six chiffres. Ce sont NOS numeros, donc ils commencent
        # par NOTRE numero de noeud : <WAN_NODE_ID>00<operateur><rang>.
        for j in "${OP_IDS[@]}"; do
            cat <<EOF
exten => _${WAN_NODE_ID}00${j}XX,1,NoOp(=== WAN IN → GSM Op${j}: \${EXTEN} ===)
 same => n,Set(CALLERID(all)=<\${CALLERID(num)}>)
 same => n,Gosub(sub-annuaire,\${CALLERID(num)},1)
 same => n,Gosub(sub-record,s,1(\${EXTEN}))
 same => n,Dial(PJSIP/\${EXTEN}@gsm_msc,,rT)
 same => n,Congestion()
 same => n,Hangup()
EOF
        done
        for sp in $softphones; do
            sp_num="${sp%%:*}"; sp_ep="${sp#*:}"
            cat <<EOF
exten => ${sp_num},1,NoOp(=== WAN IN → ${sp_ep} (${sp_num}) ===)
 same => n,Gosub(sub-record,s,1(${sp_num}))
 same => n,Dial(PJSIP/${sp_ep},,rT)
 same => n,Congestion()
 same => n,Hangup()
EOF
        done
        cat <<EOF

; Un pair qui n'aurait pas retire notre indicatif : on le retire ici.
exten => _${LOCAL_IND}X.,1,NoOp(=== WAN IN: indicatif local encore present ===)
 same => n,Goto(wan_in,\${EXTEN:${#LOCAL_IND}},1)

; Autre operateur de CE noeud : on repasse par le routage inter-operateur local.
exten => _X.,1,NoOp(=== WAN IN → routage local \${EXTEN} ===)
 same => n,Gosub(sub-record,s,1(\${EXTEN}))
 same => n,Goto(interop_out,\${EXTEN},1)

exten => 600,1,NoOp(=== WAN ECHO TEST ===)
 same => n,Answer()
 same => n,Playback(demo-echotest)
 same => n,Echo()
 same => n,Playback(demo-echodone)
 same => n,Hangup()
EOF
        echo "$END_MARK"
    } > "$body"

    # Entrees : un prefixe par indicatif distant, injecte dans [gsm_in] (appels
```

```
    # venus du GSM) et [internal] (softphones SIP).
    entry="$(mktemp)"
    {
        echo "$BEGIN_MARK"
        for r in "${REMOTES[@]}"; do
            printf 'exten => _%s.,1,NoOp(=== indicatif %s → node %s ===)\n' "${WAN_IND[$r]}" "${WAN_IND[$r]}" "$r"
            printf ' same => n,Goto(wan_out,${EXTEN},1)\n'
        done
        # Notre PROPRE indicatif compose en entier : il doit aboutir ici, pas
        # tomber dans le vide. wan_out le retire et rend la main au routage local.
        printf 'exten => _%s.,1,NoOp(=== indicatif local %s ===)\n' "$LOCAL_IND" "$LOCAL_IND"
        printf ' same => n,Goto(wan_out,${EXTEN},1)\n'
        echo "$END_MARK"
    } > "$entry"

    awk -v bf="$entry" '
        { print }
        $0 == "[gsm_in]" || $0 == "[internal]" {
            while ((getline l < bf) > 0) print l
            close(bf)
        }
    ' "$ext" > "${ext}.new" && mv "${ext}.new" "$ext"

    cat "$body" >> "$ext"
    ast_push "$i" /etc/asterisk/extensions.conf < "$ext"
    rm -f "$ext" "$body" "$entry"
    _inds=""; for r in "${REMOTES[@]}"; do _inds="${_inds}${_inds:+ }${WAN_IND[$r]}→node${r}"; done
    echo -e "  ${CYAN}Op${i}${NC} wan_out/wan_in · ${_inds}"
done
echo ""

# ═══════════════════════════════════════════════════════════════════
# [6/6] SMS - sms-routing.conf : un operateur distant par (noeud, operateur)
# ═══════════════════════════════════════════════════════════════════
# Le relais (scripts/sms-interop-relay.py) route par LONGEST-PREFIX sur le
# MSISDN compose. On lui donne donc les prefixes <indicatif><op>, la cible
# ip:port du relais distant, et le nombre de chiffres d'indicatif a RETIRER
# avant l'envoi : le HLR distant ne connait que le numero nu.
echo -e "${GREEN}[6/6] Routage SMS WAN...${NC}"
for i in "${OP_IDS[@]}"; do
    mesh_set_context "$i"
    conf="$(mktemp)"
    ast_pull "$i" /etc/osmocom/sms-routing.conf \
        | sed '/OSMO WAN MESH BEGIN/,/OSMO WAN MESH END/d' \
        | sed '/# ── WAN ROUTES/,/# ── FIN WAN/d' > "$conf"
    if [ ! -s "$conf" ]; then
        echo -e "  ${YELLOW}⚠${NC} Op${i} : sms-routing.conf absent - SMS WAN non configure"
        rm -f "$conf"; continue
    fi
    {
        echo ""
        echo "$SMS_BEGIN"
        echo "# Noeuds WAN distants. Cle = <w><node><op>, valeur = ip:port du relais."
        echo "[operators]"
        for r in "${REMOTES[@]}"; do
            rip="${WAN_IP[$r]}"
            for j in $(seq 1 "$(node_nops "$r")"); do
                # Vers un pair du backbone, le relais distant est joint dans le
                # bridge : son port publie sur l'hote (7891 pour l'operateur 2)
                # n'y mene pas, il ecoute le port de base.
                rsms="$(peer_sms_port "$r" "$j")"

                printf 'w%s%s = %s:%s\n' "$r" "$j" "$rip" "$rsms"
            done
        done
        echo ""
```

```
        # La cle de routage est le PREFIXE compose :
        # <indicatif><noeud distant>00<operateur>. Le "600" fixe qu'elle portait
        # ne correspondait plus a aucun numero depuis que le MSISDN commence par
        # le noeud : un SMS vers 11100101 ne matchait aucune route, le relais
        # rendait un succes et le message n'allait nulle part.
        # strip retire l'indicatif seul, ce qui laisse le MSISDN entier.
        echo "# <indicatif><noeud>00<op> → noeud distant ; strip = chiffres d'indicatif a retirer."
        echo "[routes]"
        for r in "${REMOTES[@]}"; do
            for j in $(seq 1 "$(node_nops "$r")"); do
                printf '%s%s00%s = w%s%s strip=%s\n' "${WAN_IND[$r]}" "$r" "$j" "$r" "$j" "${#WAN_IND[$r]}"
            done
        done
        echo "$SMS_END"
    } >> "$conf"
    ast_push "$i" /etc/osmocom/sms-routing.conf < "$conf"
    rm -f "$conf"
    nsms=0
    for r in "${REMOTES[@]}"; do nsms=$(( nsms + $(node_nops "$r") )); done
    echo -e "  ${CYAN}Op${i}${NC} ${nsms} route(s) SMS WAN"
done
echo ""

# ── Redemarrage Asterisk ──────────────────────────────────────────
# Le transport PJSIP a change : un "reload" ne suffit pas. En natif au moment
# du bootstrap Asterisk n'est pas encore lance - on ne redemarre alors rien, la
# conf sera lue a son demarrage.
if [ "$NO_RESTART" -eq 1 ]; then
    echo -e "${YELLOW}--no-restart : Asterisk gardera l'ancienne conf jusqu'a son prochain demarrage.${NC}"
elif [ "$DRY" -eq 1 ]; then
    echo "  [dry-run] redemarrage Asterisk"
else
    for i in "${OP_IDS[@]}"; do
        mesh_set_context "$i"
        if [ "$MODE" = "docker" ]; then
            docker exec "$(op_container "$i")" bash -c "
                asterisk -rx 'core stop now' 2>/dev/null || pkill asterisk 2>/dev/null || true
                sleep 2; pkill -9 asterisk 2>/dev/null || true; sleep 1
                asterisk -f & disown" 2>/dev/null || true
            echo -e "  ${CYAN}Op${i}${NC} Asterisk redemarre"
        else
            if asterisk -rx 'core show uptime' 2>/dev/null | qgrep -i uptime; then
                asterisk -rx 'core restart now' >/dev/null 2>&1 || true
                echo -e "  ${CYAN}Op${i}${NC} Asterisk redemarre"
            else
                echo -e "  ${CYAN}Op${i}${NC} Asterisk non lance - conf prete pour son demarrage"
            fi
        fi
    done
fi

# ── Sauvegarde + resume ───────────────────────────────────────────
if [ "$DRY" -eq 0 ]; then
    # Defaut 0 : appliquer une table une fois ne doit pas la rendre permanente.
    # Seule une ISO construite avec --wan fige WAN_AUTO=1 (iso_modules/70-scripts.sh).
    WAN_AUTO="${WAN_AUTO:-0}" wan_nodes_save "${CONF:-$WAN_CONF_FILE}"
fi

echo ""
echo -e "${GREEN}${BOLD}WAN mesh applique.${NC}"
echo ""
# Il faut composer l'indicatif du noeud SUIVI DU MSISDN entier, six chiffres :
# <noeud>00<operateur><rang>. Un exemple faux dans le resume final vaut une
# heure perdue a essayer un numero qui ne matche aucun motif - le resume a deja
# annonce "<indicatif>10001" puis "<indicatif>600101", deux plans revolus.
# LOCAL_IND et REMOTES gardent le contexte du DERNIER operateur boucle plus
```

```
# haut ; on repose donc le notre avant d'afficher, sinon la ligne annonce un
# indicatif qui n'est pas celui de ce noeud.
mesh_set_context "${OP_IDS[0]}"
echo -e "  ${BOLD}Depuis ce noeud (${WAN_NODE_ID}, indicatif ${LOCAL_IND}) :${NC}"
for r in "${REMOTES[@]}"; do
    for j in $(seq 1 "$(node_nops "$r")"); do
        echo -e "    ${CYAN}${WAN_IND[$r]}${NC}${r}00${j}01  → MS ${r}00${j}01 (op${j}) du noeud ${r}
        ↪  (${WAN_IP[$r]})"
    done
done
echo ""
echo -e "  ${BOLD}Ports WAN (identiques sur chaque noeud) :${NC}"
for i in "${OP_IDS[@]}"; do
    mesh_set_context "$i"
    echo -e "    Op${i}: SIP ${CYAN}$(wan_sip_port "$i")${NC}/udp  RTP ${CYAN}$(wan_rtp_start "$i")-$(wan_rtp_end
    ↪  "$i")${NC}/udp  SMS ${CYAN}$(wan_sms_port "$i")${NC}/tcp"
done
echo ""
echo -e "  Table sauvegardee : ${CYAN}${CONF:-$WAN_CONF_FILE}${NC}"
echo -e "  Diagnostic SS7/WAN : ${CYAN}checks/wan_ss7_check.sh${NC}"
echo ""
```

3.258 /opt/GSM/osmo-operator/network/setup-wan-sms.sh

4138 octets, 98 lignes → 98 lignes

```
#!/bin/bash
# ═══════════════════════════════════════════════════════════════════
# network/setup-wan-sms.sh - Routage SMS inter-serveur
#
# Ajoute le routage SMS pour le prefixe 66 dans la table sms-routing.conf
# de chaque operateur. Les SMS a destination de 66XXXXX sont relayes
# via TCP vers le sms-interop-relay.py du serveur distant.
#
# Usage :
#   sudo ./network/setup-wan-sms.sh <remote_public_ip> [n_operators]
#
# Executer sur CHAQUE serveur.
# ═══════════════════════════════════════════════════════════════════

set -euo pipefail

GREEN='\033[0;32m'; CYAN='\033[0;36m'; NC='\033[0m'

REMOTE_IP="${1:-}"
N_OPS="${2:-2}"
WAN_PREFIX="66"
RELAY_PORT=7890
CONTAINER_PREFIX="osmo-operator-"

if [ -z "$REMOTE_IP" ]; then
    echo "Usage: sudo $0 <remote_public_ip> [n_operators]"
    exit 1
fi

echo -e "${GREEN}Configuration SMS WAN → ${REMOTE_IP}${NC}"
echo ""

for i in $(seq 1 "$N_OPS"); do
    cname="${CONTAINER_PREFIX}${i}"

    echo -e "  ${CYAN}Op${i}${NC} - Mise a jour sms-routing.conf..."

    # Ajouter les routes WAN dans sms-routing.conf
```

```
    docker exec "$cname" bash -c "
        CONF='/etc/osmocom/sms-routing.conf'

        # Retirer l'ancien bloc WAN s'il existe
        sed -i '/# ── WAN ROUTES/,/# ── FIN WAN/d' \"\$CONF\" 2>/dev/null || true

        # Ajouter les operateurs distants dans [operators]
        # On utilise des IDs 100+N pour les operateurs distants
        cat >> \"\$CONF\" << SMSEOF

# ── WAN ROUTES - Serveur distant ${REMOTE_IP} ──────────────────────────
# Les IDs 100+ designent les operateurs sur le serveur distant.
# Le relay utilise l'IP du serveur distant pour la connexion TCP.

SMSEOF

        # Ajouter dans la section [operators] si pas deja present
        for j in \$(seq 1 ${N_OPS}); do
            wan_op_id=\$((100 + j))
            if ! grep -q \"^\${wan_op_id} =\" \"\$CONF\"; then
                # Inserer apres la derniere ligne de [operators]
                sed -i \"/^\\[operators\\]/,/^\\[/ {
                    /^\\[routes\\]/i \${wan_op_id} = ${REMOTE_IP}
                }\" \"\$CONF\" 2>/dev/null || true
            fi
        done

        # Ajouter les routes pour le prefixe WAN
        cat >> \"\$CONF\" << SMSEOF2
# Routes WAN : prefixe ${WAN_PREFIX}NXXXX → operateur distant N
SMSEOF2

        for j in \$(seq 1 ${N_OPS}); do
            wan_op_id=\$((100 + j))
            # Route prefixe : 66 + operateur + numero
            echo \"${WAN_PREFIX}\${j}0000 = \${wan_op_id}\" >> \"\$CONF\"
            for ms in \$(seq 1 8); do
                # <noeud>00<op><ms> : ici j numerote le pair distant, donc il
                # est a la fois son noeud et son operateur (un operateur par
                # noeud sur ce chemin). Fige a 600, le numero construit ne
                # correspondait a aucun abonne du pair.
                msisdn=\$(( j * 100000 + j * 100 + ms ))
                echo \"${WAN_PREFIX}\${msisdn} = \${wan_op_id}\" >> \"\$CONF\"
            done
        done

        echo '# ── FIN WAN ──────────────────────────────────────────────────' >> \"\$CONF\"
    "

    echo -e "  ${GREEN}✓${NC} Op${i} SMS WAN routes ajoutees"
done

echo ""
echo -e "${GREEN}SMS WAN routing configure.${NC}"
echo -e "  SMS vers ${CYAN}${WAN_PREFIX}10001${NC} → serveur distant Op1 MS1"
echo -e "  SMS vers ${CYAN}${WAN_PREFIX}20001${NC} → serveur distant Op2 MS1"
echo ""
echo -e "  ${CYAN}Note :${NC} Le sms-interop-relay.py doit etre accessible"
echo -e "  depuis l'exterieur sur le port TCP ${RELAY_PORT}."
echo -e "  Firewall : ufw allow from ${REMOTE_IP} to any port ${RELAY_PORT} proto tcp"
```

3.259 /opt/GSM/osmo-operator/network/wan-nodes.sh

20505 octets, 412 lignes → 412 lignes

```
#!/bin/bash
# ============================================================================
# network/wan-nodes.sh - la TABLE DES NOEUDS WAN, partagee par les 3 lanceurs
# ============================================================================
#
# Ce fichier ne definit que des fonctions et des variables : il est SOURCE par
#   start-docker.sh          (hote docker, N operateurs par noeud)
#   start-native.sh   (natif / ISO, 1 operateur par noeud)
#   build-iso.sh      (fige la table dans l'ISO)
#   network/setup-wan-mesh.sh  (l'applique a Asterisk / iptables / SMS)
#
# -- MODELE ------------------------------------------------------------------
# Un WAN = N noeuds. Un noeud = (id, ip publique, indicatif).
#
# L'INDICATIF est le prefixe telephonique du noeud, exactement comme un
# indicatif pays : depuis n'importe quel noeud, composer
#       <indicatif_cible><numero>
# joint <numero> sur le noeud qui porte cet indicatif. Le noeud emetteur
# retire l'indicatif avant de sortir sur le WAN ; le noeud d'arrivee recoit
# donc le numero nu, celui que son HLR et son dialplan connaissent deja.
#
# WAN_NODE_ID dit QUI est le noeud local. C'est la seule chose qui distingue
# "ce numero est chez moi" de "ce numero part sur le WAN" : sans lui, un
# noeud se renvoie ses propres appels a l'infini.
#
# -- POURQUOI UNE TABLE ET PAS UN COUPLE local/remote ------------------------
# network/setup-wan-interop.sh (l'existant) ne connait QUE deux serveurs :
# LOCAL_IP et REMOTE_IP. Il reecrit a chaque appel le bloc "WAN INTEROP" de
# pjsip.conf/extensions.conf et remet a zero sa chaine iptables - le lancer en
# boucle sur trois pairs ne laisse donc que le DERNIER. D'ou cette table : on
# genere UN bloc qui contient tous les pairs, en un seul passage.
# ----------------------------------------------------------------------------

# Table : WAN_IP[id] / WAN_IND[id] / WAN_NOPS[id], plus la liste ordonnee des ids.
declare -A WAN_IP  2>/dev/null || true
declare -A WAN_IND 2>/dev/null || true
# WAN_NOPS[id] : combien d'operateurs porte le noeud id. WAN_OPS ne parle que du
# noeud LOCAL, donc tout ce qui genere le maillage devait SUPPOSER les autres
# identiques a soi. L'hote docker en porte deux, une VM un seul : depuis l'hote,
# on fabriquait donc vers la VM un trunk, un pattern _<indicatif>2XXXX et une
# route SMS que personne ne sert - de quoi laisser partir des appels qui
# n'arrivent nulle part. D'ou ce nombre par noeud, quatrieme champ de la table.
declare -A WAN_NOPS 2>/dev/null || true
WAN_NODE_LIST=()
: "${WAN_NODE_COUNT:=0}"
: "${WAN_NODE_ID:=0}"
: "${WAN_OPS:=1}"          # operateurs (PLMN) portes par CE noeud
: "${WAN_CONF_FILE:=/etc/osmo-wan.conf}"
# WAN_DOCKER_HOST : l'adresse LAN de l'hote qui porte docker, celle par laquelle
# une VM route vers le backbone des conteneurs. Elle est en DHCP et a deja
# change sous nos pieds (.101 puis .31) : elle se DECOUVRE a l'execution
# (network/set_ip_vm.sh) et ne se fige jamais ici - la table ne fait que la
# transporter d'une machine a l'autre.
: "${WAN_DOCKER_HOST:=}"

_wanc() { [ -t 1 ] && printf '\033[0;36m%s\033[0m' "$1" || printf '%s' "$1"; }
_wan_warn() { printf '  \033[1;33m⚠\033[0m %s\n' "$*" >&2; }
_wan_err()  { printf '  \033[0;31m✗\033[0m %s\n' "$*" >&2; }

# -- Indicatifs par defaut ---------------------------------------------------
# 11/22/.../99 : un doublet par noeud, aucun n'est prefixe d'un autre (condition
# dure, cf. wan_nodes_validate) et il y en a exactement neuf - la borne des
# noeuds. Le doublet se lit aussi comme le numero du noeud, ce qui rend un
# journal d'appel dechiffrable sans la table sous les yeux.
_WAN_DEFAULT_IND=(11 22 33 44 55 66 77 88 99)
wan_default_ind() {
```

```
    local i=$1
    if [ "$i" -ge 1 ] && [ "$i" -le "${#_WAN_DEFAULT_IND[@]}" ]; then
        echo "${_WAN_DEFAULT_IND[$((i-1))]}"
    else
        echo $(( 100 + i ))
    fi
}

# ── Serialisation "id:ip:indicatif[:operateurs] ..." ──────────────────────
# Le nombre d'operateurs doit repasser par ici, sinon il ne survit pas au
# premier aller-retour : wan_nodes_save serialise avec cette fonction, et une
# table relue annoncait alors 1 operateur partout, y compris pour le noeud qui
# venait d'en declarer deux. Il n'est ecrit que lorsqu'il dit autre chose que 1,
# pour que tout ce qui lit deja WAN_NODES a trois champs (build-iso.sh,
# set_stp_ip.sh, les diagnostics) continue de voir la table qu'il connait.
wan_nodes_spec() {
    local id out="" nops
    for id in "${WAN_NODE_LIST[@]}"; do
        out="${out}${out:+ }${id}:${WAN_IP[$id]}:${WAN_IND[$id]}"
        nops="${WAN_NOPS[$id]:-1}"
        [ "$nops" = "1" ] || out="${out}:${nops}"
    done
    printf '%s' "$out"
}

wan_nodes_parse() {
    local spec="${1:-}" entry id rest ip ind nops
    WAN_IP=(); WAN_IND=(); WAN_NOPS=(); WAN_NODE_LIST=()
    spec="${spec//,/ }"
    for entry in $spec; do
        [ -n "$entry" ] || continue
        case "$entry" in
            *:*:*) ;;
            *) _wan_err "entree noeud invalide : '$entry' (attendu id:ip:indicatif[:operateurs])"; return 1 ;;
        esac
        id="${entry%%:*}"; rest="${entry#*:}"
        ip="${rest%%:*}";  ind="${rest#*:}"
        ind="${ind%%:*}"
        # Quatrieme champ optionnel. Toutes les tables ecrites avant lui - celle
        # de l'ISO, celle deja posee dans /etc/osmo-wan.conf - n'ont que trois
        # champs : elles doivent continuer de se lire, un noeud sans nombre
        # portant un operateur, ce qui est le cas d'une VM.
        nops=1
        case "$entry" in
            *:*:*:*) nops="${rest#*:}"; nops="${nops#*:}"; nops="${nops%%:*}" ;;
        esac
        [ -n "$nops" ] || nops=1
        # Un champ present mais illisible n'est pas rabattu en silence sur 1 :
        # ce serait un maillage prive des operateurs 2..n sans un mot dans le
        # journal - le genre de panne qu'on cherche longtemps.
        if ! [[ "$nops" =~ ^[1-9][0-9]*$ ]]; then
            _wan_err "noeud $id : '$nops' operateurs - attendu un entier >= 1"; return 1
        fi
        WAN_IP[$id]="$ip"; WAN_IND[$id]="$ind"; WAN_NOPS[$id]="$nops"
        WAN_NODE_LIST+=("$id")
    done
    WAN_NODE_COUNT=${#WAN_NODE_LIST[@]}
    [ "$WAN_NODE_COUNT" -gt 0 ] || { _wan_err "table de noeuds vide"; return 1; }
    return 0
}

# ── Validation ────────────────────────────────────────────────────────────
# Refuse ce qui produirait un routage silencieusement faux, pas seulement ce
# qui est mal ecrit.
wan_nodes_validate() {
    local id other rc=0
```

```
    [ "${#WAN_NODE_LIST[@]}" -ge 1 ] || { _wan_err "table de noeuds vide"; return 1; }
    for id in "${WAN_NODE_LIST[@]}"; do
        # 1..9 : un chiffre. C'est ce qui garde <indicatif><operateur><MSISDN>
        # lisible et les patterns Asterisk sans ambiguite de longueur.
        [[ "$id" =~ ^[1-9]$ ]] \
            || { _wan_err "numero de noeud invalide : '$id' (1 a 9)"; rc=1; }
        [[ "${WAN_IP[$id]}" =~ ^[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+$ ]] \
            || { _wan_err "noeud $id : '${WAN_IP[$id]}' n'est pas une IPv4"; rc=1; }
        # -- L'INDICATIF NE DOIT PAS MORDRE SUR LE PLAN DE NUMEROTATION LOCAL --
        # Les MSISDN du depot valent <noeud>00<op><ms> (100101, 200101...) :
        # six chiffres dont les deux premiers sont "<noeud>0". Un indicatif
        # "10" produirait donc "exten => _10." dans [gsm_in], qui avalerait
        # 100101 - l'abonne local deviendrait injoignable et l'appel partirait
        # sur le WAN. Un indicatif d'UN chiffre a le meme defaut, en pire.
        # Les defauts 11/22/.../99 passent tous cette regle ; c'est la regle
        # "^[1-9]0" ci-dessous qui ecarte 10, 20, ... 90.
        if ! [[ "${WAN_IND[$id]}" =~ ^[0-9]{2,4}$ ]]; then
            _wan_err "noeud $id : indicatif '${WAN_IND[$id]}' - 2 a 4 chiffres (1 chiffre entrerait en conflit
             ↪  avec les MSISDN locaux)"; rc=1
        elif [[ "${WAN_IND[$id]}" =~ ^[1-9]0 ]]; then
            _wan_err "noeud $id : indicatif '${WAN_IND[$id]}' commence comme un MSISDN local (<noeud>00<op><ms> →
             ↪  100101, 200101...) - il capterait les appels locaux ; prenez
             ↪  ${WAN_IND[$id]:0:1}${WAN_IND[$id]:0:1} par exemple"; rc=1
        fi
    done
    # Indicatifs : uniques ET aucun prefixe d'un autre. Un indicatif 3 qui
    # prefixe un indicatif 33 rend les patterns Asterisk (_3. vs _33.) et le
    # longest-prefix du relais SMS dependants de l'ordre - donc imprevisibles.
    for id in "${WAN_NODE_LIST[@]}"; do
        for other in "${WAN_NODE_LIST[@]}"; do
            [ "$id" = "$other" ] && continue
            if [ "${WAN_IND[$id]}" = "${WAN_IND[$other]}" ]; then
                _wan_err "noeuds $id et $other partagent l'indicatif ${WAN_IND[$id]}"; rc=1
            elif [ "${WAN_IND[$other]#${WAN_IND[$id]}}" != "${WAN_IND[$other]}" ]; then
                _wan_err "indicatif ${WAN_IND[$id]} (noeud $id) est un prefixe de ${WAN_IND[$other]} (noeud
                 ↪  $other)"; rc=1
            fi
            if [ "${WAN_IP[$id]}" = "${WAN_IP[$other]}" ]; then
                _wan_err "noeuds $id et $other partagent l'IP ${WAN_IP[$id]}"; rc=1
            fi
        done
    done
    if [ -n "${WAN_NODE_ID:-}" ] && [ "${WAN_NODE_ID:-0}" != "0" ]; then
        [ -n "${WAN_IP[$WAN_NODE_ID]:-}" ] \
            || { _wan_err "le noeud local ($WAN_NODE_ID) n'est pas dans la table"; rc=1; }
    fi
    return $rc
}

# -- Detection du noeud local par ses adresses ------------------------------
# Une SEULE ISO peut alors servir les N machines : chacune se reconnait a son
# IP. Sans ca il faudrait fabriquer une image par noeud.
wan_nodes_detect_self() {
    local id addrs
    addrs="$(ip -4 -o addr show scope global 2>/dev/null | awk '{print $4}' | cut -d/ -f1)"
    [ -n "$addrs" ] || return 1
    for id in "${WAN_NODE_LIST[@]}"; do
        if printf '%s\n' $addrs | grep -qx "${WAN_IP[$id]}"; then
            WAN_NODE_ID="$id"; return 0
        fi
    done
    return 1
}

# -- L'ADRESSE DE CE NOEUD SUR LE SEGMENT WAN -------------------------------
# "La premiere IP globale" est un mauvais critere sur une VM du banc : il y en
```

```
# a au moins trois, et deux sont des pieges.
#   • le NAT VirtualBox - toujours 10.0.2.x - sort vers Internet mais ne voit
#     aucun pair : un ASP lie dessus n'atteint jamais le hub ;
#   • les alias 172.20.0.1 / 172.20.0.11 / 172.20.1.10 que l'ISO pose elle-meme
#     sur le NIC (build-iso.sh, 20-dhcp.network) - ils appartiennent au plan
#     docker et n'existent sur aucun segment.
# La bonne adresse est celle de la carte "pont" VirtualBox. On la cherche dans
# cet ordre, du plus sur au plus deduit.
wan_detect_local_ip() {
    local addrs a

    addrs="$(ip -4 -o addr show scope global 2>/dev/null | awk '{print $4}' | cut -d/ -f1)"
    [ -n "$addrs" ] || return 1

    # 1. La table fait foi - a condition que l'adresse soit reellement montee.
    #    Une table qui annonce une adresse absente est une table perimee, pas
    #    une source d'autorite.
    if [ "${WAN_NODE_ID:-0}" != "0" ] && [ -n "${WAN_IP[${WAN_NODE_ID}]:-}" ]; then
        for a in $addrs; do
            [ "$a" = "${WAN_IP[$WAN_NODE_ID]}" ] && { printf '%s' "$a"; return 0; }
        done
    fi

    # 2. La plage host-only de VirtualBox (192.168.56.0/21), la seule qu'il
    #    autorise sans /etc/vbox/networks.conf. C'est le segment du banc.
    for a in $addrs; do
        case "$a" in
            192.168.5[6-9].*|192.168.6[0-3].*) printf '%s' "$a"; return 0 ;;
        esac
    done

    # 3. Toute autre adresse en 192.168.x : c'est la carte "pont" (acces par
    #    pont / bridged adapter), celle qui met la VM sur le meme reseau que
    #    l'hote. C'est l'adresse par laquelle les pairs la joignent.
    for a in $addrs; do
        case "$a" in 192.168.*) printf '%s' "$a"; return 0 ;; esac
    done

    # 4. A defaut, la premiere adresse qui ne soit ni le NAT VirtualBox, ni un
    #    alias du plan docker, ni la boucle locale.
    for a in $addrs; do
        case "$a" in
            10.0.2.*|172.20.*|127.*|169.254.*) ;;
            *) printf '%s' "$a"; return 0 ;;
        esac
    done

    return 1
}

wan_local_ip()  { printf '%s' "${WAN_IP[${WAN_NODE_ID}]:-}"; }
wan_local_ind() { printf '%s' "${WAN_IND[${WAN_NODE_ID}]:-}"; }

# ── Mise a jour de CE noeud dans la table, sans question ──────────────
# --node N dit quel noeud est cette machine. Son adresse dans la table peut
# etre perimee (une table copiee d'une autre machine) : on la remplace par
# l'adresse LOCALE reelle - celle qui route vers le hub si on le connait, sinon
# wan_detect_local_ip - puis on reecrit /etc/osmo-wan.conf. Les autres noeuds
# gardent leur entree : ce script ne sait pas ou ils sont.
#   $1 = numero du noeud (1-9)   $2 = hub (facultatif)
wan_self_refresh() {
    local id="$1" hub="${2:-}" ip="" ind

    [[ "$id" =~ ^[1-9]$ ]] || { _wan_err "wan_self_refresh : noeud '$id' invalide"; return 1; }
    WAN_NODE_ID="$id"
```

```
    if [ -n "$hub" ]; then
        ip="$(ip route get "$hub" 2>/dev/null | sed -n 's/.*src \([0-9.]*\).*/\1/p' | head -1)"
    fi
    [ -n "$ip" ] || ip="$(wan_detect_local_ip 2>/dev/null || true)"
    if [ -z "$ip" ]; then
        _wan_err "noeud $id : aucune adresse locale detectee - table non reecrite"
        return 1
    fi

    ind="${WAN_IND[$id]:-$(wan_default_ind "$id")}"
    WAN_IP[$id]="$ip"
    WAN_IND[$id]="$ind"
    WAN_NOPS[$id]="${WAN_NOPS[$id]:-1}"
    case " ${WAN_NODE_LIST[*]:-} " in
        *" $id "*) ;;
        *) WAN_NODE_LIST+=("$id") ;;
    esac
    WAN_NODE_COUNT="${#WAN_NODE_LIST[@]}"

    wan_nodes_save "${WAN_CONF_FILE:-/etc/osmo-wan.conf}" || return 1
    printf '  \033[0;36mWAN\033[0m        noeud %s = %s (table reecrite : %s)\n' \
        "$id" "$ip" "${WAN_CONF_FILE:-/etc/osmo-wan.conf}"
    return 0
}

# ── Saisie ──────────────────────────────────────────────────────────
# whiptail quand on en a un ET un terminal ; sinon `read`. Le mode "ni l'un ni
# l'autre" (cron, CI, docker exec -d) n'invente rien : il echoue en nommant
# les variables a poser.
_wan_ask() {   # titre  question  defaut
    local title="$1" q="$2" def="${3:-}" ans
    if [ -t 0 ] && [ -t 2 ] && command -v whiptail >/dev/null 2>&1 && [ -z "${WAN_NO_WHIPTAIL:-}" ]; then
        ans=$(whiptail --backtitle "osmo-operator - WAN" --title "$title" \
                --inputbox "$q" 10 72 "$def" 3>&1 1>&2 2>&3) || return 1
    elif [ -t 0 ]; then
        printf '  %s [%s] : ' "$q" "$def" >&2
        read -r ans || return 1
    else
        _wan_err "pas de terminal : renseignez WAN_NODES / WAN_NODE_ID / WAN_OPS"
        return 1
    fi
    printf '%s' "${ans:-$def}"
}

# wan_nodes_prompt [nb_ops_local]
#   1. combien de noeuds
#   2. pour CHAQUE noeud : son IP publique, puis son indicatif
#   3. quel numero porte le noeud construit par ce lancement
wan_nodes_prompt() {
    local n i ip ind me auto_ip def_n
    def_n="${WAN_NODE_COUNT:-0}"; [ "$def_n" -lt 1 ] && def_n=3

    n=$(_wan_ask "WAN - noeuds" "Nombre de noeuds du WAN (1-9) :" "$def_n") || return 1
    [[ "$n" =~ ^[1-9]$ ]] \
        || { _wan_err "nombre de noeuds invalide : '$n' (1 a 9)"; return 1; }

    auto_ip=$(ip route get 1.1.1.1 2>/dev/null | awk '{for(i=1;i<=NF;i++) if($i=="src") print $(i+1)}' | head -1)
    [ -z "$auto_ip" ] && auto_ip=$(hostname -I 2>/dev/null | awk '{print $1}')

    # ${x[@]+"${x[@]}"} : sous `set -u`, un tableau VIDE developpe sans cette
    # garde est une erreur fatale sur bash < 4.4 - et la premiere saisie part
    # justement d'une table vide.
    local -a old_list=(${WAN_NODE_LIST[@]+"${WAN_NODE_LIST[@]}"})
    local -A old_ip old_ind
    for i in "${old_list[@]}"; do old_ip[$i]="${WAN_IP[$i]}"; old_ind[$i]="${WAN_IND[$i]}"; done
    WAN_IP=(); WAN_IND=(); WAN_NODE_LIST=()
```

```
    for i in $(seq 1 "$n"); do
        ip=$(_wan_ask "WAN - noeud $i/$n" "IP publique du noeud $i :" "${old_ip[$i]:-}") || return 1
        ind=$(_wan_ask "WAN - noeud $i/$n" "Indicatif du noeud $i (prefixe d'appel) :" \
                       "${old_ind[$i]:-$(wan_default_ind "$i")}") || return 1
        WAN_IP[$i]="$ip"; WAN_IND[$i]="$ind"; WAN_NODE_LIST+=("$i")
    done
    WAN_NODE_COUNT="$n"

    # Numero du noeud LOCAL. On propose celui dont l'IP est la notre.
    me="${WAN_NODE_ID:-0}"
    if [ "$me" = "0" ] && [ -n "$auto_ip" ]; then
        for i in "${WAN_NODE_LIST[@]}"; do [ "${WAN_IP[$i]}" = "$auto_ip" ] && me="$i"; done
    fi
    [ "$me" = "0" ] && me=1
    me=$(_wan_ask "WAN - ce noeud" "Numero du noeud construit par ce lancement (1-$n) :" "$me") || return 1
    [[ "$me" =~ ^[0-9]+$ ]] && [ -n "${WAN_IP[$me]:-}" ] \
        || { _wan_err "noeud local '$me' hors table"; return 1; }
    WAN_NODE_ID="$me"

    wan_nodes_validate || return 1
    return 0
}

# ── Persistance ──────────────────────────────────────────────────────
wan_nodes_save() {
    local f="${1:-$WAN_CONF_FILE}" host="${WAN_DOCKER_HOST:-}"
    mkdir -p "$(dirname "$f")"
    # L'hote docker n'est connu que de qui vient de le decouvrir. Comme ce
    # fichier est reecrit ENTIER a chaque sauvegarde, une sauvegarde faite par un
    # script qui ne l'a pas dans son environnement effacait l'adresse : la VM
    # perdait, sans rien dire, la seule indication de par ou joindre le
    # backbone. On relit donc celle qui est deja dans le fichier avant d'ecrire.
    if [ -z "$host" ] && [ -r "$f" ]; then
        host="$(sed -n 's/^WAN_DOCKER_HOST=//p' "$f" | tr -d "\"' " | tail -1)"
    fi
    # [2026-10-18] WAN_AUTO_SAVE, s'il est pose, l'emporte sur WAN_AUTO : relire
    # la table (wan_nodes_load la source) remettait WAN_AUTO=1 dans le shell, et
    # chaque sauvegarde suivante le reecrivait - un seul --wan, et le WAN
    # remontait a chaque demarrage. start-native.sh le pose : 1 seulement si
    # c'est WAN_AUTO=1 lui-meme qui a monte le WAN (noeud d'ISO construite avec
    # --wan), 0 pour un --wan tape a la main.
    cat > "$f" <<EOF
# /etc/osmo-wan.conf - table des noeuds WAN (genere par network/wan-nodes.sh)
# Format : WAN_NODES="id:ip:indicatif[:operateurs] ..."  (operateurs : defaut 1)
WAN_NODES="$(wan_nodes_spec)"
WAN_NODE_COUNT=${WAN_NODE_COUNT}
WAN_NODE_ID=${WAN_NODE_ID}
WAN_OPS=${WAN_OPS}
# WAN_AUTO=1 : les lanceurs appliquent le WAN sans qu'on repasse --wan.
WAN_AUTO=${WAN_AUTO_SAVE:-${WAN_AUTO:-0}}
EOF
    # Ecrite seulement si on la connait : une ligne vide serait lue comme une
    # passerelle, et une passerelle vide ne se distingue pas d'une passerelle
    # fausse au moment de poser la route.
    if [ -n "$host" ]; then
        {
            printf '# Hote qui porte docker, par ou une VM route vers le backbone des\n'
            printf '# conteneurs. Adresse DHCP : elle se redecouvre, elle ne se fige pas.\n'
            printf 'WAN_DOCKER_HOST="%s"\n' "$host"
        } >> "$f"
    fi
    return 0
}

wan_nodes_load() {
```

```
    local f="${1:-$WAN_CONF_FILE}"
    [ -r "$f" ] || return 1
    # Le sourcing ramene WAN_NODES, le noeud local, et WAN_DOCKER_HOST quand le
    # fichier la porte - c'est par elle qu'une VM sait vers qui router 172.20.
    # shellcheck disable=SC1090
    . "$f"
    wan_nodes_parse "${WAN_NODES:-}" || return 1
    return 0
}

wan_nodes_summary() {
    local id tag
    printf '  Noeuds WAN : %s   (ce noeud : %s)\n' "$WAN_NODE_COUNT" "$(_wanc "${WAN_NODE_ID}")"
    for id in "${WAN_NODE_LIST[@]}"; do
        tag="   "; [ "$id" = "${WAN_NODE_ID}" ] && tag="◄ ce noeud"
        printf '    node %-2s  %-15s  indicatif %-4s %s\n' "$id" "${WAN_IP[$id]}" "${WAN_IND[$id]}" "$tag"
    done
    printf '    Composer %s + numero pour joindre le noeud correspondant.\n' \
        "$(_wanc "<indicatif>")"
}
```

3.260 /opt/GSM/osmo-operator/opt-gsm/grgsm_fft_live.py

2768 octets, 62 lignes → 62 lignes

```
#!/usr/bin/env python3
# grgsm_fft_live.py — FFT EN DIRECT (spectre ASCII). Lit le FIFO live
# (defaut /tmp/iq_asciifft.fifo, fc32 4 SPS) que qemu pousse trame-par-trame,
# OU un cfile fichier (seek-tail) en fallback. Marque le pic FCCH.
#
# FIFO : ouvert O_RDWR|O_NONBLOCK -> open non-bloquant + pas d'EOF (on garde un
# write-end) => le write non-bloquant de qemu trouve toujours un lecteur. A
# chaque tour on DRAINE le backlog et on ne garde que les N derniers samples
# (le plus frais), donc pas de lag qui s'accumule.
import numpy as np, time, os, stat
CFILE=os.environ.get("CFILE","/tmp/iq_asciifft.fifo")
FS=float(os.environ.get("FS","1083333"))    # 4 SPS relay (FIFO). cfile BSP 1 SPS=270833
N=int(os.environ.get("NFFT","4096"))
COLS=70
FCCH=1625000.0/24                           # +67708 Hz

def _is_fifo(p):
    try: return stat.S_ISFIFO(os.stat(p).st_mode)
    except OSError: return False

fifo_fd=None
if _is_fifo(CFILE) or CFILE.endswith(".fifo"):
    if not os.path.exists(CFILE): os.mkfifo(CFILE,0o666)
    fifo_fd=os.open(CFILE, os.O_RDWR|os.O_NONBLOCK)

def _read_iq():
    if fifo_fd is not None:
        buf=b""
        while True:
            try: c=os.read(fifo_fd, 1<<16)
            except BlockingIOError: break
            if not c: break
            buf+=c
            if len(buf) > N*8*4: buf=buf[-(N*8):]     # garde le frais, borne la RAM
        buf=buf[-(N*8):]
        if len(buf) < N*8: return np.array([],dtype=np.complex64)
        return np.frombuffer(buf[:(len(buf)//8)*8], dtype=np.complex64)
    try: sz=os.path.getsize(CFILE)
    except OSError: sz=0
```

```
    if sz < N*8: return np.array([],dtype=np.complex64)
    with open(CFILE,"rb") as f:
        f.seek(max(0,(sz//8-N)*8)); return np.frombuffer(f.read(N*8),dtype=np.complex64)

while True:
    iq=_read_iq()
    os.system("clear")
    print("== FFT LIVE %s (%s, fs=%.0f, N=%d) ==  FCCH attendu %+.0f Hz"%(
        CFILE, "FIFO" if fifo_fd is not None else "file", FS, N, FCCH))
    if len(iq) < N:
        print("\n  pas (encore) de donnees -> qemu ecrit-il le FIFO ? (relance ./run.sh)")
        time.sleep(1); continue
    sp=np.abs(np.fft.fftshift(np.fft.fft(iq*np.hanning(len(iq)))))
    fr=np.fft.fftshift(np.fft.fftfreq(len(iq),1/FS))
    pk=fr[np.argmax(sp)]; mag=np.abs(iq)
    print("  ampl moy=%.5f max=%.4f | PIC %+.0f Hz | dphi std=%.2f (GMSK~1.57)\n"%(
        mag.mean(),mag.max(),pk,np.std(np.angle(iq[1:]*np.conj(iq[:-1])))))
    B=24; step=len(sp)//B; spb=np.array([sp[i*step:(i+1)*step].max() for i in range(B)])
    frb=np.array([fr[i*step] for i in range(B)]); m=spb.max()+1e-9
    for i in range(B):
        bar=int(COLS*spb[i]/m); tag=" <-FCCH" if abs(frb[i]-FCCH)<FS/B else ""
        print("%+7.0f|%s%s"%(frb[i],"#"*bar,tag))
    time.sleep(0.5)
```

3.261 /opt/GSM/osmo-operator/opt-gsm/record_drain.py

5655 octets, 119 lignes → 119 lignes

```
#!/usr/bin/env python3
# record_drain.py — draine le FIFO live iq_record.fifo -> record.cfile en RING
# (128 MB, fseek wrap), HORS du hot-path qemu. C'est l'EXTERNALISATION du ring
# cfile que qemu ecrivait avant dans son hot-path DL (= ce qui causait les
# underruns). Ici qemu ne fait qu'un write() non-bloquant vers le FIFO ; ce
# process (independant) absorbe le flux et l'ecrit sur disque a son rythme.
# grgsm_decode -c relit record.cfile offline comme avant => SI preserve.
#
# Ouverture du FIFO en O_RDWR : ne bloque pas a l'ouverture ET ne voit jamais
# d'EOF (on garde nous-memes un write-end), donc qemu (O_WRONLY|O_NONBLOCK)
# trouve toujours un lecteur => son open reussit et le flux coule.
#
# ============================================================================
# [2026-08-11] CAPTURE COMPLETE, DECLENCHABLE — a cote du ring, pas a sa place
# ============================================================================
# Le ring ci-dessus rembobine tous les 128 Mo (~15,5 s) : parfait pour rejouer
# le passe immediat, inutilisable pour capturer un appel entier. La capture
# complete ci-dessous ne rembobine JAMAIS.
#
# POURQUOI UN TUBE DEDIE, et pas une lecture de iq_record.fifo : un FIFO n'est
# pas diffusable — deux lecteurs se PARTAGENT les octets, ils ne les voient pas
# chacun en entier. Draîner iq_record.fifo ici volerait donc une partie du flux
# au ring et casserait le decodage SI. On ajoute un 8e tube au relai
# (CALYPSO_RELAY_FIFOS), que qemu alimente comme les autres.
# ⚠ RELAY_NFIFO_MAX vaut 8 dans qemu_wrap.c et 7 tubes etaient deja declares :
# ce tube consomme LA DERNIERE PLACE. En ajouter un 9e exige de relever la
# constante ET de recompiler, sinon il est ignore EN SILENCE.
#
# ⚠ PAS SUR TMPFS. /dev/shm et /tmp sont de la RAM. Le flux fait 1083333 ech/s
# x 8 o = 8,7 Mo/s = 520 Mo/min = 31 Go/h : une capture complete sur tmpfs
# epuise la memoire de la machine en quelques minutes. Le defaut pointe donc sur
# l'overlay (disque). Verifier `df -h` avant une longue capture.
#
# DECLENCHEMENT : la capture n'ecrit que TANT QUE le fichier declencheur existe.
#     touch /tmp/iq_full.trigger    -> demarre (ouvre/tronque le .cfile)
#     rm    /tmp/iq_full.trigger    -> arrete et FERME proprement le fichier
```

```
# Hors declenchement le tube est quand meme LU et jete : sinon l'anneau memoire
# du relai se remplit et qemu se met a jeter des trames entieres pour TOUS les
# tubes.
#
# PLAFOND : a CALYPSO_RECORD_FULL_MAX octets la capture S'ARRETE et le DIT. Elle
# ne rembobine pas — un fichier qui se remord la queue en silence est exactement
# ce qu'on veut eviter ici.
import os, sys, threading

FIFO = os.environ.get("CALYPSO_RECORD_FIFO", "/tmp/iq_record.fifo")
OUT  = os.environ.get("CALYPSO_RECORD_FILE", "/tmp/record.cfile")
RING = int(os.environ.get("CALYPSO_RECORD_RING", str(128 << 20)))   # 128 MB

FULL_ON      = os.environ.get("CALYPSO_RECORD_FULL", "0") == "1"
FULL_FIFO    = os.environ.get("CALYPSO_RECORD_FULL_FIFO", "/tmp/iq_full.fifo")
FULL_FILE    = os.environ.get("CALYPSO_RECORD_FULL_FILE", "/opt/GSM/captures/full.cfile")
FULL_TRIGGER = os.environ.get("CALYPSO_RECORD_FULL_TRIGGER", "/tmp/iq_full.trigger")
FULL_MAX     = int(os.environ.get("CALYPSO_RECORD_FULL_MAX", str(8 << 30)))   # 8 GB

def log(msg):
    sys.stderr.write(msg + "\n")
    sys.stderr.flush()

def full_drain():
    """Capture complete : lit toujours, n'ecrit que sous declencheur, ne wrap jamais."""
    if not os.path.exists(FULL_FIFO):
        os.mkfifo(FULL_FIFO, 0o666)
    os.makedirs(os.path.dirname(FULL_FILE) or ".", exist_ok=True)
    fd = os.open(FULL_FIFO, os.O_RDWR)       # meme idiome que le ring : ni blocage ni EOF
    log("[record-full] %s -> %s (plafond %d MB) — declencheur : %s"
        % (FULL_FIFO, FULL_FILE, FULL_MAX >> 20, FULL_TRIGGER))
    out = None
    w = 0
    plafond_dit = False
    while True:
        b = os.read(fd, 1 << 16)
        if not b:
            continue
        armed = os.path.exists(FULL_TRIGGER)
        if armed and out is None:
            out = open(FULL_FILE, "wb", buffering=1 << 20)
            w = 0
            plafond_dit = False
            log("[record-full] DEMARRE -> %s" % FULL_FILE)
        elif not armed and out is not None:
            out.flush(); out.close(); out = None
            log("[record-full] ARRETE : %s, %.1f MB (%.1f s d'IQ)"
                % (FULL_FILE, w / 1048576.0, w / 8666664.0))
        if out is None:
            continue                          # on jette, mais on a LU : le relai ne bouche pas
        if w + len(b) > FULL_MAX:
            if not plafond_dit:
                log("[record-full] PLAFOND %d MB atteint : capture ARRETEE (pas de "
                    "rembobinage). Retirer %s, vider %s, puis relancer."
                    % (FULL_MAX >> 20, FULL_TRIGGER, FULL_FILE))
                plafond_dit = True
            out.flush(); out.close(); out = None
            continue
        out.write(b)
        w += len(b)
        out.flush()

if not os.path.exists(FIFO):
    os.mkfifo(FIFO, 0o666)

if FULL_ON:
```

```
    threading.Thread(target=full_drain, daemon=True).start()
else:
    log("[record-full] inactif (CALYPSO_RECORD_FULL != 1)")

fd = os.open(FIFO, os.O_RDWR)            # O_RDWR : pas de blocage / pas d'EOF
out = open(OUT, "wb", buffering=1 << 20)
w = 0
log("[record-drain] %s -> %s (ring %d MB)" % (FIFO, OUT, RING >> 20))
while True:
    b = os.read(fd, 1 << 16)             # 64 KB
    if not b:
        continue
    out.write(b)
    w += len(b)
    if w >= RING:                        # RING : rembobine (grgsm relit depuis 0)
        out.seek(0); w = 0
    out.flush()
```

3.262 /opt/GSM/osmo-operator/opt-gsm/si_bridge.py

19242 octets, 389 lignes → 389 lignes

```
#!/usr/bin/env python3
# Bridge : grgsm_decode -v (DECODE le vrai SI du flux continu) -> parse les
# lignes SI (PD=0x06) -> GSMTAP 16B + L2 -> 4730 (shunt feed_si -> DATA_IND ->
# mobile). Le SCH reel (patch grgsm) -> UDP 4731 (feed_sb).
#
# GESTION DU FN (corrige le bug "GSMTAP frame_number=0") :
#   - le FN reel vient des lignes SCHBSIC <bsic> <fn> <toa> (horloge fiable),
#     ou d'un "FN: <n>" explicite si grgsm -v en imprime un.
#   - on l'ecrit dans le champ frame_number du GSMTAP envoye a feed_si.
#   - rappel 51-multiframe BCCH non-combine TS0 DL (FN % 51) :
#       0,10,20,30,40 = FCCH | 1,11,21,31,41 = SCH | 2-5 = BCCH
#       6-9,12-19,22-29,32-39,42-49 = CCCH (PCH/AGCH) | 50 = idle
#     -> 31 = position SCH, 32 = debut d'un bloc CCCH, 51 = longueur multiframe.
import subprocess, socket, struct, re, sys, os, threading, time

ARFCN = int(os.environ.get("CALYPSO_CCCH_ARFCN", "514"))
CF = sys.argv[1] if len(sys.argv) > 1 else "/tmp/iq_grgsm.fifo"
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

SI = {0x1b:"SI3",0x1a:"SI2",0x1c:"SI4",0x19:"SI1",0x1d:"SI2bis",0x1e:"SI2ter",0x06:"RR"}  # NB: 0x21 = PAGING
 ↪  (pas SI13=0x00)
# CCCH/AGCH downlink (PORTE 3a) : l'IMM ASSIGN doit etre forwarde au mobile, pas
# seulement les SI. gr-gsm le decode (06 3f) mais si_bridge ne forwardait que les SI.
CCCH = {0x3f:"IMM-ASSIGN", 0x39:"IMM-ASSIGN-EXT", 0x3a:"IMM-ASSIGN-REJ"}
# Paging Request (PCH) : type1=0x21 (== GSM48_MT_RR_PAG_REQ_1, PAS SI13), type2=0x22, type3=0x24.
PAGING = {0x21:"PAG-REQ-1", 0x22:"PAG-REQ-2", 0x24:"PAG-REQ-3"}
GSMTAP_BCCH = 0x01
GSMTAP_AGCH = 0x04
GSMTAP_SDCCH4 = 0x07
GSMTAP_SACCH  = 0x07 | 0x80   # SDCCH/4 + ACCH = SACCH dediee SS0 (SI5/SI6 reels)

def fn51_role(fn):
    m = fn % 51
    if m in (0,10,20,30,40): return "FCCH"
    if m in (1,11,21,31,41): return "SCH"
    if m in (2,3,4,5):       return "BCCH"
    if m == 50:              return "idle"
    return "CCCH"

def gsmtap(fn, chan, l2):
    # v2, hdr_len=4 (16B), type=UM(1), ts=0, arfcn, signal_dbm=0 (RIEN d'injecte),
    # snr=0, frame_number=fn (REEL, decode), sub_type=chan, antenna=0, sub_slot=0, res=0
```

```
    hdr = struct.pack(">BBBBHbbIBBBB", 2, 4, 0x01, 0, ARFCN & 0xffff,
                      0, 0, fn & 0xffffffff, chan, 0, 0, 0)
    return hdr + l2

n = 0; nsch = 0; nsd = 0; nsa = 0; last_fn = 0; last_sd_fn = -1; last_sa_fn = -1

def handle_line(line):
    global n, nsch, nsd, nsa, last_fn, last_sd_fn, last_sa_fn
    # --- SCH reel : "SCHBSIC <bsic> <fn> <toa>" -> 4731 + horloge FN ---
    msch = re.search(r"SCHBSIC\s+(\d+)\s+(\d+)\s+(-?\d+)", line)
    if msch:
        bsic, fn, toa = int(msch.group(1)), int(msch.group(2)), int(msch.group(3))
        last_fn = fn
        s.sendto(b"SCH1" + struct.pack("<iii", bsic, fn, toa), ("127.0.0.1", 4731))
        nsch += 1
        if nsch <= 10 or nsch % 100 == 0:
            print("[si-bridge] SCH -> 4731 : BSIC=%d (ncc=%d bcc=%d) FN=%d (%%51=%d %s) TOA=%d  #%d"
                  % (bsic,(bsic>>3)&7,bsic&7,fn,fn%51,fn51_role(fn),toa,nsch), flush=True)
        return
    # --- FN explicite eventuel dans la ligne -v ---
    mfn = re.search(r"\bFN[:=]?\s*(\d+)", line)
    if mfn: last_fn = int(mfn.group(1))
    # --- ligne SI hexa ---
    lstr = line.strip()
    m = re.search(r":\s+([0-9a-fA-F][0-9a-fA-F ]+)$", lstr)
    if not m: return
    try: by = bytes(int(x,16) for x in m.group(1).split())
    except: return
    if len(by) >= 3 and by[1] == 0x06 and (by[2] in SI or by[2] in CCCH or by[2] in PAGING):
        mline_fn = re.match(r"^\s*(\d+)\b", lstr)
        si_fn = int(mline_fn.group(1)) if mline_fn else last_fn
        L2 = (by[:23] + b"\x2b"*23)[:23]
        # IMM ASSIGN (AGCH) ET Paging Request (PCH) -> meme chemin AGCH/PCH cote shunt
        # (feed_agch -> bloc CCCH -> firmware chan_nr=0x90 -> gsm48_rr_rx_pch_agch qui
        # dispatch IMM-ASS vs PAGING par msg type). SI -> BCCH.
        is_ccch = (by[2] in CCCH) or (by[2] in PAGING)
        chan = GSMTAP_AGCH if is_ccch else GSMTAP_BCCH
        s.sendto(gsmtap(si_fn, chan, L2), ("127.0.0.1", 4730))
        n += 1
        name = CCCH.get(by[2]) or PAGING.get(by[2]) or SI.get(by[2])
        if is_ccch or n <= 20 or n % 50 == 0:
            print("[si-bridge] %s (mt=0x%02x) FN=%d (%%51=%d %s) -> %s (4730)  #%d"
                  % (name, by[2], si_fn, si_fn%51, fn51_role(si_fn),
                     "feed_agch/PCH" if is_ccch else "feed_si", n), flush=True)
    # --- SDCCH/4 SS0 DL (#2) : LAPDm (UA/AUTH), pas du RR -> gate fn%51 in {22-25} ---
    mfn_sd = re.match(r"^\s*(\d+)\b", lstr)
    sd_fn = int(mfn_sd.group(1)) if mfn_sd else last_fn
    if len(by) >= 3 and (sd_fn % 51) in (22, 23, 24, 25):
        L2 = (by[:23] + b"\x2b"*23)[:23]
        # DEDUP PAR FN (corrige) : gr-gsm peut re-decoder le MEME bloc on-air (meme FN)
        # -> il faut sauter ce doublon (re-decode). MAIS la BTS RE-EMET legitimement le
        # MEME contenu (UA byte-identique) sur un FN DIFFERENT en reponse a une
        # retransmission SABM T200 du mobile : ce re-UA DOIT etre forwarde sinon le
        # mobile ne confirme jamais (UA rate -> boucle SABM -> "SABM not expected in
        # timer recovery"). L'ancien dedup-par-contenu (L2 != last_sd_l2) DROPPAIT ce
        # re-UA legitime. On dedup donc par FN decode (sd_fn) : grgsm emet chaque FN
        # on-air UNE seule fois (FN monotone, verifie) -> dedup-par-FN supprime le
        # re-decode (meme FN) ET forwarde le re-envoi BTS (FN distinct, meme contenu).
        # On saute toujours le bourrage LAPDm (UI c=0x03).
        ctrl = by[1] if len(by) > 1 else 0
        is_fill = (ctrl == 0x03)
        if (not is_fill) and sd_fn != last_sd_fn:
            last_sd_fn = sd_fn
            s.sendto(gsmtap(sd_fn, GSMTAP_SDCCH4, L2), ("127.0.0.1", 4730))
            nsd += 1
            if nsd <= 20 or nsd % 50 == 0:
```

```
                print("[si-bridge] SDCCH/4 SS0 DL (a0=0x%02x c=0x%02x) FN=%d (%%51=%d) -> feed_sdcch (4730)  #%d
                ↪  [new]"
                      % (by[0], ctrl, sd_fn, sd_fn%51, nsd), flush=True)
            # ARMEMENT DU TCH : l'ASSIGNMENT COMMAND passe par ce meme bloc DL.
            check_assignment(by, sd_fn)
    # --- SACCH (SDCCH/4 SS0) DL : SI5(0x1d)/SI6(0x1e) REELS -> feed_sacch ---
    # Slots SACCH SS0 (combine CCCH+SDCCH/4) : fn%51 in {42-45}. grgsm donne le
    # bloc 23o = [L1 hdr 2][LAPDm: 03 03 len 06 mt L3...]. On detecte le RR header
    # "06 1d"/"06 1e" et on forwarde le bloc TEL QUEL au shunt (sub_type 0x87 =
    # SDCCH4|ACCH) -> feed_sacch REEL, remplace la fabrication SI3->SI6.
    if len(by) >= 8 and (sd_fn % 51) in (42, 43, 44, 45):
        mt_sa = None
        for off in range(2, min(9, len(by) - 1)):
            if by[off] == 0x06 and by[off + 1] in (0x1d, 0x1e):
                mt_sa = by[off + 1]; break
        if mt_sa is not None and sd_fn != last_sa_fn:
            last_sa_fn = sd_fn
            L2 = (by[:23] + b"\x2b" * 23)[:23]
            s.sendto(gsmtap(sd_fn, GSMTAP_SACCH, L2), ("127.0.0.1", 4730))
            nsa += 1
            if nsa <= 20 or nsa % 50 == 0:
                print("[si-bridge] SACCH SI%d (mt=0x%02x) FN=%d (%%51=%d) -> feed_sacch REEL (4730)  #%d"
                      % (5 if mt_sa == 0x1d else 6, mt_sa, sd_fn, sd_fn % 51, nsa), flush=True)

# === STAGE 3 : DECHIFFREMENT DL — DEUX grgsm QUI SE DELEGUENT ================
# PROBLEME resolu : tuer grgsm pour le relancer avec -k au CIPHER MODE retirait
# le lecteur de la FIFO -> l'ipc-device se figeait (DL FIFO plein -> osmo-trx
# SETPOWER no-response -> feed_iq gele -> pas de LU accept).
# SOLUTION (design B) : on ne tue JAMAIS le grgsm CLAIR. Deux process tournent
# en parallele, chacun sur SA FIFO (les deux alimentees par l'ipc-device relay) :
#   - grgsm CLAIR  sur CF (iq_grgsm.fifo)  : decode etablissement/ID/AUTH(RAND)/
#     CIPHER MODE COMMAND. Ne sort QUE des trames CRC-valides -> se tait tout seul
#     sur le dedie une fois le DL chiffre.
#   - grgsm CHIFFRE sur CIPH_CF (-e -k)    : spawne des qu'un Kc apparait (fenetre
#     RAND->cipher : il a le temps de locker FCCH/SCH AVANT que le DL passe chiffre),
#     decode le dedie chiffre (LU ACCEPT...). Tue seulement au release (Kc efface).
# Les deux sorties sont fusionnees par handle_line (sous lock) : le CRC de grgsm
# fait le tri, le dedup par FN evite les doublons. ZERO trou de lecteur sur la
# FIFO clair -> l'ipc-device ne gele plus. Le churn (spawn/kill) du grgsm chiffre
# est inoffensif grace au writer ipc-device non-bloquant (relay_fifo_writer).
CIPH_CF = os.environ.get("CALYPSO_CIPH_FIFO", "/tmp/iq_grgsm_ciph.fifo")
_hl_lock = threading.Lock()

# grgsm_decode HARDCODE le port GSMTAP UDP 4729 (socket_pdu UDP_SERVER+CLIENT, pas
# d'option). Deux instances -> "bind: Address already in use" -> le 2e crash
# (silencieux car stderr=DEVNULL) -> jamais de decode chiffre. On lit le STDOUT
# (-v), pas le 4729 -> on patche une COPIE pour le grgsm chiffre sur un port libre
# (4799). Genere au demarrage -> reproductible, survit au reclone du conteneur.
import shutil
def make_ciph_binary():
    src = shutil.which("grgsm_decode")
    if not src:
        return "grgsm_decode"
    dst = "/tmp/grgsm_decode_ciph"
    try:
        with open(src) as f:
            code = f.read()
        code = code.replace('"4729"', '"4799"')   # port GSMTAP unique (pas de conflit avec le clair)
        with open(dst, "w") as f:
            f.write(code)
        os.chmod(dst, 0o755)
        print("[si-bridge] grgsm chiffre patche -> %s (GSMTAP 4729->4799)" % dst, flush=True)
        return dst
    except Exception as e:
        print("[si-bridge] patch grgsm chiffre echoue (%s) -> fallback partage 4729" % e, flush=True)
        return src
```

```
GRGSM_CIPH = make_ciph_binary()

def spawn_grgsm(fifo, algo=0, kc_hex=None, binary="grgsm_decode", errlog=None):
    cmd = [binary, "-m", "BCCH_SDCCH4", "-t", "0", "-a", str(ARFCN),
           "-c", fifo, "-s", "1083333", "-v"]
    if kc_hex and algo in (1, 2, 3):
        cmd += ["-e", str(algo), "-k", kc_hex]
    err = open(errlog, "w") if errlog else subprocess.DEVNULL
    return subprocess.Popen(cmd, stdout=subprocess.PIPE, stderr=err, text=True)

# ============================================================================
# ARMEMENT DU TCH : ASSIGNMENT COMMAND -> calypso_tch_cfg -> 3e grgsm_decode
# ============================================================================
# [2026-08-11] RECONSTRUIT. Cette logique existait (08/08) et a disparu de
# l'arbre le 11/08. Sans elle la panne est en cascade et SILENCIEUSE :
#   calypso_tch_cfg reste a 0 octet -> aucun `grgsm_decode -m TCHF` -> aucun
#   GSMTAP dedie -> feed_tch_sacch JAMAIS appele -> la SACCH du TCH n'est pas
#   alimentee -> le MS y lit le SI5/SI6 du camp (a_cd est PARTAGE camp<->TCH,
#   avec DEUX conventions d'en-tete) -> "Short header message type 0x07
#   unsupported" une fois par seconde, a partir de ~30 s apres l'ASSIGNMENT.
#
# CONTRAT DU FICHIER — lu chez son CONSOMMATEUR (qemu_wrap.c
# calypso_tch_cfg_read), jamais devine :
#   0..3 seq uint32 LE (≠0)   4 TN   5 TSC   6..7 ARFCN uint16 LE   (16 o)
# Le seq est ecrit EN DERNIER : publication atomique, le lecteur ne voit jamais
# un demi-enregistrement.
#
# JUGES : calypso_tch_cfg non vide ; un process `-m TCHF` vivant ;
#         `feed_tch_sacch` > 0 dans qemu.log ; `Short header 0x07` = 0.
TCH_CFG_PATH = "/dev/shm/calypso_tch_cfg"
TCH_FIFO     = os.environ.get("CALYPSO_TCH_FIFO", "/tmp/iq_grgsm_tch.fifo")
TCH_SPEECH   = "/dev/shm/calypso_tch_speech.gsm"
_tch_proc = None
_tch_cur  = None          # (tn, tsc, arfcn) actuellement arme
_tch_seq  = 0

def _tch_write_cfg(tn, tsc, arfcn):
    """Publie TN/TSC/ARFCN. seq en dernier = publication atomique."""
    global _tch_seq
    _tch_seq += 1
    buf = bytearray(16)
    buf[4] = tn & 0xff
    buf[5] = tsc & 0xff
    buf[6:8] = int(arfcn).to_bytes(2, "little")
    buf[0:4] = _tch_seq.to_bytes(4, "little")
    fd = os.open(TCH_CFG_PATH, os.O_CREAT | os.O_WRONLY, 0o644)
    try:
        os.pwrite(fd, bytes(buf), 0)
    finally:
        os.close(fd)

def make_tch_binary():
    """Copie patchee de grgsm_decode pour le TCH — DEUX ports a corriger.

    grgsm_decode code en dur 127.0.0.1:4729 DEUX fois (l.122-123) :
      - l.122 UDP_SERVER : ne sert QU'A eviter les ICMP port-unreachable. Deux
        instances -> « bind: Address already in use » -> la 2e MEURT. C'est ce
        qui faisait mourir le decodeur TCH aussitot lance (14 reamorcages
        mesures le 11/08), donc feed_tch_sacch = 0, donc SACCH dediee muette.
      - l.123 UDP_CLIENT : c'est la VRAIE sortie GSMTAP. En mode TCHF,
        tch_f_decoder ET cch_decoder y publient (l.213-214).
    On bind donc le serveur sur un port libre, et on pointe le client droit sur
    4730, que le shunt ecoute : il dispatche par sub_type — TCH_F (0x02) ->
    feed_facch, TCH_ACCH (0x82) -> feed_tch_sacch (calypso_dsp_shunt.c:3126-3134).
    Pas de parsing de stdout : les donnees vont du decodeur au shunt en direct.
    ⚠ Remplacements CIBLES ligne par ligne : un `replace("4729","4730")` global
```

```
    ferait binder le serveur sur 4730 et entrerait en conflit avec le shunt.
    """
    src = shutil.which("grgsm_decode")
    if not src:
        return "grgsm_decode"
    dst = "/tmp/grgsm_decode_tch"
    try:
        with open(src) as f:
            code = f.read()
        a = '"UDP_SERVER", "127.0.0.1", "4729"'
        b = '"UDP_CLIENT", "127.0.0.1", "4729"'
        if a not in code or b not in code:
            print("[si-bridge] grgsm TCH : motifs de port introuvables -> "
                  "fallback non patche (conflit 4729 probable)", flush=True)
            return src
        code = code.replace(a, '"UDP_SERVER", "127.0.0.1", "4739"')
        code = code.replace(b, '"UDP_CLIENT", "127.0.0.1", "4730"')
        with open(dst, "w") as f:
            f.write(code)
        os.chmod(dst, 0o755)
        print("[si-bridge] grgsm TCH patche -> %s (bind 4739, GSMTAP -> 4730)" % dst,
              flush=True)
        return dst
    except Exception as e:
        print("[si-bridge] patch grgsm TCH echoue (%s)" % e, flush=True)
        return src
GRGSM_TCH = make_tch_binary()

def _tch_spawn(tn, arfcn):
    """3e decodeur, sur le tube IQ dedie (alimente par CALYPSO_RELAY_FIFOS)."""
    cmd = [GRGSM_TCH, "-m", "TCHF", "-t", str(tn), "-a", str(arfcn),
           "-c", TCH_FIFO, "-s", "1083333", "-v", "-d", "FR", "-o", TCH_SPEECH]
    try:
        err = open("/tmp/osmo-nitb/logs/grgsm_decode_tch.log", "w")
    except Exception:
        err = subprocess.DEVNULL
    return subprocess.Popen(cmd, stdout=subprocess.DEVNULL, stderr=err, text=True)

def arm_tch(tn, tsc, arfcn):
    global _tch_proc, _tch_cur
    cfg = (tn, tsc, arfcn)
    vivant = _tch_proc is not None and _tch_proc.poll() is None
    if cfg == _tch_cur and vivant:
        return                                  # deja arme sur cette config
    if _tch_proc is not None and vivant:
        try: _tch_proc.terminate()
        except Exception: pass
    _tch_write_cfg(tn, tsc, arfcn)
    _tch_proc = _tch_spawn(tn, arfcn)
    _tch_cur = cfg
    print("[si-bridge] TCH arme : TN=%d TSC=%d ARFCN=%d -> %s + grgsm -m TCHF (pid %s)"
          % (tn, tsc, arfcn, TCH_CFG_PATH, _tch_proc.pid), flush=True)

def check_assignment(by, fn):
    """Detecte un ASSIGNMENT COMMAND (RR PD=0x06, MT=0x2e) sur le SDCCH DL.

    Channel Description 2 (GSM 04.08 10.5.2.5), 3 octets, decodage aligne sur
    `struct gsm48_chan_desc` de libosmocore :
        b0 : chan_nr           -> TN = b0 & 0x07
        b1 : tsc(3) h(1) arfcn_high(2) spare(2)
             TSC = (b1>>5)&7   H = (b1>>4)&1   ARFCN_high = b1 & 0x03
        b2 : ARFCN_low         -> ARFCN = (ARFCN_high<<8) | b2   [si H=0]
    """
    for off in range(2, min(7, len(by) - 5)):
        if by[off] != 0x06 or by[off + 1] != 0x2e:
            continue
```

```
        b0, b1, b2 = by[off + 2], by[off + 3], by[off + 4]
        tn  = b0 & 0x07
        tsc = (b1 >> 5) & 0x07
        h   = (b1 >> 4) & 0x01
        if h:
            # ⚠ REFUS EXPLICITE. Cette chaine ne gere PAS le saut de frequence
            # (cf. la meme limite cote injection UL). Fabriquer une ARFCN a
            # partir des champs MAIO/HSN donnerait un nombre FAUX en silence,
            # et le decodeur ecouterait a cote sans que rien ne le signale.
            print("[si-bridge] ASSIGNMENT COMMAND avec SAUT DE FREQUENCE (H=1) : "
                  "TCH NON arme (chaine sans hopping). FN=%d" % fn, flush=True)
            return
        arfcn = ((b1 & 0x03) << 8) | b2
        print("[si-bridge] ASSIGNMENT COMMAND FN=%d -> chan_nr=0x%02x TN=%d TSC=%d ARFCN=%d"
              % (fn, b0, tn, tsc, arfcn), flush=True)
        arm_tch(tn, tsc, arfcn)
        return

def reader_loop(proc, tag):
    # log brut par grgsm (diagnostic decipher) : /root/grgsm_clair.raw / _ciph.raw
    try: raw = open("/root/grgsm_%s.raw" % tag, "w")
    except Exception: raw = None
    for line in proc.stdout:
        if raw:
            try: raw.write(line); raw.flush()
            except Exception: pass
        with _hl_lock:
            handle_line(line)
    if raw:
        try: raw.close()
        except Exception: pass
    print("[si-bridge] grgsm %s termine (n=%d nsch=%d nsd=%d nsa=%d)"
          % (tag, n, nsch, nsd, nsa), flush=True)

def read_kc():
    try:
        with open("/dev/shm/calypso_kc", "rb") as f:
            b = f.read(32)
    except OSError:
        return (0, 0, None)
    if len(b) < 14:
        return (0, 0, None)
    seq = int.from_bytes(b[0:4], "little")
    if seq == 0:
        return (0, 0, None)
    return (seq, b[4], b[6:14].hex())

# --- grgsm CLAIR : toujours vivant (watchdog respawn si crash), JAMAIS tue au cipher ---
clear = {"proc": None}
def ensure_clear():
    p = clear["proc"]
    if p is None or p.poll() is not None:
        clear["proc"] = spawn_grgsm(CF)
        threading.Thread(target=reader_loop, args=(clear["proc"], "clair"),
                         daemon=True).start()
        print("[si-bridge] grgsm CLAIR spawn sur %s (jamais tue au cipher)" % CF, flush=True)

ensure_clear()

# --- grgsm CHIFFRE : spawne/tue selon le Kc, TOUJOURS sur CIPH_CF (jamais sur CF) ---
ciph = {"applied": (0, None), "proc": None}
while True:
    time.sleep(0.5)
    ensure_clear()                      # watchdog du grgsm clair
    seq, algo, kc = read_kc()
    want = (algo, kc) if (seq and algo in (1, 2, 3)) else (0, None)
```

```
    if want == ciph["applied"]:
        continue
    # etat Kc change -> on (re)cycle UNIQUEMENT le grgsm chiffre (sur CIPH_CF)
    p = ciph["proc"]
    if p is not None:
        try: p.kill(); p.wait(timeout=2)
        except Exception: pass
        ciph["proc"] = None
    ciph["applied"] = want
    if want[0]:
        np = spawn_grgsm(CIPH_CF, algo=want[0], kc_hex=want[1], binary=GRGSM_CIPH,
                         errlog="/root/grgsm_ciph.err")
        ciph["proc"] = np
        threading.Thread(target=reader_loop, args=(np, "chiffre"),
                         daemon=True).start()
        print("[si-bridge] grgsm CHIFFRE spawn sur %s : A5/%d kc=%s (decipher DL)"
              % (CIPH_CF, want[0], want[1]), flush=True)
    else:
        print("[si-bridge] Kc efface -> grgsm chiffre arrete (clair seul)", flush=True)
```

3.263 /opt/GSM/osmo-operator/opt-gsm/si_bridge_loop.sh

598 octets, 12 lignes → 12 lignes

```
#!/bin/bash
# Lance si_bridge (2 grgsm : clair toujours-on + chiffre spawne sur Kc) en boucle.
# --line-buffered : sinon le grep bufferise par blocs (4KB) -> logs en retard,
# impossible de voir le respawn/Kc en temps reel. On laisse passer aussi les
# erreurs/tracebacks Python (sinon invisibles).
source /root/.env/bin/activate 2>/dev/null
echo "[si-bridge] 2 grgsm (clair iq_grgsm.fifo + chiffre iq_grgsm_ciph.fifo) -> feed_*"
while true; do
  python3 ${GSM_ROOT}/si_bridge.py /tmp/iq_grgsm.fifo 2>&1 \
    | grep --line-buffered -E "si-bridge|SI|Traceback|Error|error|Exception"
  sleep 2
done
```

3.264 /opt/GSM/osmo-operator/packaging/.gitignore

6 octets, 1 lignes → 1 lignes

```
dist/
```

3.265 /opt/GSM/osmo-operator/packaging/apt-fast-install.sh

8534 octets, 186 lignes → 186 lignes

```
#!/bin/bash
# ================================================================================
#  apt-fast-install - UNE seule facon d installer apt-fast, partout
# ================================================================================
#
#  Utilise par : Dockerfile, Dockerfile.stp (COPY puis RUN), build-iso.sh (dans
#  le chroot du rootfs), build.sh et tools/make-docker-image.sh (sur l hote).
#  Apres ce script, tout le monde parle a apt de la meme facon :
#
#      apt-fast update
#      apt-fast install -y --no-install-recommends <paquets>
#      apt-fast build-dep -y <paquet>
```

```
#
#  apt-fast telecharge en parallele (aria2), dpkg installe ensuite comme
#  d habitude : le gain porte sur le telechargement, et il est net des qu on
#  pose plusieurs centaines de paquets. Si GitHub est injoignable, on pose un
#  apt-fast qui appelle apt-get : la commande existe toujours, le build
#  continue, plus lentement.
#
#  DLDIR = /var/cache/apt/archives : apt-fast telecharge la ou apt range ses
#  paquets. C est ce qui permet a build-iso.sh de monter un cache PERSISTANT
#  a cet endroit dans le chroot (les .deb du rootfs sont telecharges une fois,
#  pour toutes les passes et toutes les reconstructions).
#
#  Il pose aussi /etc/apt/apt.conf.d/90osmo-operator - les reglages apt qui
#  etaient repetes dans le chroot de l ISO et en ligne de commande sur l hote :
#  pas de traductions, 3 essais, pipeline HTTP, dpkg sans pty. Ce sont des
#  reglages de TELECHARGEMENT ; ils ne touchent ni aux recommends ni a la
#  facon dont dpkg ecrit.
#
#  Idempotent : deja installe -> ne refait que la conf.
# ============================================================================
set -euo pipefail
export DEBIAN_FRONTEND=noninteractive

APT_FAST_URL="${APT_FAST_URL:-https://raw.githubusercontent.com/ilikenwf/apt-fast/master/apt-fast}"
BIN=/usr/local/sbin/apt-fast

# Timeout 30 s et pas de pipelining : un miroir qui ne repond pas echoue en
# 30 s et passe a l essai suivant, au lieu de bloquer 120 s (le defaut) fois
# trois essais, en silence. Le pipelining HTTP n apporte rien ici (aria2
# parallelise deja) et c est une source connue de connexions qui restent
# pendues derriere certains equilibreurs.
cat > /etc/apt/apt.conf.d/90osmo-operator <<'CONF'
// osmo-operator : reglages apt communs (image docker, rootfs ISO, hote)
Acquire::Languages "none";
Acquire::Retries "3";
Acquire::http::Timeout "30";
Acquire::http::Pipeline-Depth "0";
Dpkg::Use-Pty "0";
// Avant CHAQUE dpkg : les .deb a installer sont decompresses de bout en bout ;
// un fichier corrompu est supprime du cache et l installation s arrete, a
// relancer (osmo-apt-get, sous apt-fast, relance tout seul). Voir
// /usr/local/sbin/osmo-deb-check.
DPkg::Pre-Install-Pkgs { "/usr/local/sbin/osmo-deb-check"; };
DPkg::Tools::Options::/usr/local/sbin/osmo-deb-check::Version "1";
CONF

# ── Le cache apt est PERSISTANT partout ici : un .deb corrompu y reste ──────
# [2026-09-04] "Corrupted block detected" a l installation de qemu-user-static
# sur l hote : le fichier en cache avait la taille attendue et un mauvais
# SHA256 - un telechargement altere sur un miroir capricieux. apt ne revérifie
# pas un fichier deja present dans /var/cache/apt/archives, et ce cache ne
# s efface jamais chez nous (Keep-Downloaded-Packages, montage cache du docker
# build, cache persistant du chroot de l ISO) : le fichier pourri serait reste
# la a chaque build. Deux pieces :
#   osmo-deb-check   hook DPkg::Pre-Install-Pkgs : apt lui passe les chemins des
#                    .deb qu il va installer ; chacun est decompresse en entier
#                    (dpkg-deb --fsys-tarfile) ; les mauvais sont SUPPRIMES, un
#                    marqueur est pose et apt s arrete avant dpkg. Le prochain
#                    apt retelecharge ce qui manque, avec verification du hash.
#   osmo-apt-get     le "apt-get" que lance apt-fast (_APTMGR) : si apt echoue
#                    et que le marqueur est la, il relance une fois. Pour un
#                    apt-get appele directement, le message dit de relancer.
cat > /usr/local/sbin/osmo-deb-check <<'CHK'
#!/bin/sh
# osmo-deb-check - hook apt (DPkg::Pre-Install-Pkgs, protocole 1 : un chemin
# de .deb par ligne sur stdin). Supprime du cache les .deb dont l archive de
```

```
# donnees ne se decompresse pas, et refuse l installation s il y en a eu.
MARK=/run/osmo-deb-check.bad
rm -f "$MARK"
bad=0; n=0
while IFS= read -r f; do
    [ -f "$f" ] || continue
    n=$((n + 1))
    if ! dpkg-deb --fsys-tarfile "$f" >/dev/null 2>&1; then
        echo "osmo-deb-check: CORROMPU, supprime du cache : $f" >&2
        rm -f "$f"
        bad=$((bad + 1))
    fi
done
if [ "$bad" -gt 0 ]; then
    echo "$bad" > "$MARK"
    echo "osmo-deb-check: $bad paquet(s) corrompu(s) retire(s) sur $n - relancez la commande, apt les
↪  retelechargera" >&2
    exit 1
fi
exit 0
CHK
chmod 755 /usr/local/sbin/osmo-deb-check
cat > /usr/local/sbin/osmo-apt-get <<'OAG'
#!/bin/sh
# osmo-apt-get - apt-get avec UNE relance quand osmo-deb-check a retire des
# .deb corrompus du cache (marqueur /run/osmo-deb-check.bad).
MARK=/run/osmo-deb-check.bad
rm -f "$MARK"
apt-get "$@" && exit 0
rc=$?
if [ -f "$MARK" ]; then
    echo "osmo-apt-get: cache apt nettoye ($(cat "$MARK") .deb corrompu(s)) - relance de : apt-get $*" >&2
    rm -f "$MARK"
    exec apt-get "$@"
fi
exit "$rc"
OAG
chmod 755 /usr/local/sbin/osmo-apt-get

# ── Le miroir Ubuntu : celui que l appelant a mesure (packaging/apt-mirror.sh)
# OSMO_UBUNTU_MIRROR=http://... remplace archive.ubuntu.com dans les sources
# du systeme ou l on est - l image docker (build-arg), le rootfs de l ISO.
# Jamais sur l hote sans qu on le demande : build.sh ne l exporte pas.
if [ -n "${OSMO_UBUNTU_MIRROR:-}" ]; then
    _m="${OSMO_UBUNTU_MIRROR%/}"
    for _f in /etc/apt/sources.list /etc/apt/sources.list.d/ubuntu.sources; do
        [ -f "$_f" ] || continue
        # archive ET security : security.ubuntu.com vit sur les memes adresses
        # Canonical qu archive (91.189.9x, 185.125.190.x) - quand l un est
        # injoignable, l autre aussi. Tout miroir Ubuntu sert noble-security.
        sed -i -E "s#https?://(archive|[a-z]{2}\.archive|security)\.ubuntu\.com/ubuntu/?#${_m}#g" "$_f"
    done
    echo "[apt-fast] miroir Ubuntu : $_m"
fi

# aria2 et curl sont les seuls paquets que l on pose encore avec apt-get :
# apt-fast n existe pas avant eux.
if ! command -v aria2c >/dev/null 2>&1 || ! command -v curl >/dev/null 2>&1; then
    apt-get update -qq
    apt-get install -y --no-install-recommends aria2 curl ca-certificates
fi

if [ ! -x "$BIN" ] || ! grep -q 'aria2' "$BIN" 2>/dev/null; then
    if curl -fsSL --retry 3 -o "$BIN.tmp" "$APT_FAST_URL" && grep -q 'aria2' "$BIN.tmp"; then
        mv -f "$BIN.tmp" "$BIN"
        echo "[apt-fast] installe depuis $APT_FAST_URL"
```

```
    else
        rm -f "$BIN.tmp"
        printf '#!/bin/sh\n# apt-fast de repli (telechargement impossible) : apt-get tel quel\nexec apt-get
        ↪  "$@"\n' > "$BIN"
        echo "[apt-fast] WARN: telechargement impossible - repli sur apt-get sous le meme nom" >&2
    fi
    chmod 755 "$BIN"
fi

# La conf d apt-fast : 16 connexions, aucune question. Ecrite a chaque fois,
# elle est courte et c est ce qui garantit que tous les environnements se
# ressemblent.
cat > /etc/apt-fast.conf <<'CONF'
_APTMGR=/usr/local/sbin/osmo-apt-get
DOWNLOADBEFORE=true
_MAXNUM=16
_MAXCONPERSRV=10
_SPLITCON=8
_MINSPLITSZ="1M"
_PIECEALGO="default"
DLDIR=/var/cache/apt/archives
DLLIST=/tmp/apt-fast.list
APT_FAST_TIMEOUT=60
VERBOSE_OUTPUT=false
CONF
mkdir -p /var/cache/apt/archives/partial

# Garder les .deb telecharges : c est le principe du cache. Les images docker
# posent /etc/apt/apt.conf.d/docker-clean, qui les efface apres chaque
# installation - il part. Keep-Downloaded-Packages vaut pour apt >= 1.2.
rm -f /etc/apt/apt.conf.d/docker-clean
cat > /etc/apt/apt.conf.d/91osmo-keep-debs <<'CONF'
// osmo-operator : les paquets telecharges restent dans /var/cache/apt/archives
Binary::apt::APT::Keep-Downloaded-Packages "true";
APT::Keep-Downloaded-Packages "true";
CONF
# Si /usr/local/sbin n est pas dans le PATH de l appelant (root sans sudo, PATH
# perso), la commande resterait introuvable juste apres "pret" : un lien dans
# /usr/local/bin, qui y est toujours, regle le cas sans toucher au PATH.
if ! command -v apt-fast >/dev/null 2>&1; then
    ln -sfn "$BIN" /usr/local/bin/apt-fast
    echo "[apt-fast] /usr/local/sbin absent du PATH - lien /usr/local/bin/apt-fast"
fi
echo "[apt-fast] pret : $BIN"
```

3.266 /opt/GSM/osmo-operator/packaging/apt-mirror.sh

3849 octets, 79 lignes → 79 lignes

```
#!/bin/bash
# ============================================================================
#  apt-mirror - LE miroir Ubuntu le plus rapide d ici, mesure, pas suppose
# ============================================================================
#
#  Affiche une URL de miroir (http://.../ubuntu) sur la sortie standard.
#
#  [2026-09-04] archive.ubuntu.com repondait a 3 Ko/s depuis l hote de build
#  (20 s et un timeout pour un InRelease de 250 Ko) quand fr.archive, ovh et
#  ikoula servaient le meme fichier en 0,1 s. Le build docker restait muet a
#  l etape 3/64 (apt-get update -qq) pendant dix minutes, sans rien dire.
#  L adresse "archive.ubuntu.com" est un point d entree geo-equilibre, pas
#  un engagement de debit : on MESURE.
#
#  Candidats : $OSMO_UBUNTU_MIRRORS (liste separee par des espaces) si donne,
```

```
#  sinon les miroirs que mirrors.ubuntu.com propose pour CE pays (GeoIP, les
#  quatre premiers) plus archive.ubuntu.com. Chaque candidat telecharge
#  dists/<suite>/InRelease avec 5 s au maximum ; le plus rapide gagne. Si rien
#  ne repond, archive.ubuntu.com, comme avant.
#
#  OSMO_UBUNTU_MIRROR=http://... court-circuite tout : c est ce miroir, sans
#  mesure (un miroir local, un proxy apt).
#
#  Usage :  MIRROR="$(packaging/apt-mirror.sh [suite])"   (defaut : noble)
#           Les mesures vont sur stderr, l URL seule sur stdout.
#           packaging/apt-mirror.sh --list [suite]
#           Tous les miroirs qui repondent, "<ms>\t<url>", du plus rapide au
#           plus lent, sur stdout - c est ce que lit l installeur du live pour
#           proposer le choix (osmo-install, page packagechooser@mirror).
# ============================================================================
set -uo pipefail
LIST=0
if [ "${1:-}" = "--list" ]; then LIST=1; shift; fi
SUITE="${1:-noble}"
DEFAULT="http://archive.ubuntu.com/ubuntu"

if [ -n "${OSMO_UBUNTU_MIRROR:-}" ] && [ "$LIST" = 0 ]; then
    echo "${OSMO_UBUNTU_MIRROR%/}"; exit 0
fi
command -v curl >/dev/null 2>&1 || { [ "$LIST" = 1 ] && printf '0\t%s\n' "$DEFAULT" || echo "$DEFAULT"; exit 0; }

CANDIDATES="${OSMO_UBUNTU_MIRRORS:-}"
if [ -z "$CANDIDATES" ]; then
    # mirrors.txt (GeoIP) renvoie parfois... archive.ubuntu.com lui-meme. On
    # y ajoute le miroir du pays de la locale (fr.archive, de.archive...), deux
    # gros hebergeurs europeens et kernel.org : de quoi toujours avoir un
    # miroir qui repond.
    CANDIDATES="$(curl -fsSL -m 5 http://mirrors.ubuntu.com/mirrors.txt 2>/dev/null \
                  | grep -E '^https?://' | head -4 | tr '\n' ' ')"
    _cc="$(echo "${LC_ALL:-${LANG:-}}" | sed -nE 's/^[a-z]+_([A-Z]{2}).*/\1/p' | tr 'A-Z' 'a-z')"
    [ -n "$_cc" ] && CANDIDATES="$CANDIDATES http://${_cc}.archive.ubuntu.com/ubuntu"
    CANDIDATES="$CANDIDATES http://ubuntu.mirrors.ovh.net/ubuntu http://mirror.ubuntu.ikoula.com/ubuntu
                http://mirrors.edge.kernel.org/ubuntu $DEFAULT"
fi
# Sans doublon, dans l ordre.
CANDIDATES="$(for m in $CANDIDATES; do echo "${m%/}"; done | awk '!seen[$0]++' | tr '\n' ' ')"

best=""; best_t=999999; results=""
for m in $CANDIDATES; do
    m="${m%/}"
    # Un seul fichier, petit et toujours present ; 5 s max, et le temps est
    # celui de curl lui-meme (ms), pas une horloge externe.
    t="$(curl -s -m 5 -o /dev/null -w '%{http_code} %{time_total}' "$m/dists/$SUITE/InRelease" 2>/dev/null ||
     ↪  true)"
    code="${t%% *}"; secs="${t##* }"
    if [ "$code" = "200" ]; then
        ms="$(awk -v s="$secs" 'BEGIN{printf "%d", s*1000}')"
        printf '  %-52s %6d ms\n' "$m" "$ms" >&2
        results="${results}${ms}    ${m}
"
        if [ "$ms" -lt "$best_t" ]; then best="$m"; best_t="$ms"; fi
    else
        printf '  %-52s   -    (%s)\n' "$m" "${code:-timeout}" >&2
    fi
done
if [ "$LIST" = 1 ]; then
    printf '%s' "$results" | sort -n
    exit 0
fi
echo "${best:-$DEFAULT}"
```

3.267 /opt/GSM/osmo-operator/packaging/build-debs.sh

14840 octets, 298 lignes → 298 lignes

```
#!/bin/bash
# ============================================================================
#  packaging/build-debs.sh - un paquet .deb par composant du banc
# ============================================================================
#
#      osmo-operator     ce depot, tel quel, en /opt/GSM/osmo-operator
#                        (+ menu, icones, /usr/bin/osmo-start-direct)
#      pont              pont/ - le transceiver Python (pont/pont.py)
#      qemu-calypso      qosmo : qemu-system-arm (machine calypso),
#                        run.sh + run_modules + environnement + cfgs +
#                        tmux_modules + keymaps, en /opt/GSM/qosmo
#      calypso-firmware  /opt/GSM/firmware/board/compal_e88/layer1.highram.*
#
#  Les chemins d installation sont CEUX du banc (/opt/GSM/...) : tout le depot
#  les nomme en dur (bench.env, start-native.sh, les unites systemd), et c est
#  ce qui permet a une machine installee par paquets de tourner exactement
#  comme l ISO. Le prix : un paquet refuse de s installer par-dessus un clone
#  git au meme endroit (preinst) - on ne melange pas un arbre gere par git et
#  un arbre gere par dpkg.
#
#      ./build-debs.sh                  les quatre, dans packaging/dist/
#      ./build-debs.sh --only pont,calypso-firmware
#      ./build-debs.sh --out /tmp/debs
#      ./build-debs.sh --no-strip       garde les symboles de qemu-system-arm
#
#  Outils : dpkg-deb (dpkg), git, strip (binutils, sinon --no-strip), ldd.
#  Pas de debhelper : chaque paquet est un arbre + DEBIAN/control construit a
#  la main, avec --root-owner-group (pas besoin de fakeroot).
#
#  Sources, surchargeables par l environnement :
#      OSMO_OPERATOR_SRC  (defaut : le depot qui contient ce script)
#      QOSMO_SRC          (defaut : /opt/GSM/qosmo)
#      FIRMWARE_SRC       (defaut : /opt/GSM/firmware)
# ----------------------------------------------------------------------------
set -euo pipefail

HERE="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
: "${OSMO_OPERATOR_SRC:=$(cd "$HERE/.." && pwd)}"
: "${QOSMO_SRC:=/opt/GSM/qosmo}"
: "${FIRMWARE_SRC:=/opt/GSM/firmware}"
OUT="$HERE/dist"
ONLY=""
STRIP=1
while [ $# -gt 0 ]; do
    case "$1" in
        --out)      OUT="${2:?}"; shift ;;
        --out=*)    OUT="${1#*=}" ;;
        --only)     ONLY="${2:?}"; shift ;;
        --only=*)   ONLY="${1#*=}" ;;
        --no-strip) STRIP=0 ;;
        -h|--help)  sed -n '2,33p' "$0" | sed 's/^#\{1,2\} \{0,1\}//'; exit 0 ;;
        *) echo "option inconnue : $1" >&2; exit 2 ;;
    esac
    shift
done
for t in dpkg-deb git; do
    command -v "$t" >/dev/null 2>&1 || { echo "outil manquant : $t" >&2; exit 1; }
done
mkdir -p "$OUT"
WORK="$(mktemp -d "${TMPDIR:-/var/tmp}/osmo-debs.XXXXXX")"
trap 'rm -rf "$WORK"' EXIT
```

```
MAINT="osmo-operator <bastienbaranoff@gmail.com>"
ARCH_HOST="$(dpkg --print-architecture)"

wanted() { [ -z "$ONLY" ] || case ",$ONLY," in *",$1,"*) return 0;; *) return 1;; esac; }

# Version = 0.1+git<date du dernier commit>.<hash court> de l arbre source.
# Deux constructions du meme commit donnent le meme numero : c est voulu.
tree_version() {
    local d="$1"
    if git -C "$d" rev-parse --is-inside-work-tree >/dev/null 2>&1; then
        printf '0.1+git%s.%s' \
            "$(git -C "$d" log -1 --format=%cd --date=format:%Y%m%d)" \
            "$(git -C "$d" rev-parse --short HEAD)"
    else
        printf '0.1+local%s' "$(date +%Y%m%d)"
    fi
}

# preinst commun : refuse un clone git au meme endroit (sauf mise a jour du
# paquet lui-meme, ou dpkg passe "upgrade").
write_preinst_guard() {   # $1=DEBIAN dir  $2=chemin installe
    cat > "$1/preinst" <<PRE
#!/bin/sh
set -e
if [ "\$1" = install ] && [ -d "$2/.git" ]; then
    echo "$2 est un clone git : ce paquet ne s installe pas par-dessus." >&2
    echo "Deplacez ce clone (ou supprimez-le) avant d installer le paquet." >&2
    exit 1
fi
exit 0
PRE
    chmod 755 "$1/preinst"
}

write_control() {   # $1=DEBIAN dir, puis champs sur stdin
    cat > "$1/control"
}

installed_size() { du -sk --exclude=DEBIAN "$1" | cut -f1; }

build_pkg() {   # $1=staging  $2=nom  $3=version  $4=arch
    printf 'Installed-Size: %s\n' "$(installed_size "$1")" >> "$1/DEBIAN/control"
    find "$1" -path "$1/DEBIAN" -prune -o -type d -exec chmod 755 {} + 2>/dev/null || true
    dpkg-deb --build --root-owner-group "$1" "$OUT/${2}_${3}_${4}.deb" >/dev/null
    echo "  $OUT/${2}_${3}_${4}.deb  ($(du -h "$OUT/${2}_${3}_${4}.deb" | cut -f1))"
}

# Dependances runtime d un binaire, par ldd -> dpkg -S. Les bibliotheques que
# dpkg ne connait pas (/usr/local/lib) sont ignorees : elles ne viennent d aucun
# paquet, donc aucun paquet ne peut les garantir.
shlib_depends() {
    # ldd rend /lib/x86_64-linux-gnu/... ; sur un systeme a /usr fusionne dpkg
    # ne connait souvent que /usr/lib/... : on tente les deux. Tout est en
    # "|| true" : sous pipefail, une seule bibliotheque inconnue faisait
    # echouer la substitution entiere, et set -e arretait le script.
    local so
    { ldd "$1" 2>/dev/null | awk '/=> \//{print $3}' || true; } | while read -r so; do
        { dpkg -S "$so" 2>/dev/null || dpkg -S "/usr$so" 2>/dev/null || true; } | cut -d: -f1 | head -1
    done | grep -v '^$' | sort -u | paste -sd, - | sed 's/,/, /g' || true
}

# ── osmo-operator ──────────────────────────────────────────────────
if wanted osmo-operator; then
    V="$(tree_version "$OSMO_OPERATOR_SRC")"
    P="$WORK/osmo-operator"; D="$P/opt/GSM/osmo-operator"
    mkdir -p "$D" "$P/DEBIAN" "$P/usr/bin" "$P/usr/share/applications" \
```

```
            "$P/usr/share/osmo-operator/icons" "$P/usr/share/icons/hicolor/scalable/apps"
    tar -C "$OSMO_OPERATOR_SRC" \
        --exclude=.git --exclude=./pont --exclude=./packaging/dist \
        --exclude='__pycache__' --exclude='*.iso' --exclude='vty-debug-dump-*' \
        -cf - . | tar -C "$D" -xf -
    ln -s /opt/GSM/osmo-operator/start-native.sh "$P/usr/bin/osmo-start-native"
    ln -s /opt/GSM/osmo-operator/start-native.sh "$P/usr/bin/osmo-start-direct"   # ancien nom
    for e in osmo-launch osmo-multi; do
        install -m644 "$OSMO_OPERATOR_SRC/data/desktop/$e.desktop" "$P/usr/share/applications/$e.desktop"
        install -m644 "$OSMO_OPERATOR_SRC/data/$e.svg" "$P/usr/share/osmo-operator/icons/$e.svg"
        install -m644 "$OSMO_OPERATOR_SRC/data/$e.svg" "$P/usr/share/icons/hicolor/scalable/apps/$e.svg"
    done
    install -m644 "$OSMO_OPERATOR_SRC/data/osmo-tutorial.svg" "$P/usr/share/osmo-operator/icons/" 2>/dev/null ||
     ↪  true
    install -m644 "$OSMO_OPERATOR_SRC/data/tutorial.html" "$P/usr/share/osmo-operator/" 2>/dev/null || true
    write_preinst_guard "$P/DEBIAN" /opt/GSM/osmo-operator
    cat > "$P/DEBIAN/postinst" <<'POST'
#!/bin/sh
set -e
command -v update-desktop-database >/dev/null 2>&1 && update-desktop-database /usr/share/applications || true
command -v gtk-update-icon-cache >/dev/null 2>&1 && gtk-update-icon-cache -f -q /usr/share/icons/hicolor || true
exit 0
POST
    chmod 755 "$P/DEBIAN/postinst"
    write_control "$P/DEBIAN" <<CTL
Package: osmo-operator
Version: $V
Section: net
Priority: optional
Architecture: all
Maintainer: $MAINT
Depends: bash, python3, socat, netcat-openbsd, tcpdump, tmux, expect, whiptail, iproute2, sqlite3, git, pont (>=
 ↪  0.1), qemu-calypso, calypso-firmware
Recommends: pulseaudio, pulseaudio-utils, asterisk, wireshark
Homepage: https://github.com/bbaranoff/osmo-operator
Description: banc GSM/EGPRS Osmocom - lanceur, configurations, outils
 Le depot osmo-operator installe en /opt/GSM/osmo-operator : start-native.sh
 (lanceur natif, qui delegue a qosmo/run.sh), les gabarits de
 configuration Osmocom et Asterisk, les scripts reseau/WAN/SS7, les checks, le
 menu et les icones du bureau. Les demons Osmocom eux-memes (osmo-stp, hlr,
 msc, bsc, bts...) ne sont pas dans ce paquet : ils viennent du depot Osmocom
 ou d une compilation locale (/usr/local).
CTL
    build_pkg "$P" osmo-operator "$V" all
fi

# ── pont ──────────────────────────────────────────────────────────────
if wanted pont; then
    V="$(tree_version "$OSMO_OPERATOR_SRC")"
    P="$WORK/pont"; D="$P/opt/GSM/osmo-operator/pont"
    mkdir -p "$D" "$P/DEBIAN"
    install -m644 "$OSMO_OPERATOR_SRC"/pont/*.py "$D/"
    # [2026-09-23] Le pont DSP (pont_dsp.py -> pont.dsp.main).
    mkdir -p "$D/dsp"
    install -m644 "$OSMO_OPERATOR_SRC"/pont/dsp/*.py "$D/dsp/"
    install -m644 "$OSMO_OPERATOR_SRC/pont/README.md" "$D/" 2>/dev/null || true
    chmod 755 "$D/pont.py" "$D/pont_dsp.py"
    write_preinst_guard "$P/DEBIAN" /opt/GSM/osmo-operator
    write_control "$P/DEBIAN" <<CTL
Package: pont
Version: $V
Section: net
Priority: optional
Architecture: all
Maintainer: $MAINT
Depends: python3 (>= 3.10), python3-numpy
```

```
Enhances: osmo-operator
Homepage: https://github.com/bbaranoff/osmo-operator
Description: pont TRX entre osmo-bts-trx et le Calypso emule (qosmo)
 Transceiver TRX-UDP en Python : se presente a osmo-bts-trx sur 5700-5702,
 decode le descendant vers GSMTAP 4730/4731 (lu par le modele QEMU) et encode
 le montant depuis les sidebands /dev/shm/calypso_*. Installe en
 /opt/GSM/osmo-operator/pont, la ou start-native.sh l attend (PONT_PY).
 Requiert libosmocoding (libosmocore) accessible par ld.so - fournie par la
 pile Osmocom, pas par un paquet Debian.
CTL
    build_pkg "$P" pont "$V" all
fi

# ── qemu-calypso ──────────────────────────────────────────────────────
if wanted qemu-calypso; then
    BIN="$QOSMO_SRC/build/qemu-system-arm"
    [ -x "$BIN" ] || { echo "qemu-system-arm absent : $BIN (construisez qosmo d abord)" >&2; exit 1; }
    V="$(tree_version "$QOSMO_SRC")"
    P="$WORK/qemu-calypso"; D="$P/opt/GSM/qosmo"
    mkdir -p "$D/build" "$D/share/qemu" "$P/DEBIAN"
    for d in run_modules environnement cfgs tmux_modules opt-gsm-scripts; do
        [ -d "$QOSMO_SRC/$d" ] || continue
        tar -C "$QOSMO_SRC" --exclude='__pycache__' -cf - "$d" | tar -C "$D" -xf -
    done
    install -m755 "$QOSMO_SRC/run.sh" "$D/run.sh"
    for f in LICENSE COPYING; do [ -f "$QOSMO_SRC/$f" ] && install -m644 "$QOSMO_SRC/$f" "$D/$f"; done
    # Le binaire reste en build/ : bench.env le nomme la (QEMU_BIN), et QEMU
    # calcule son datadir PAR RAPPORT au binaire (build/../share/qemu) - d ou
    # les keymaps en share/qemu/keymaps, sinon "could not read keymap file".
    install -m755 "$BIN" "$D/build/qemu-system-arm"
    if [ "$STRIP" = 1 ] && command -v strip >/dev/null 2>&1; then
        strip --strip-unneeded "$D/build/qemu-system-arm" || true
    fi
    # Lanceur C qosmo (tools/qosmo-launch) : c'est lui que 40-qemu.sh
    # appelle a la place de qemu-system-arm. Compile dans son dossier, livre
    # dans /usr/local/bin/qosmo ; la source part avec l'arbre.
    if [ -f "$QOSMO_SRC/tools/qosmo-launch/qosmo-launch.c" ]; then
        mkdir -p "$D/tools/qosmo-launch" "$P/usr/local/bin"
        install -m644 "$QOSMO_SRC/tools/qosmo-launch/qosmo-launch.c" "$QOSMO_SRC/tools/qosmo-launch/Makefile"
         ↪  "$D/tools/qosmo-launch/"
        make -s -C "$QOSMO_SRC/tools/qosmo-launch" ALIAS=qosmo TREE=/opt/GSM/qosmo qosmo >/dev/null \
            || { echo "compilation du lanceur qosmo echouee ($QOSMO_SRC/tools/qosmo-launch)" >&2; exit 1; }
        install -m755 "$QOSMO_SRC/tools/qosmo-launch/qosmo" "$P/usr/local/bin/qosmo"
    fi
    # Console gdb en telnet (run_modules/44-gdb-telnet.sh) : le serveur, le
    # generateur du panneau et le panneau genere. gdb-multiarch et telnet
    # passent en Depends : sans eux le module se declare en echec (non bloquant).
    if [ -f "$QOSMO_SRC/tools/gdb-telnet.py" ]; then
        install -m755 "$QOSMO_SRC/tools/gdb-telnet.py" "$D/tools/gdb-telnet.py"
        install -m755 "$QOSMO_SRC/tools/gdb_cmd.sh" "$D/tools/gdb_cmd.sh"
        (cd "$D/tools" && bash ./gdb_cmd.sh >/dev/null) || { echo "gdb_cmd.sh : generation de cmd.gdb echouee"
         ↪  >&2; exit 1; }
    fi
    km=""
    for c in "$QOSMO_SRC/build/pc-bios/keymaps" "$QOSMO_SRC/pc-bios/keymaps" \
             /opt/GSM/qemu-install/share/qemu/keymaps /usr/share/qemu/keymaps; do
        [ -f "$c/en-us" ] && { km="$c"; break; }
    done
    [ -n "$km" ] || { echo "keymaps QEMU introuvables (en-us)" >&2; exit 1; }
    cp -a "$km" "$D/share/qemu/keymaps"
    DEPS="$(shlib_depends "$D/build/qemu-system-arm")"
    write_preinst_guard "$P/DEBIAN" /opt/GSM/qosmo
    write_control "$P/DEBIAN" <<CTL
Package: qemu-calypso
Version: $V
Section: otherosfs
```

```
Priority: optional
Architecture: $ARCH_HOST
Maintainer: $MAINT
Depends: ${DEPS:+$DEPS, }bash, socat, tmux, tcpdump, procps, psmisc, gawk, python3, iproute2, gdb-multiarch,
 ↪  telnet
Recommends: osmo-operator, calypso-firmware
Homepage: https://github.com/bbaranoff/qosmO
Description: QEMU avec la machine calypso (telephone Osmocom-BB emule) et son lanceur
 qemu-system-arm construit depuis qosmo, avec le modele de SoC Calypso
 (ARM946, TPU/TSP, SIM, couche 1 gr-gsm), le lanceur run.sh et ses modules
 (run_modules/), l environnement du banc (environnement/bench.env), les
 configurations du mobile (cfgs/) et les dispositions tmux. Installe en
 /opt/GSM/qosmo, sans les sources ni l arbre de construction.
CTL
    build_pkg "$P" qemu-calypso "$V" "$ARCH_HOST"
fi

# ── calypso-firmware ──────────────────────────────────────────────
if wanted calypso-firmware; then
    FW="$FIRMWARE_SRC/board/compal_e88"
    [ -f "$FW/layer1.highram.elf" ] || { echo "firmware absent : $FW/layer1.highram.elf" >&2; exit 1; }
    V="$(tree_version "$FIRMWARE_SRC")"
    P="$WORK/calypso-firmware"; D="$P/opt/GSM/firmware/board/compal_e88"
    mkdir -p "$D" "$P/DEBIAN"
    install -m644 "$FW/layer1.highram.elf" "$FW/layer1.highram.bin" "$D/"
    [ -f "$FIRMWARE_SRC/COPYING" ] && install -m644 "$FIRMWARE_SRC/COPYING" "$P/opt/GSM/firmware/COPYING"
    write_preinst_guard "$P/DEBIAN" /opt/GSM/firmware
    write_control "$P/DEBIAN" <<CTL
Package: calypso-firmware
Version: $V
Section: misc
Priority: optional
Architecture: all
Maintainer: $MAINT
Homepage: https://github.com/bbaranoff/firmware
Description: firmware layer1 Osmocom-BB (compal_e88) pour le Calypso emule
 layer1.highram.elf (charge par qemu-system-arm -kernel) et layer1.highram.bin
 (envoye par osmocon), pre-compiles depuis osmocom-bb. Chemin fixe attendu par
 bench.env : /opt/GSM/firmware/board/compal_e88/.
CTL
    build_pkg "$P" calypso-firmware "$V" all
fi
echo "termine : $OUT"
```

3.268 /opt/GSM/osmo-operator/packaging/build-pmos-kernel-deb.sh

6831 octets, 122 lignes → 122 lignes

```
#!/bin/bash
# ============================================================================
# build-pmos-kernel-deb.sh - LE NOYAU postmarketOS AVEC PPP, EN .deb.
#
# [2026-09-08] Le noyau du telephone (linux-postmarketos-stable, rebati avec
# CONFIG_PPP pour la data 4G du modem du banc) est un .apk signe par la cle
# du build pmbootstrap. Il se construit en 15 a 40 minutes, sous un compte
# NON root, avec des montages de boucle : impossible dans un `docker build`.
# Ce script prend donc le noyau DEJA CONSTRUIT (le dossier de travail
# pmbootstrap de la machine de reference) et en fait un paquet Debian :
#
#     osmo-build-pmos-kernel_<version>+ppp~<suite>_<arch>.deb
#
# qui pose, sous /opt/user_interface/kernel/pmos/ :
#     linux-postmarketos-stable-<ver>.apk   le noyau (modules PPP signes dedans)
#     APKINDEX.tar.gz                       l index du depot local pmbootstrap
```

```
#     <cle>.rsa.pub                          la cle PUBLIQUE qui a signe l apk
#     pmaports.commit                        le commit pmaports d ou il vient
#     pmaports-linux-postmarketos-stable-ppp.patch   la config (CONFIG_PPP=m...)
#
# tools/osmo-pmos-build.sh sait lire ce dossier : il copie l apk et l index
# dans le depot local de pmbootstrap, la cle dans config_apk_keys, et
# « pmbootstrap install » prend ce noyau-la sans rien recompiler.
#
# Le .deb va dans /opt/user_interface/kernel/ (c est la qu on le range) ET
# dans /var/cache/osmo-debs/ : iso_modules/50-injection-image.sh y prend tous
# les osmo-build-*.deb de la suite, le noyau part donc dans l ISO tout seul.
#
#   sudo packaging/build-pmos-kernel-deb.sh [dossier de travail pmbootstrap]
#       defaut : le dossier « work » du pmbootstrap_v3.cfg du compte de session
#                (SUDO_USER), sinon ~<user>/test, sinon $OSMO_PMB_WORK
# ============================================================================
set -euo pipefail

REPO="$(cd "$(dirname "$0")/.." && pwd)"
UI="${OSMO_UI_ROOT:-/opt/user_interface}"
KDIR="$UI/kernel"
CACHE="${OSMO_DEB_CACHE:-/var/cache/osmo-debs}"
ARCH="$(dpkg --print-architecture 2>/dev/null || echo amd64)"
SUITE="$(. /etc/os-release 2>/dev/null; echo "${VERSION_CODENAME:-noble}")"

RED='\033[0;31m'; GREEN='\033[0;32m'; YELLOW='\033[1;33m'; CYAN='\033[0;36m'; NC='\033[0m'
say() { echo -e "${CYAN}→${NC} $*"; }; ok() { echo -e "${GREEN}✓${NC} $*"; }
die() { echo -e "${RED}✗${NC} $*" >&2; exit 1; }

# Le dossier de travail pmbootstrap : argument, sinon celui du compte de session.
_user="${SUDO_USER:-}"
[ -z "$_user" ] && [ -n "${PKEXEC_UID:-}" ] && _user="$(getent passwd "$PKEXEC_UID" | cut -d: -f1)"
_home="$(getent passwd "${_user:-root}" | cut -d: -f6)"
WORK="${1:-${OSMO_PMB_WORK:-}}"
if [ -z "$WORK" ]; then
    for c in "$_home/.config/pmbootstrap_v3.cfg" "$_home/.config/pmbootstrap.cfg"; do
        [ -f "$c" ] && WORK="$(sed -n 's/^work *= *//p' "$c" | head -1)" && [ -n "$WORK" ] && break
    done
fi
[ -n "$WORK" ] || WORK="$_home/test"
[ -d "$WORK/packages" ] || die "pas de dossier packages/ dans $WORK (dossier de travail pmbootstrap ?)"

apk="$(find "$WORK/packages" -name 'linux-postmarketos-stable-[0-9]*.apk' -printf '%T@ %p\n' 2>/dev/null | sort
 ↪  -n | tail -1 | cut -d' ' -f2-)"
[ -n "$apk" ] || die "aucun linux-postmarketos-stable-*.apk dans $WORK/packages - le noyau n a pas ete construit
 ↪  (osmo-pmos-build)"
# (grep -q ferme le tube avant la fin de tar : sous pipefail ce serait un
# faux echec - d ou la substitution de processus.)
grep -q 'ppp_async\.ko' < <(tar tzf "$apk" 2>/dev/null) || die "$apk n a PAS ppp_async.ko : ce n est pas le noyau
 ↪  PPP"
ver="$(basename "$apk" .apk | sed 's/^linux-postmarketos-stable-//')"          # 7.2.3-r0
dver="$(echo "$ver" | sed 's/-r/./')+ppp"                                      # 7.2.3.0+ppp
index="$(dirname "$apk")/APKINDEX.tar.gz"
channel="$(basename "$(dirname "$(dirname "$apk")")")"                         # edge
parch="$(basename "$(dirname "$apk")")"                                        # x86_64
pub="$(ls -1 "$WORK"/config_abuild/*.rsa.pub 2>/dev/null | head -1)"
[ -n "$pub" ] || pub="$(ls -1 "$WORK"/config_apk_keys/pmos@*.rsa.pub 2>/dev/null | head -1)"
[ -n "$pub" ] || die "pas de cle publique pmos@*.rsa.pub dans $WORK/config_abuild ni config_apk_keys"
aports="$(sed -n 's/^aports *= *//p' "$_home/.config/pmbootstrap_v3.cfg" 2>/dev/null | head -1)"
[ -n "$aports" ] && [ -d "$aports/.git" ] || aports="$WORK/cache_git/pmaports"
commit="$(git -C "$aports" rev-parse HEAD 2>/dev/null || echo inconnu)"

say "noyau : $apk ($ver, $channel/$parch), cle $(basename "$pub"), pmaports $commit"

stage="$(mktemp -d /var/tmp/pmos-kernel-deb.XXXXXX)"; trap 'rm -rf "$stage"' EXIT
d="$stage$KDIR/pmos"
```

```
mkdir -p "$d" "$stage/DEBIAN"
cp -f "$apk" "$d/"
[ -f "$index" ] && cp -f "$index" "$d/"
cp -f "$pub" "$d/"
echo "$commit" > "$d/pmaports.commit"
echo "$channel $parch $ver" > "$d/apk.info"
if [ -f "$REPO/patches/pmaports-linux-postmarketos-stable-ppp.patch" ]; then
    cp -f "$REPO/patches/pmaports-linux-postmarketos-stable-ppp.patch" "$d/"
elif git -C "$aports" diff --quiet 2>/dev/null; then :; else
    git -C "$aports" diff > "$d/pmaports-linux-postmarketos-stable-ppp.patch"
fi
cat > "$d/README" <<EOF
Noyau postmarketOS $ver (x86_64, qemu-amd64) rebati avec PPP pour la data 4G
du telephone du banc. Produit par packaging/build-pmos-kernel-deb.sh depuis
$WORK le $(date -I).
  - l apk est signe par $(basename "$pub") : la cle doit etre dans
    config_apk_keys/ du dossier de travail pmbootstrap qui l installe ;
  - pmaports doit etre au commit $commit (meme pkgver/pkgrel : $ver),
    avec le patch ci-joint applique, pour que pmbootstrap le prefere au depot ;
  - tools/osmo-pmos-build.sh (osmo-pmos-build) fait tout cela.
EOF
size="$(du -sk --exclude=DEBIAN "$stage" | cut -f1)"
cat > "$stage/DEBIAN/control" <<CTL
Package: osmo-build-pmos-kernel
Version: ${dver}~${SUITE}
Architecture: ${ARCH}
Maintainer: osmo-operator <bastienbaranoff@gmail.com>
Installed-Size: ${size}
Section: misc
Priority: optional
Description: osmo-operator - noyau postmarketOS ${ver} avec PPP (telephone du banc)
 L apk linux-postmarketos-stable rebati avec CONFIG_PPP pour la data 4G du
 modem du banc, son index, sa cle publique et le patch pmaports, sous
 ${KDIR}/pmos. Lu par osmo-pmos-build (tools/osmo-pmos-build.sh).
CTL
mkdir -p "$KDIR" "$CACHE"
deb="$KDIR/osmo-build-pmos-kernel_${dver}~${SUITE}_${ARCH}.deb"
dpkg-deb --build --root-owner-group -Zzstd "$stage" "$deb" >/dev/null 2>&1 \
    || dpkg-deb --build --root-owner-group -Zxz "$stage" "$deb" >/dev/null
cp -f "$deb" "$CACHE/"
ok "$deb ($(du -h "$deb" | cut -f1)) - copie dans $CACHE"
# Et pose sur cette machine : /opt/user_interface/kernel/pmos rempli.
dpkg -i --force-overwrite "$deb" >/dev/null 2>&1 && ok "pose : $KDIR/pmos/" || echo "  (dpkg -i a echoue : dpkg -i
 ↪  $deb)"
```

3.269 /opt/GSM/osmo-operator/packaging/osmo-deb.sh

7908 octets, 176 lignes → 176 lignes

```
#!/bin/bash
# ============================================================================
#  osmo-deb - un .deb par dossier compile, garde en cache, reinstalle au rebuild
# ============================================================================
#
#  PRINCIPE. Tout ce qui se compile dans l image (libosmocore, osmo-*, gapk,
#  QEMU, osmocom-bb, gr-gsm...) sort sous forme de paquet .deb, range dans
#      /var/cache/osmo-debs/osmo-build-<nom>_<version>~<suite>_<arch>.deb
#  Au rebuild suivant, si le paquet est la, on l installe (dpkg -i) au lieu de
#  recloner et recompiler. Le cache vit SUR L HOTE (/var/cache/osmo-debs) :
#  build.sh le recopie dans le contexte (.deb-cache/) avant `docker build`, et
#  ramene les paquets produits par l image une fois le build fini.
#
#  Trois verbes :
#
```

```
#    osmo-deb install <nom> <version>
#        Installe le .deb du cache s il existe. Sort en 0 dans ce cas, en 1
#        sinon (= il faut compiler). Avec OSMO_DEB_REFRESH=1, sort toujours
#        en 1 : le cache est ignore et sera reecrit (build.sh --no-cache).
#
#    osmo-deb pack <nom> <version> <commande...>
#        Lance <commande> avec DESTDIR=<staging> dans l environnement -
#        `make install`, `ninja install`, `cmake --install` l honorent tous -
#        fabrique le .deb depuis le staging, l installe dans la racine et le
#        range dans le cache. Usage type :
#            ./configure && make -j && osmo-deb pack libosmocore 1.12.1 make install
#        L ARBRE DE SOURCES PART AVEC : le dossier courant, quand il est sous
#        /opt/GSM, est ajoute tel quel au paquet (.git, objets compiles). Au
#        rebuild depuis le cache, /opt/GSM/<nom> existe donc toujours - l image
#        et l ISO restent un atelier ou l on peut relire, patcher et refaire
#        `make`. Sans cela, un cache plein donnait une image SANS aucune source.
#        OSMO_DEB_SRC=/chemin designe un autre arbre ; OSMO_DEB_NO_SRC=1 le retire.
#
#    osmo-deb snapshot <nom> <version> <chemin...>
#        Pour ce qui n a pas d etape d installation (un arbre de sources qui
#        sert tel quel au runtime : osmocom-bb, qosmo-grgsm, un venv) : le .deb
#        est fabrique depuis des fichiers DEJA en place. Rien n est reinstalle.
#
#    osmo-deb list
#
#  La <version> est un choix de l appelant : la version amont quand elle est
#  epinglee (1.12.1), `0.git` quand on suit une branche. Le nom de la suite
#  Ubuntu (noble, jammy) est ajoute automatiquement : un paquet compile sur
#  jammy ne sera jamais pose sur un rootfs noble.
#
#  Variables : OSMO_DEB_CACHE (defaut /var/cache/osmo-debs)
#              OSMO_DEB_REFRESH=1  ignore le cache, reconstruit et remplace
# ============================================================================
set -euo pipefail

CACHE="${OSMO_DEB_CACHE:-/var/cache/osmo-debs}"
REFRESH="${OSMO_DEB_REFRESH:-0}"
PREFIX="osmo-build-"
SRC_ROOT="${OSMO_DEB_SRC_ROOT:-/opt/GSM}"   # les arbres de sources vivent la
ARCH="$(dpkg --print-architecture 2>/dev/null || echo amd64)"
SUITE="$(. /etc/os-release 2>/dev/null; echo "${VERSION_CODENAME:-unknown}")"
MAINT="osmo-operator <bastienbaranoff@gmail.com>"

log() { echo "[osmo-deb] $*" >&2; }
die() { log "ERREUR : $*"; exit 1; }

usage() { sed -n '2,45p' "$0" | sed 's/^#\{1,2\} \{0,1\}//'; exit "${1:-0}"; }

check_name() {
    [[ "$1" =~ ^[a-z0-9][a-z0-9.+-]*$ ]] || die "nom de paquet invalide : '$1' (minuscules, chiffres, . + -)"
    [[ "$2" =~ ^[0-9][A-Za-z0-9.+~-]*$ ]] || die "version invalide : '$2' (doit commencer par un chiffre)"
}

deb_path() { printf '%s/%s%s_%s~%s_%s.deb' "$CACHE" "$PREFIX" "$1" "$2" "$SUITE" "$ARCH"; }

# Fabrique le .deb depuis un staging ($1) pour <nom> ($2) <version> ($3).
make_deb() {
    local stage="$1" name="$2" ver="$3" deb size
    deb="$(deb_path "$name" "$ver")"
    mkdir -p "$CACHE" "$stage/DEBIAN"
    size="$(du -sk --exclude=DEBIAN "$stage" | cut -f1)"
    cat > "$stage/DEBIAN/control" <<CTL
Package: ${PREFIX}${name}
Version: ${ver}~${SUITE}
Architecture: ${ARCH}
Maintainer: ${MAINT}
```

```
Installed-Size: ${size}
Section: misc
Priority: optional
Description: osmo-operator - ${name} ${ver}, compile dans l image (${SUITE})
 Paquet produit par osmo-deb au build de l image osmocom-nitb. Il ne declare
 aucune dependance : les bibliotheques systeme viennent de la liste apt du
 Dockerfile, qui fait autorite.
CTL
    # zstd quand dpkg le sait (jammy et noble), xz sinon : sur un arbre QEMU de
    # 1,5 Go la difference est de plusieurs minutes.
    rm -f "$deb.tmp"
    if ! dpkg-deb --build --root-owner-group -Zzstd "$stage" "$deb.tmp" >/dev/null 2>&1; then
        dpkg-deb --build --root-owner-group -Zxz "$stage" "$deb.tmp" >/dev/null
    fi
    mv -f "$deb.tmp" "$deb"
    log "$name $ver -> $deb ($(du -h "$deb" | cut -f1))"
}

cmd_install() {
    [ $# -eq 2 ] || usage 2
    check_name "$1" "$2"
    local deb; deb="$(deb_path "$1" "$2")"
    if [ "$REFRESH" = "1" ]; then
        log "$1 $2 : cache ignore (OSMO_DEB_REFRESH=1) -> compilation"; return 1
    fi
    if [ ! -s "$deb" ]; then
        log "$1 $2 : pas en cache ($(basename "$deb")) -> compilation"; return 1
    fi
    log "$1 $2 : installe depuis le cache ($(basename "$deb"))"
    dpkg -i --force-overwrite "$deb" >/dev/null
    ldconfig 2>/dev/null || true
    return 0
}

cmd_pack() {
    [ $# -ge 3 ] || usage 2
    local name="$1" ver="$2"; shift 2
    check_name "$name" "$ver"
    local stage; stage="$(mktemp -d "${TMPDIR:-/tmp}/osmo-deb.XXXXXX")"
    DESTDIR="$stage" "$@"
    if [ -z "$(find "$stage" -mindepth 1 -maxdepth 1 ! -name DEBIAN 2>/dev/null)" ]; then
        rm -rf "$stage"
        die "$name : rien n a ete installe dans DESTDIR - la commande l ignore-t-elle ?"
    fi
    # L arbre de sources (le dossier courant, sous /opt/GSM) part dans le
    # paquet : les sources restent dans /opt/GSM meme quand le cache dispense
    # de cloner et de compiler. Voir l en-tete.
    local src="${OSMO_DEB_SRC:-$PWD}"
    if [ "${OSMO_DEB_NO_SRC:-0}" != "1" ] && [ -d "$src" ] \
       && case "$src" in "$SRC_ROOT"/?*) true ;; *) false ;; esac; then
        mkdir -p "$stage$(dirname "$src")"
        # [2026-09-09] `cp -an`, et plus de `[ ! -e "$stage$src" ]` : le prefixe
        # d installation peut vivre SOUS l arbre de sources (Open5GS :
        # /opt/LTE/open5gs/install), auquel cas il est DEJA dans le staging.
        # L ancien test sautait alors tout l arbre, et osmo-build-open5gs
        # sortait sans une seule source (13 Mo : bin/ et lib/, rien d autre) -
        # sur l ISO, /opt/LTE/open5gs n avait que install/, et un --build y
        # reclonait. On complete autour de ce qui est deja la (-n : jamais
        # par-dessus ce que la commande d installation vient de poser).
        cp -an "$src" "$stage$(dirname "$src")/"
        log "$name : arbre de sources $src inclus dans le paquet"
    fi
    make_deb "$stage" "$name" "$ver"
    dpkg -i --force-overwrite "$(deb_path "$name" "$ver")" >/dev/null
    ldconfig 2>/dev/null || true
    rm -rf "$stage"
```

```
}

cmd_snapshot() {
    [ $# -ge 3 ] || usage 2
    local name="$1" ver="$2"; shift 2
    check_name "$name" "$ver"
    local stage p; stage="$(mktemp -d "${TMPDIR:-/tmp}/osmo-deb.XXXXXX")"
    for p in "$@"; do
        [ -e "$p" ] || { rm -rf "$stage"; die "$name : $p n existe pas"; }
        case "$p" in /*) ;; *) rm -rf "$stage"; die "$name : chemin absolu requis ($p)" ;; esac
        mkdir -p "$stage$(dirname "$p")"
        cp -a "$p" "$stage$(dirname "$p")/"
    done
    make_deb "$stage" "$name" "$ver"
    rm -rf "$stage"
}

cmd_list() {
    ls -1sh "$CACHE"/${PREFIX}*.deb 2>/dev/null || log "cache vide ($CACHE)"
}

case "${1:-}" in
    install)  shift; cmd_install "$@" ;;
    pack)     shift; cmd_pack "$@" ;;
    snapshot) shift; cmd_snapshot "$@" ;;
    list)     cmd_list ;;
    -h|--help|help|'') usage 0 ;;
    *) die "verbe inconnu : $1 (install|pack|snapshot|list)" ;;
esac
```

3.270 /opt/GSM/osmo-operator/packaging/snapshot-lte-debs.sh

7218 octets, 113 lignes → 113 lignes

```
#!/bin/bash
# ============================================================================
# snapshot-lte-debs.sh - LA 4G ET LE TELEPHONE DU BANC, EN .deb, DEPUIS LE
# NATIF - le raccourci.
#
# [2026-09-08] Deux chemins menent aux memes paquets :
#   1. LE RACCOURCI (ce script) : la machine de reference a DEJA srsRAN (ZeroMQ)
#      compile dans /opt/LTE/srsRAN_4G, libzmq/czmq dans /opt/LTE, Open5GS dans
#      /opt/LTE/open5gs (prefixe install/), et le noyau postmarketOS PPP dans
#      le dossier de travail pmbootstrap du compte de session. On photographie
#      tout cela en paquets, tels quels (osmo-deb snapshot : rien n est
#      recompile, rien n est reinstalle) :
#          osmo-build-libzmq_4.3.5+git~noble_amd64.deb
#          osmo-build-srsgui_0.1+git~noble_amd64.deb        (avant srsran : il s y lie)
#          osmo-build-srsran_25.10+zmq+x86-64-v2~noble_amd64.deb  (ISA comprise)
#          osmo-build-open5gs_2.8.0+git~noble_amd64.deb
#          osmo-build-pmos-kernel_7.2.3.0+ppp~noble_amd64.deb   (packaging/build-pmos-kernel-deb.sh)
#      dans /var/cache/osmo-debs : le build ISO (50-injection-image.sh) les
#      pose dans le rootfs, et le Dockerfile (osmo-deb install) les prend au
#      lieu de compiler.
#   2. LES SOURCES : le Dockerfile et tools/osmo-lte-install.sh --build
#      compilent srsRAN et Open5GS quand le cache n a pas ces paquets. Le
#      noyau pmOS, lui, ne se construit que par osmo-pmos-build (pmbootstrap,
#      sous le compte de session) : il n a PAS de chemin docker.
#
#   sudo packaging/snapshot-lte-debs.sh            tout ce qui est la
#   sudo packaging/snapshot-lte-debs.sh --no-pmos  sans le noyau du telephone
# ============================================================================
set -u
REPO="$(cd "$(dirname "$0")/.." && pwd)"
```

```
LTE="${OSMO_LTE_ROOT:-/opt/LTE}"
CACHE="${OSMO_DEB_CACHE:-/var/cache/osmo-debs}"
OSMODEB="$REPO/packaging/osmo-deb.sh"
PMOS=1; [ "${1:-}" = "--no-pmos" ] && PMOS=0
GREEN='\033[0;32m'; YELLOW='\033[1;33m'; CYAN='\033[0;36m'; NC='\033[0m'
say() { echo -e "${CYAN}→${NC} $*"; }; ok() { echo -e "${GREEN}✓${NC} $*"; }; warn() { echo -e "${YELLOW}!${NC}
 ↪  $*"; }
[ "$(id -u)" -eq 0 ] || { echo "root requis" >&2; exit 1; }
export OSMO_DEB_CACHE="$CACHE"
mkdir -p "$CACHE"

# ── libzmq + czmq (compiles a la main dans /opt/LTE, installes en /usr/local) ─
if [ -e /usr/local/lib/libzmq.so.5 ]; then
    say "libzmq / czmq"
    _z=( /usr/local/lib/libzmq.so* /usr/local/lib/libzmq.la /usr/local/lib/pkgconfig/libzmq.pc
     ↪  /usr/local/include/zmq.h /usr/local/include/zmq_utils.h )
    [ -e /usr/local/lib/libczmq.so ] && _z+=( /usr/local/lib/libczmq.so* /usr/local/lib/libczmq.la
     ↪  /usr/local/lib/pkgconfig/libczmq.pc /usr/local/include/czmq.h /usr/local/include/czmq_library.h
     ↪  /usr/local/include/czmq_prelude.h )
    [ -x /usr/local/bin/zmakecert ] && _z+=( /usr/local/bin/zmakecert )
    for d in "$LTE/libzmq" "$LTE/czmq"; do [ -d "$d" ] && _z+=( "$d" ); done
    _zv="$(git -C "$LTE/libzmq" describe --tags 2>/dev/null | sed 's/^v//; s/-.*//')"
    bash "$OSMODEB" snapshot libzmq "${_zv:-4.3.5}+git" "${_z[@]}" && ok "libzmq"
else
    warn "pas de /usr/local/lib/libzmq.so.5 : libzmq d Ubuntu (libzmq5) suffira"
fi

# ── srsGUI : les traces temps reel, AVANT srsRAN (qui s y lie) ────────────────
# [2026-09-10] srsRAN_4G est desormais compile avec ENABLE_GUI=ON : srsenb et
# srsue se LIENT a libsrsgui. Un .deb de srsRAN sans celui-la donnerait des
# binaires qui ne demarrent pas (« libsrsgui.so: cannot open shared object
# file »). Il passe donc avant, dans ce fichier comme dans le cache.
if [ -e /usr/local/lib/libsrsgui.so ] || [ -e /usr/local/lib/libsrsgui.a ]; then
    say "srsGUI (traces srsenb / srsue)"
    _g=( /usr/local/lib/libsrsgui.* )
    [ -d /usr/local/include/srsgui ] && _g+=( /usr/local/include/srsgui )
    [ -e /usr/local/lib/pkgconfig/srsgui.pc ] && _g+=( /usr/local/lib/pkgconfig/srsgui.pc )
    [ -d "$LTE/srsGUI" ] && _g+=( "$LTE/srsGUI" )
    bash "$OSMODEB" snapshot srsgui "${OSMO_SRSGUI_VER:-0.1+git}" "${_g[@]}" && ok "srsgui"
else
    warn "pas de /usr/local/lib/libsrsgui : srsGUI non empaquete (srsRAN sera sans traces)"
fi

# ── srsRAN_4G : l arbre (sources + build) et ce que make install a pose ──────
if [ -f "$LTE/srsRAN_4G/build/install_manifest.txt" ]; then
    say "srsRAN_4G (ZeroMQ)"
    mapfile -t _s < <(grep -v '^$' "$LTE/srsRAN_4G/build/install_manifest.txt" | while read -r f; do [ -e "$f" ]
     ↪  && echo "$f"; done)
    _sv="$(git -C "$LTE/srsRAN_4G" describe --tags 2>/dev/null | sed 's/^release_//; s/-.*//; s/_/./g')"
    # ── L ISA VA DANS LA VERSION, ET ELLE EST CONSTATEE, PAS SUPPOSEE ─────────
    # [2026-09-14] Ce script PHOTOGRAPHIE des binaires deja compiles : rien ne
    # dit ici avec quel -march ils l ont ete. Tant que le .deb s appelait
    # « 25.10+zmq », un srsenb natif de la machine de reference satisfaisait le
    # « osmo-deb install srsran » du Dockerfile, partait dans l image et dans
    # l ISO, et faisait « Illegal instruction » chez l operateur (voir
    # SRS_ARCH dans tools/osmo-lte-install.sh). On regarde donc ce que le
    # binaire CONTIENT - une instruction %ymm/%zmm et il est natif - et on le
    # dit dans son nom. Sans objdump on ne tranche pas : +inconnu, qui ne sera
    # jamais confondu avec le portable non plus.
    _sisa=inconnu
    if command -v objdump >/dev/null 2>&1 && [ -x /usr/local/bin/srsenb ]; then
        if objdump -d /usr/local/bin/srsenb 2>/dev/null | grep -qE '%[yz]mm[0-9]'; then _sisa=native
        else _sisa="$(printf '%s' "${OSMO_SRSRAN_ARCH:-x86-64-v2}" | tr -c 'A-Za-z0-9.' '-')"; fi
    fi
    [ "$_sisa" = native ] && warn "srsenb porte de l AVX : paquete en +native (il ne tournera que sur CETTE
     ↪  machine)"
```

```
    bash "$OSMODEB" snapshot srsran "${_sv:-25.10}+zmq+${_sisa}" "$LTE/srsRAN_4G" "${_s[@]}" && ok "srsran
     ↪  ${_sv:-25.10}+zmq+${_sisa} (${#_s[@]} fichiers installes + l arbre)"
else
    warn "pas de $LTE/srsRAN_4G/build/install_manifest.txt : srsRAN non empaquete"
fi

# ── Open5GS : l arbre entier, prefixe install/ compris ──────────────────
if [ -x "$LTE/open5gs/install/bin/open5gs-mmed" ]; then
    say "Open5GS ($LTE/open5gs)"
    _ov="$(git -C "$LTE/open5gs" describe --tags 2>/dev/null | sed 's/^v//; s/-.*//')"
    # Les journaux et le .ctxt ne voyagent pas.
    find "$LTE/open5gs/install/var/log" -type f -delete 2>/dev/null || true
    rm -f "$LTE/open5gs/.ctxt"
    bash "$OSMODEB" snapshot open5gs "${_ov:-2.8.0}+git" "$LTE/open5gs" && ok "open5gs"
else
    warn "pas de $LTE/open5gs/install/bin/open5gs-mmed : Open5GS non empaquete (il vit encore dans /root ?)"
fi

# ── Le noyau postmarketOS PPP ──────────────────────────────────────────
if [ "$PMOS" = 1 ]; then
    say "noyau postmarketOS (PPP)"
    bash "$REPO/packaging/build-pmos-kernel-deb.sh" "${OSMO_PMB_WORK:-}" || warn "noyau pmOS non empaquete (voir
     ↪  ci-dessus)"
fi
echo; bash "$OSMODEB" list 2>/dev/null | grep -E 'libzmq|srsgui|srsran|open5gs|pmos-kernel' || true
```

3.271 /opt/GSM/osmo-operator/patches/c54x_exe-apm-par-fenetre.patch

6036 octets, 86 lignes → 86 lignes

```
diff --git a/src/pont.c b/src/pont.c
index f4a3080..ab4cc3f 100644
--- a/src/pont.c
+++ b/src/pont.c
@@ -853,13 +868,80 @@ static void tx_tsp_publier(const C54xState *dsp, uint32_t fn)
     static int nlog;
     if (nlog++ < 6 || (b.type == TSP_TX_RACH && n_rach <= 3))
         printf("  [rom-tx] #%u tick=%u fn=%u (%s) %s%s offset=%d @%04x : %s...\n", w_compte, fn, fn_gsm,
-               src == 1 ? "a_a5fn" : src == 2 ? "tick+dec" : "TICK NU",
+               src == 1 ? "a_a5fn" : src == 2 ? "tick+dec" : src == 3 ? "calage BSP" : "TICK NU",
                b.type == TSP_TX_RACH ? "RACH" : b.type == TSP_TX_NB ? "NB" : b.type == TSP_TX_SB ? "SB" : b.type
 ↪  == TSP_TX_DUMMY ? "DUMMY" : b.type == TSP_TX_FB ? "FB" : "?", b.type == TSP_TX_NB ? (char[]){' ', 'T', 'S',
 ↪  'C', (char)('0' + b.tsc), 0} : "",
              b.offset, b.adresse, (char[]){ '0' + b.bits[0], '0' + b.bits[1], '0' + b.bits[2], '0' + b.bits[3],
 ↪  '0' + b.bits[4], '0' + b.bits[5], '0' + b.bits[6], '0' + b.bits[7], 0 });
     if ((n_rach + n_nb + n_sb + n_dummy + n_fb + n_inconnu) % 2000 == 0)
         printf("  [rom-tx] %u bursts publies : RACH %u, NB %u, SB %u, DUMMY %u, FB %u, inconnus %u\n", n_rach +
 ↪  n_nb + n_sb + n_dummy + n_fb + n_inconnu, n_rach, n_nb, n_sb, n_dummy, n_fb, n_inconnu);
 }

+/* [2026-10-21 19:00] RACH VERS L'AIR (CALYPSO_RACH_VERS_AIR=1, Magnetite). La capture au script TSP de la
+ * ROM (0x3cbb/0x3ccd, chemin du firmware osmocom) ne voit pas le burst d'acces du firmware TI : montant.c
+ * detectait bien les RACH (d_task_ra, ra/bsic) mais ils n'allaient qu'a /dev/shm/calypso_rach, que l'air
+ * ne lit pas -> B->A 0, pas de CHAN RQD, « Network forbidden ». On code le burst d'acces (05.03,
+ * gsm0503_rach_ext_encode) et on le publie dans calypso_tx_rom, date de la trame BTS du calage. */
+/* [2026-10-21 19:40] a_pm JAMAIS NUL (CALYPSO_BSP_BRUIT_FOND=1, Magnetite). La L1 TI lit pm = a_pm >> 5
+ * (l1ddsp_meas_read) et compte chaque pm nul comme une panne (l1_check_pm_error : 100 -> recovery, DSP
+ * relance). Le modele de puissance du banc (calypso_bsp_rssi_apm) ne passe que sur les ecritures du DSP ;
+ * un a_pm laisse a 0 par la ROM partait tel quel. Avant de rendre la trame a l'ARM, chaque mot a_pm nul
+ * des deux pages R recoit le modele (plancher 64). */
+static void apm_jamais_nul(uint16_t *api_ram)
+{
+    static int on = -1;
+    if (on < 0) { const char *e = calypso_getenv("CALYPSO_BSP_BRUIT_FOND"); on = (e && *e == '1'); }
```

```
+    if (!on) return;
+    for (int pg = 0; pg < 2; pg++) {
+        uint16_t *a = &api_ram[(API_R_PAGE(pg) + RP_A_PM) / 2];
+        for (int k = 0; k < 3; k++)
+            if ((a[k] >> 5) == 0) a[k] = calypso_bsp_rssi_apm_k(k);   /* [2026-10-21] une mesure par fenetre */
+    }
+    /* [2026-10-21 20:00] MESURE PLEINE LISTE (RXL_R 548) : la L1 TI (DSP 36) programme d_task_md = 0x200 + n
+     * (n <= 8 mesures, l1_sync.c) et lit les n PM dans les mots 12..12+n-1 de la page R (l1ddsp_meas_read,
+     * nbmeas = 8 vu au gdb) : au-dela des 3 mots a_pm, ce sont les mots de a_sch. Le modele de puissance du
+     * banc ne couvrait que les mots 12..14 : 15..19 restaient a 0 (vu au moniteur, d_task_md 0x0208), 5 PM
+     * nuls sur 8 a chaque trame du balayage -> recovery. La page R reprend d_task_md (mot 4) : on s'en sert. */
+    /* [2026-10-21 21:50] MODE DEDIE : les taches SDCCH/SACCH (ADL 22, DDL 24 cote L1 TI) lisent la puissance dans
+     * a_serv_demod[D_PM], mot 9 de la page R pour le DSP 36 (l1_defty.h : a_serv_demod 8..11). Le DSP emule l'y
+     * laissait a 0 apres l'IMM ASS : « > PM 1778 22 », « > PM 1808 24 » a chaque trame, ABORT, retour en veille.
+     * Meme plancher que a_pm, des qu'une tache descendante (d_task_d, mot 0) est rapportee. */
+    for (int pg = 0; pg < 2; pg++) {
+        uint16_t *r = &api_ram[API_R_PAGE(pg) / 2];
+        if ((r[0] & 0x7fff) && (r[9] >> 5) == 0) r[9] = calypso_bsp_rssi_apm();
+    }
+    for (int pg = 0; pg < 2; pg++) {
+        uint16_t *r = &api_ram[API_R_PAGE(pg) / 2];
+        uint16_t md = r[4];
+        if (!(md & 0x200)) continue;
+        int n = md & 0xff; if (n > 8) n = 8;
+        for (int k = 0; k < n; k++)
+            if ((r[12 + k] >> 5) == 0) r[12 + k] = calypso_bsp_rssi_apm_k(k);
+    }
+}
+
+void tx_rach_vers_air(uint8_t ra, uint8_t bsic, uint32_t tick)
+{
+    static int actif = -1;
+    if (actif < 0) { const char *e = calypso_getenv("CALYPSO_RACH_VERS_AIR"); actif = (e && *e == '1'); }
+    if (!actif) return;
+    /* [2026-10-21 20:55] le BURST D'ACCES ENTIER (05.02 5.2.7), comme le chemin ROM (tsp_tx.c, offset 8) :
+     * 8 bits de queue etendue, 41 de synchro, les 36 codes, 3 de queue. gsm0503_rach_ext_encode ne rend que
+     * les 36 bits codes : publies seuls en tete, le TRX n'avait aucune synchro a correler (transceiver :
+     * aucun « TOA : », osmo-bts rach:rcvd 0). */
+    static const char queue[] = "00111010", sync[] = "01001011011111111100110011010101000111100";
+    uint8_t code[36], bits[148] = {0};
+    if (gsm0503_rach_ext_encode(code, ra, bsic, false) < 0) return;
+    for (int i = 0; i < 8; i++) bits[i] = (uint8_t)(queue[i] - '0');
+    for (int i = 0; i < 41; i++) bits[8 + i] = (uint8_t)(sync[i] - '0');
+    memcpy(bits + 49, code, 36);
+    int src = 0;
+    uint32_t fn_gsm = tx_tsp_fn(tick, &src);
+    if (!ring_publier(TSP_TX_RACH, bits, -1, 8, (uint8_t)src, fn_gsm)) return;
+    static unsigned n;
+    if (++n <= 20 || n % 100 == 0)
+        printf("  [rach-air] RACH ra=0x%02x bsic=%u tick=%u -> fn BTS %u (%s) publie vers l'air [%u]\n",
+               ra, bsic, tick, fn_gsm, src == 1 ? "a_a5fn" : src == 2 ? "tick+dec" : src == 3 ? "calage BSP" : src
 ↪  == 4 ? "fn du mobile (SB)" : "TICK NU", n);
+}
+
 /* [2026-10-20] MODE TRX : PUBLICATION DEPUIS LA PAGE DE L'EXTCODE. Le firmware trx remplit tx[p] (cmd,
  * pointeur de donnees) pour chaque slot p de sa fenetre TX (prim_bts.c patche : slot p = TS (tx_start + p)),
  * et l'extcode du DSP les module ; la ROM emulee ne joue qu'un burst par trame (le script TSP) et le
```

3.272 /opt/GSM/osmo-operator/patches/c54x_exe-magnetite-run-can-nb-toa.patch

32583 octets, 480 lignes → 480 lignes

```
diff --git a/run.sh b/run.sh
index 0a6e5bf..54a0d64 100755
```

```
--- a/run.sh
+++ b/run.sh
@@ -55,6 +55,14 @@ PONT="${PONT:-0}"
 # reste lisible en /run/calypso-shm-hote (l'air y lit l'anneau TX de l'instance 1). Les processus
 # lances ici heritent de l'espace ; --stop/--status relances avec INSTANCE=N y entrent aussi.
 INSTANCE="${INSTANCE:-1}"
+# [2026-10-21] REGLAGES PAR INSTANCE : instances/instance-$INSTANCE.env (ecrit par la page web
 ↪ tools/osmo-reglages.py,
+# patron instances/instance.env.example). Lignes `: "${VAR:=valeur}"` : l'environnement (start-native.sh, ligne
 ↪ de
+# commande) garde la priorite, puis le fichier, puis les defauts ci-dessous. Exporte (set -a) pour
 ↪ c54x_exe/QEMU/pont.
+INSTANCE_ENV="${INSTANCE_ENV:-/opt/GSM/osmo-operator/instances/instance-$INSTANCE.env}"
+if [ -r "$INSTANCE_ENV" ]; then
+    set -a; . "$INSTANCE_ENV"; set +a
+    printf '\033[1m[run dsp]\033[0m instance %s : reglages de %s\n' "$INSTANCE" "$INSTANCE_ENV"
+fi
 HOTE_SHM=/run/calypso-shm-hote
 if [ "$INSTANCE" != 1 ] && [ -z "${CALYPSO_NS:-}" ]; then
     command -v unshare >/dev/null || { echo "[run] ECHEC : unshare absent (util-linux)" >&2; exit 1; }
@@ -151,6 +159,10 @@ if [ "$OSMO_TRX" = 1 ]; then
     # calee sur le fn du SB emis (pont.c -> calypso_bsp_trx_calage) ; CALYPSO_BSP_TRX_DELTA ajuste (defaut -2
 ↪ avec le
     # firmware patche BTS_RX_LAG 3 : le burst montant fn X est joue une trame apres la fenetre native, marge pour
 ↪ l'air)
     export CALYPSO_BSP_TRX="${CALYPSO_BSP_TRX:-1}"
+    # [2026-10-21] firmware trx BTS_RX_LAG 12 (prim_bts.c) : le burst montant fn X est joue 11 trames apres sa
 ↪ fenetre
+    # native ; marge pour la latence aller-retour de l'air + le retard du mobile emule (RACH du Magnetite arrives
+    # 27 trames trop tard avec LAG 4 / delta -3)
+    export CALYPSO_BSP_TRX_DELTA="${CALYPSO_BSP_TRX_DELTA:--11}"
     # telephone 2 (TS1) : pas de SB emis, offset fn/tick fixe (voir OSMO_TRX_TS1)
     [ "$OSMO_TRX_PHONE" = 2 ] && export CALYPSO_BSP_TRX_OFFSET="${CALYPSO_BSP_TRX_OFFSET:--1}"
 fi
@@ -170,7 +182,46 @@ if [ -n "$FIRMWARE_FLASH" ] && { [ "$OSMO_TRX" = 1 ] || [ "$OSMO_TRX_PHONE" = 2
     printf '\033[1m[run dsp]\033[0m OSMO_TRX=1 : FIRMWARE_FLASH=%s ignore, cette instance est le TRX de la BTS
 ↪ (firmware trx)\n' "$FIRMWARE_FLASH"
     FIRMWARE_FLASH=""
 fi
+# [2026-10-21] Magnetite (vrai firmware TI en flash) : pas de rattrapage du pacer. Avec le rattrapage (20),
+# l ARM enchaine les trames en retard, la L1 sature et le chien de garde reset le firmware (gel).
 if [ -n "$FIRMWARE_FLASH" ]; then
+    export CALYPSO_PACER_RATTRAPAGE="${CALYPSO_PACER_RATTRAPAGE:-0}"
+    export CALYPSO_PONT_ARM_LOCK="${CALYPSO_PONT_ARM_LOCK:-1}"
+    # [2026-10-21] L1 TI : fenetre SB ouverte en fin de trame precedente (sinon BCCH decale d'une trame,
+    # CRC KO sur chaque bloc) et TOA NB vise a 3 (marge 1) ; voir calypso_bsp.c sb_fin_trame()
+    export CALYPSO_BSP_SB_FIN_TRAME="${CALYPSO_BSP_SB_FIN_TRAME:-1}"
+    export CALYPSO_BSP_NB_MARGE="${CALYPSO_BSP_NB_MARGE:-1}"
+    # TOA NB canne a la cible TI (pont.c). [21/10 17:05] coupe par defaut tant que c54x_exe n'est pas
+    # recompile avec la garde init_done : sans elle le cannage ecrase le code DSP pendant le boot.
+    export PONT_CAN_NB_TOA="${PONT_CAN_NB_TOA:-}"
+    # [21/10] faux air : bruit hors de l'ARFCN de la cellule (calypso_bsp.c faux_air_*), ARFCN du plan radio
+    # [21/10 19:50] SB module comme FB/NB : les reglages SB du 30/09 (instant 0.35, elargissement 1.0, bruit 3000,
+    # faits pour la layer1 osmocom qui ignore l'angle du SB) biaisaient l'angle mesure (+166 a 0 Hz) ; la L1 TI
+    # s'en sert pour l'AFC (-700 -> -673, ~224 Hz de vrai decalage) et perdait SB puis BCCH
+    export CALYPSO_BSP_SB_DEC="${CALYPSO_BSP_SB_DEC:-0.5}" CALYPSO_BSP_SB_SYM="${CALYPSO_BSP_SB_SYM:-0.3}"
 ↪ CALYPSO_BSP_SB_NOISE="${CALYPSO_BSP_SB_NOISE:-0}"
+    export CALYPSO_BSP_BRUIT_FOND="${CALYPSO_BSP_BRUIT_FOND:-1}"   # bruit au lieu de zeros (PM = 0 -> recovery
 ↪ L1)
+    # rattraper le direct du BTS (calypso_bsp.c) : PAR L'AIR SEULEMENT (AIR=1, la BTS a sa propre horloge).
 ↪ [21/10 21:37]
+    # en BTS classique pont_dsp.py cadence osmo-bts sur les trames livrees par le DSP : chaque saut de +33
 ↪ accelerait la
+    # BTS (pont a ~350 trames/s), d'ou plus de retard et plus de sauts ; RA_C 1708/1933/2165 contre RACH du pont a
+    # 1737/2059/2457, IMM ASS jamais reconnues.
+    export CALYPSO_BSP_RATTRAPE="${CALYPSO_BSP_RATTRAPE:-$AIR}"
```

```
+    export CALYPSO_RACH_VERS_AIR="${CALYPSO_RACH_VERS_AIR:-1}"   # RACH du firmware TI publie vers l'air (pont.c)
+    # [21/10 21:55] BTS classique : les bursts SDCCH/SACCH du firmware TI sont bien dans calypso_tx_rom (TSC 7,
 ↪  fn de
+    # la ROM), mais pont_dsp les jetait faute de canal dedie annonce (calypso_dcch_cfg vient du tap L1CTL de QEMU,
+    # absent sans layer1 osmocom) : « aucun canal dedie connu apres 300 ms, perdu », « DL: >N200 of SABM ».
+    # PONT_UL_TN_DEFAUT=0 : TS0 (SDCCH/4 du banc) faute d'annonce. PONT_UL_RAW_DCCH_HOTE=1 (codage hote) reste
 ↪  dispo.
+    export PONT_UL_TN_DEFAUT="${PONT_UL_TN_DEFAUT:-0}"
+    # [21/10 23:05] TCH montant par l'HOTE (FACCH, SACCH/T, parole codees par le pont depuis a_fu/a_du) : le
 ↪  script TSP
+    # de la ROM ne sort que 2 bursts « inconnus » sur le TCH du firmware TI, le SABM FACCH n'atteignait jamais la
 ↪  BTS
+    # (BSC : rll_ready=no, ASSIGNMENT FAILURE, « >N200 of SABM »). 14-demarrage.sh exporte PONT_UL_RAW_TCH=1 pour
 ↪  le
+    # vocodeur ROM du mobile osmocom : on force 0 ici (PONT_UL_RAW_TCH_MAGNETITE pour revenir dessus).
+    export PONT_UL_RAW_TCH="${PONT_UL_RAW_TCH_MAGNETITE:-0}"
+    # [21/10 21:35] PAS de correction de fn ni de RA (AIR_B_FN_DELTA, MONTANT_RACH_DIFFERE : 0). Les ecarts vus a
 ↪  21:05
+    # et 21:25 comparaient au RA_R de la L1 TI, qui trace la DEMANDE (MPHC_RA_REQ) ; le RACH part apres le tirage
+    # aleatoire, juste avant RA_C (RA_R 9394 ra 12 -> RA_C 9403 ; c54x_exe publie ra 12 a 9403) : fn et RA
 ↪  etaient justes.
+    export AIR_B_FN_DELTA="${AIR_B_FN_DELTA:-0}"
+    export MONTANT_RACH_DIFFERE="${MONTANT_RACH_DIFFERE:-0}"
+    export CALYPSO_FAUX_AIR="${CALYPSO_FAUX_AIR:-$( . /etc/osmocom/radio-plan.env 2>/dev/null; echo
 ↪  "${PLAN_ARFCN:-}")}"
     [ -r "$FIRMWARE_FLASH" ] || { printf '\033[1;31m[run] ECHEC :\033[0m image flash absente : %s\n'
 ↪  "$FIRMWARE_FLASH" >&2; exit 1; }
     QEMU_CPU="${QEMU_CPU:-ti925t}"
    export CALYPSO_LCD="${CALYPSO_LCD:-c139}"
@@ -226,7 +277,9 @@ GDB="${GDB:-1}"                                # gdbstub QEMU + console telnet (et
 GDB_STUB="${GDB_STUB:-1234}"
 GDB_TELNET="${GDB_TELNET:-44444}"
 GDB_TELNET_PY="${GDB_TELNET_PY:-$QOSMO/tools/gdb-telnet.py}"   # [2026-10-03] etait /opt/GSM/qosmo-dsp/...
 ↪  (depot disparu)
-INSNS="${INSNS:-16000}"   # [2026-09-23] 60000 debordait en TCH (jusqu a 87000 insn/trame), voir start-native.sh
+# [2026-10-21] 120000 par defaut : c est le budget du DSP du TRX (instance 1, boote au premier IDLE) ;
+# 16000 laissait le DSP de l instance Magnetite en « occupe boot » pour toujours.
+INSNS="${INSNS:-120000}"   # [2026-09-23] 60000 debordait en TCH (jusqu a 87000 insn/trame), voir start-native.sh
 # [2026-09-20] Pas-a-pas DSP/QEMU par defaut (LOCKSTEP=0 pour le mode horloge murale) :
 # le C54x emule coute ~6,7 ms par trame contre 4,615 ms de temps reel, QEMU sautait
 # donc 3 trames sur 4 (« DSP en retard, tick saute »), la ROM ne voyait qu'une trame
@@ -539,6 +592,9 @@ etape4() {   # le mobile
         # l'indentation : « $3 = ... » reconstruisait la ligne sans l'espace de tete et le VTY la refusait)
         awk -v inc=$((100 * (INSTANCE - 1))) '/^[[:space:]]*bind 127\.0\.0\.1 [0-9]+/ { n = $3 + inc;
 ↪  sub(/[0-9]+$/, n) } { print }' "$_src" \
             | sed -e "s#/tmp/osmocom_l2\b#$L2_SOCK#" -e "s#/tmp/osmocom_sap\b#/tmp/osmocom_sap_$INSTANCE#" -e
 ↪  "s#/tmp/ms_data\b#/tmp/ms_data_$INSTANCE#" > "$_dst"
+        # [2026-10-21] « stick » suit l'ARFCN du plan radio (BTS#0) quand il a change (gabarit : 514)
+        local _arfcn; _arfcn="$( . /etc/osmocom/radio-plan.env 2>/dev/null; echo "${PLAN_ARFCN:-}")"
+        [ -n "$_arfcn" ] && sed -i "s|^\([[:space:]]*\)stick [0-9]*\$|\1stick $_arfcn|" "$_dst"
         MOBILE_CFG="$_dst"
         dire "4. instance $INSTANCE : config mobile derivee -> $MOBILE_CFG"
     fi
diff --git a/src/montant.c b/src/montant.c
index 7a555d3..b17f594 100644
--- a/src/montant.c
+++ b/src/montant.c
@@ -967,8 +967,11 @@ int montant_tn_dedie(void)
     return g_tn_tch_sb;
 }

+void tx_rach_vers_air(uint8_t ra, uint8_t bsic, uint32_t tick);   /* pont.c */
+
 static void publier_rach(uint8_t ra, uint8_t bsic, uint32_t fn)
 {
+    tx_rach_vers_air(ra, bsic, fn);   /* [2026-10-21] Magnetite : vers l'air (CALYPSO_RACH_VERS_AIR) */
```

```
     static int fd = -2;
     static uint32_t seq;
     if (fd == -2) {
@@ -1138,6 +1141,9 @@ static void publier_kc(uint16_t *api_ram)
     fflush(stdout);
 }

+void calypso_bsp_set_dedie_ti(int actif);   /* calypso_bsp.c */
+static bool g_ti_dedie;                      /* firmware TI en dedie (taches ALLC), voir plus bas */
+
 void montant_scruter(uint16_t *api_ram, uint32_t fn, unsigned page)
 {
     static int coupe = -1;
@@ -1220,12 +1226,16 @@ void montant_scruter(uint16_t *api_ram, uint32_t fn, unsigned page)
     {
         static int sonde = -1;
         if (sonde < 0) { const char *e = calypso_getenv("MONTANT_ACD"); sonde = (e && *e == '0') ? 0 : 1; }
-        if (sonde && g.dedie_arme) {
+        if (sonde && (g.dedie_arme || g_ti_dedie)) {
             static uint16_t prec;
             static unsigned long ok, ko;
             uint16_t etat = api_ram[(API_NDB + NDB_A_CD) / 2];
-            if ((etat & B_BLUD) && etat != prec) {
-                const uint8_t *d = (const uint8_t *)api_ram + API_NDB + NDB_A_CD + 6;
+            /* [22:15] en dedie TI : chaque bloc (le filtre etat != prec cachait les FIRE KO consecutifs) */
+            static uint8_t d_prec[4];
+            const uint8_t *d = (const uint8_t *)api_ram + API_NDB + NDB_A_CD + 6;
+            bool nouveau = (etat != prec) || (g_ti_dedie && memcmp(d, d_prec, 4) != 0);
+            if ((etat & B_BLUD) && nouveau) {
+                memcpy(d_prec, d, 4);
                 bool fire = (etat & 0x0040) != 0;
                 if (fire) ko++; else ok++;
                 if ((ok + ko) <= 60 || ((ok + ko) % 50) == 0) {
@@ -1244,6 +1254,46 @@ void montant_scruter(uint16_t *api_ram, uint32_t fn, unsigned page)
     uint16_t task_ra = wp[WP_D_TASK_RA / 2];
     uint16_t d_rach  = api_ram[(API_NDB + NDB_D_RACH) / 2];

+    /* [2026-10-21 21:50] SONDE DU MONTANT DEDIE (Magnetite, BTS classique). Apres l'IMM ASS la L1 TI passe sur
+     * le SDCCH (DBA_I) mais aucun bloc montant n'a ete publie : capture_sdcch_ul n'etait meme jamais appelee
+     * (task_u toujours 0 ici). A la premiere tache ALLC vue dans une page W, 300 lignes : taches des deux pages,
+     * d_dsp_page, page choisie, en-tete de a_cu. Et si a_cu porte B_BLUD sans tache montante vue, on le prend
+     * quand meme (la L1 TI le pose avec le bloc a emettre, l1ddsp_load_info). */
+    {
+        static unsigned reste;
+        static bool arme;
+        uint16_t td0 = api_ram[(API_W_PAGE(0) + WP_D_TASK_D) / 2] & 0x7fffu;
+        uint16_t td1 = api_ram[(API_W_PAGE(1) + WP_D_TASK_D) / 2] & 0x7fffu;
+        uint16_t cu0 = api_ram[(API_NDB + NDB_A_CU) / 2];
+        if (!arme && (td0 == ALLC_DSP_TASK || td1 == ALLC_DSP_TASK)) {
+            arme = true;
+            reste = 300;
+            printf("  [ul-dedie] premiere tache ALLC (tick %u) : sonde du montant sur 300 trames\n", fn);
+        }
+        if (reste) {
+            reste--;
+            printf("  [ul-dedie] tick=%u dsp_page=%04x pg=%u W0 d=%u u=%u W1 d=%u u=%u a_cu[0]=%04x\n", fn,
 ↪  v_page, pg,
+                   td0, api_ram[(API_W_PAGE(0) + WP_D_TASK_U) / 2] & 0x7fffu,
+                   td1, api_ram[(API_W_PAGE(1) + WP_D_TASK_U) / 2] & 0x7fffu, cu0);
+        }
+        if (arme && task_u == 0 && (cu0 & B_BLUD))
+            capture_sdcch_ul(api_ram, 0, fn);
+        /* [2026-10-21 22:05] etat dedie du firmware TI : actif tant qu'une tache ALLC a ete vue dans les 102
+         * dernieres trames. Arme la sonde a_cd et le decalage BSP en dedie (CALYPSO_BSP_DEDIE_DECAL). */
+        /* [22:15] sur les TACHES MONTANTES (d_task_u, 11/12 vus a la sonde) : ALLC (24) sert aussi au BCCH et
+         * au CCCH en veille (a_cd : SI3, paging lus « en dedie »), le decalage aurait frappe la veille. */
```

```
+        static uint32_t derniere_allc;
+        static bool vu_allc;
+        uint16_t tu0 = api_ram[(API_W_PAGE(0) + WP_D_TASK_U) / 2] & 0x7fffu;
+        uint16_t tu1 = api_ram[(API_W_PAGE(1) + WP_D_TASK_U) / 2] & 0x7fffu;
+        if (tu0 || tu1) { derniere_allc = fn; vu_allc = true; }
+        bool dedie = vu_allc && (uint32_t)(fn - derniere_allc) < 102u;
+        if (dedie != g_ti_dedie) {
+            g_ti_dedie = dedie;
+            calypso_bsp_set_dedie_ti(dedie);
+        }
+    }
+
     /* RACH : la tache, pas la valeur.
      *
     * [2026-09-21, mesure sur le banc reel] Le mot NDB d_rach (octet 0x474
@@ -1300,9 +1350,29 @@ void montant_scruter(uint16_t *api_ram, uint32_t fn, unsigned page)
     }
     bool nouveau_rach = !front_seul || !tache_avant || front_valeur;
     tache_avant = tache_rach;
+    /* [2026-10-21 21:25] RACH DIFFERE (MONTANT_RACH_DIFFERE=N trames, Magnetite : 1 par run.sh). La L1 TI pose
+     * d_task_ra AVANT d'ecrire le d_rach de la tentative : lu au front, d_rach etait celui de la tentative
+     * PRECEDENTE (capture CCCH du 21/10 21:17 : RA_R 9868 ra 4 publie ra 8, 10114 ra 8 publie ra 4...), l'IMM ASS
+     * portait le bon fn mais le RA d'avant et Magnetite l'ignorait. On garde le tick du front et on lit d_rach
+     * N trames plus tard. */
+    static int differe = -1;
+    static bool rach_attente;
+    static uint32_t fn_attente;
+    if (differe < 0) {
+        const char *e = calypso_getenv("MONTANT_RACH_DIFFERE");
+        differe = (e && *e) ? atoi(e) : 0;
+        if (differe > 0) printf("  [montant] RACH differe de %d trame(s) : d_rach lu apres le front de
 ↪  d_task_ra\n", differe);
+    }
+    if (rach_attente && (uint32_t)(fn - fn_attente) >= (uint32_t)differe) {
+        rach_attente = false;
+        if (d_rach != 0)
+            publier_rach((uint8_t)(d_rach >> 8), (uint8_t)((d_rach & 0xff) >> 2), fn_attente);
+        g.prev_rach = d_rach;
+    }
     if (d_rach != 0 && (tache_rach || (sur_drach && front_valeur)) && nouveau_rach &&
         (!g.rach_vu || (uint32_t)(fn - g.fn_rach) >= RACH_GARDE_TRAMES)) {
-        publier_rach((uint8_t)(d_rach >> 8), (uint8_t)((d_rach & 0xff) >> 2), fn);
+        if (differe > 0) { rach_attente = true; fn_attente = fn; }
+        else publier_rach((uint8_t)(d_rach >> 8), (uint8_t)((d_rach & 0xff) >> 2), fn);
         g.prev_rach = d_rach;
         g.fn_rach = fn;
         g.rach_vu = true;
diff --git a/src/pont.c b/src/pont.c
index f4a3080..56761a9 100644
--- a/src/pont.c
+++ b/src/pont.c
@@ -686,7 +686,7 @@ static void tx_rom_pose(const uint16_t *api_ram, int blud_avant)
  * les pages sans tache RX/TX (RACH) gardent un a_a5fn perime : on prend alors tick + decalage, le
  * decalage etant celui de la derniere page datee. */
 #define GSM_HYPERFRAME 2715648u
-static struct { bool valide; uint32_t fn; bool dec_valide; int32_t dec; uint32_t tick; bool tache_tx; } g_tsp_fn;
+static struct { bool valide; uint32_t fn; bool dec_valide; int32_t dec; uint32_t tick, tick_dec; bool tache_tx; }
 ↪  g_tsp_fn;
 static uint32_t gsmtime2fn(unsigned t1, unsigned t2, unsigned t3)
 {
     return (1326u * t1 + 51u * ((t3 + 26u - t2) % 26u) + t3) % GSM_HYPERFRAME;
@@ -713,11 +713,23 @@ static void tx_tsp_dater(const C54xState *dsp, uint32_t tick)
     g_tsp_fn.fn = gsmtime2fn(t1, t2, t3); g_tsp_fn.valide = true;
     g_tsp_fn.dec = (int32_t)((g_tsp_fn.fn + GSM_HYPERFRAME - (tick % GSM_HYPERFRAME)) % GSM_HYPERFRAME);
     g_tsp_fn.dec_valide = true;
+    g_tsp_fn.tick_dec = tick;
```

```
 }
 static uint32_t tx_tsp_fn(uint32_t tick, int *source)
 {
     if (g_tsp_fn.valide && g_tsp_fn.tick == tick) { *source = 1; return g_tsp_fn.fn; }
-    if (g_tsp_fn.dec_valide) { *source = 2; return (uint32_t)((tick + (uint32_t)g_tsp_fn.dec) % GSM_HYPERFRAME); }
+    /* [2026-10-21 23:10] decalage a_a5fn PERIME au-dela de 102 trames (CALYPSO_TSP_DEC_TTL ; 0 = jamais) : la L1
 ↪  TI ne
+     * pose a_a5fn qu'en connexion chiffree ; en veille le mobile se resynchronise (SB) et son fn glisse d'une
 ↪  trame
+     * par rapport au decalage fige de la derniere connexion -> RACH date fn+1 (RA_C 33884 contre 33885 publie),
+     * IMM ASS ignorees, echec d'acces, reselection (barres perdues), paging sans reponse (MT SMS en echec).
 ↪  Au-dela,
+     * le calage BSP (source 3), qui suit les recalages de l'offset. */
+    static int ttl = -1;
+    if (ttl < 0) { const char *e = calypso_getenv("CALYPSO_TSP_DEC_TTL"); ttl = (e && *e) ? atoi(e) : 102; }
+    if (g_tsp_fn.dec_valide && (ttl == 0 || (uint32_t)(tick - g_tsp_fn.tick_dec) <= (uint32_t)ttl)) {
+        *source = 2; return (uint32_t)((tick + (uint32_t)g_tsp_fn.dec) % GSM_HYPERFRAME);
+    }
+    /* [2026-10-21 18:45] firmware sans a_a5fn (L1 TI, Magnetite) : la trame BTS du calage TS0 */
+    { uint32_t f; if (calypso_bsp_fn_bts_du_tick(tick, &f)) { *source = 3; return f; } }
     *source = 0; return tick;
 }
 #define ROM_TX_SHM   "/dev/shm/calypso_tx_rom"
@@ -853,13 +865,80 @@ static void tx_tsp_publier(const C54xState *dsp, uint32_t fn)
     static int nlog;
     if (nlog++ < 6 || (b.type == TSP_TX_RACH && n_rach <= 3))
         printf("  [rom-tx] #%u tick=%u fn=%u (%s) %s%s offset=%d @%04x : %s...\n", w_compte, fn, fn_gsm,
-               src == 1 ? "a_a5fn" : src == 2 ? "tick+dec" : "TICK NU",
+               src == 1 ? "a_a5fn" : src == 2 ? "tick+dec" : src == 3 ? "calage BSP" : "TICK NU",
                b.type == TSP_TX_RACH ? "RACH" : b.type == TSP_TX_NB ? "NB" : b.type == TSP_TX_SB ? "SB" : b.type
 ↪  == TSP_TX_DUMMY ? "DUMMY" : b.type == TSP_TX_FB ? "FB" : "?", b.type == TSP_TX_NB ? (char[]){' ', 'T', 'S',
 ↪  'C', (char)('0' + b.tsc), 0} : "",
               b.offset, b.adresse, (char[]){ '0' + b.bits[0], '0' + b.bits[1], '0' + b.bits[2], '0' + b.bits[3],
 ↪  '0' + b.bits[4], '0' + b.bits[5], '0' + b.bits[6], '0' + b.bits[7], 0 });
     if ((n_rach + n_nb + n_sb + n_dummy + n_fb + n_inconnu) % 2000 == 0)
         printf("  [rom-tx] %u bursts publies : RACH %u, NB %u, SB %u, DUMMY %u, FB %u, inconnus %u\n", n_rach +
 ↪  n_nb + n_sb + n_dummy + n_fb + n_inconnu, n_rach, n_nb, n_sb, n_dummy, n_fb, n_inconnu);
 }

+/* [2026-10-21 19:00] RACH VERS L'AIR (CALYPSO_RACH_VERS_AIR=1, Magnetite). La capture au script TSP de la
+ * ROM (0x3cbb/0x3ccd, chemin du firmware osmocom) ne voit pas le burst d'acces du firmware TI : montant.c
+ * detectait bien les RACH (d_task_ra, ra/bsic) mais ils n'allaient qu'a /dev/shm/calypso_rach, que l'air
+ * ne lit pas -> B->A 0, pas de CHAN RQD, « Network forbidden ». On code le burst d'acces (05.03,
+ * gsm0503_rach_ext_encode) et on le publie dans calypso_tx_rom, date de la trame BTS du calage. */
+/* [2026-10-21 19:40] a_pm JAMAIS NUL (CALYPSO_BSP_BRUIT_FOND=1, Magnetite). La L1 TI lit pm = a_pm >> 5
+ * (l1ddsp_meas_read) et compte chaque pm nul comme une panne (l1_check_pm_error : 100 -> recovery, DSP
+ * relance). Le modele de puissance du banc (calypso_bsp_rssi_apm) ne passe que sur les ecritures du DSP ;
+ * un a_pm laisse a 0 par la ROM partait tel quel. Avant de rendre la trame a l'ARM, chaque mot a_pm nul
+ * des deux pages R recoit le modele (plancher 64). */
+static void apm_jamais_nul(uint16_t *api_ram)
+{
+    static int on = -1;
+    if (on < 0) { const char *e = calypso_getenv("CALYPSO_BSP_BRUIT_FOND"); on = (e && *e == '1'); }
+    if (!on) return;
+    for (int pg = 0; pg < 2; pg++) {
+        uint16_t *a = &api_ram[(API_R_PAGE(pg) + RP_A_PM) / 2];
+        for (int k = 0; k < 3; k++)
+            if ((a[k] >> 5) == 0) a[k] = calypso_bsp_rssi_apm();
+    }
+    /* [2026-10-21 20:00] MESURE PLEINE LISTE (RXL_R 548) : la L1 TI (DSP 36) programme d_task_md = 0x200 + n
+     * (n <= 8 mesures, l1_sync.c) et lit les n PM dans les mots 12..12+n-1 de la page R (l1ddsp_meas_read,
+     * nbmeas = 8 vu au gdb) : au-dela des 3 mots a_pm, ce sont les mots de a_sch. Le modele de puissance du
+     * banc ne couvrait que les mots 12..14 : 15..19 restaient a 0 (vu au moniteur, d_task_md 0x0208), 5 PM
+     * nuls sur 8 a chaque trame du balayage -> recovery. La page R reprend d_task_md (mot 4) : on s'en sert. */
+    /* [2026-10-21 21:50] MODE DEDIE : les taches SDCCH/SACCH (ADL 22, DDL 24 cote L1 TI) lisent la puissance dans
+     * a_serv_demod[D_PM], mot 9 de la page R pour le DSP 36 (l1_defty.h : a_serv_demod 8..11). Le DSP emule l'y
```

```
+     * laissait a 0 apres l'IMM ASS : « > PM 1778 22 », « > PM 1808 24 » a chaque trame, ABORT, retour en veille.
+     * Meme plancher que a_pm, des qu'une tache descendante (d_task_d, mot 0) est rapportee. */
+    for (int pg = 0; pg < 2; pg++) {
+        uint16_t *r = &api_ram[API_R_PAGE(pg) / 2];
+        if ((r[0] & 0x7fff) && (r[9] >> 5) == 0) r[9] = calypso_bsp_rssi_apm();
+    }
+    for (int pg = 0; pg < 2; pg++) {
+        uint16_t *r = &api_ram[API_R_PAGE(pg) / 2];
+        uint16_t md = r[4];
+        if (!(md & 0x200)) continue;
+        int n = md & 0xff; if (n > 8) n = 8;
+        for (int k = 0; k < n; k++)
+            if ((r[12 + k] >> 5) == 0) r[12 + k] = calypso_bsp_rssi_apm();
+    }
+}
+
+void tx_rach_vers_air(uint8_t ra, uint8_t bsic, uint32_t tick)
+{
+    static int actif = -1;
+    if (actif < 0) { const char *e = calypso_getenv("CALYPSO_RACH_VERS_AIR"); actif = (e && *e == '1'); }
+    if (!actif) return;
+    /* [2026-10-21 20:55] le BURST D'ACCES ENTIER (05.02 5.2.7), comme le chemin ROM (tsp_tx.c, offset 8) :
+     * 8 bits de queue etendue, 41 de synchro, les 36 codes, 3 de queue. gsm0503_rach_ext_encode ne rend que
+     * les 36 bits codes : publies seuls en tete, le TRX n'avait aucune synchro a correler (transceiver :
+     * aucun « TOA : », osmo-bts rach:rcvd 0). */
+    static const char queue[] = "00111010", sync[] = "0100101101111111100110011010101000111100";
+    uint8_t code[36], bits[148] = {0};
+    if (gsm0503_rach_ext_encode(code, ra, bsic, false) < 0) return;
+    for (int i = 0; i < 8; i++) bits[i] = (uint8_t)(queue[i] - '0');
+    for (int i = 0; i < 41; i++) bits[8 + i] = (uint8_t)(sync[i] - '0');
+    memcpy(bits + 49, code, 36);
+    int src = 0;
+    uint32_t fn_gsm = tx_tsp_fn(tick, &src);
+    if (!ring_publier(TSP_TX_RACH, bits, -1, 8, (uint8_t)src, fn_gsm)) return;
+    static unsigned n;
+    if (++n <= 20 || n % 100 == 0)
+        printf("  [rach-air] RACH ra=0x%02x bsic=%u tick=%u -> fn BTS %u (%s) publie vers l'air [%u]\n",
+               ra, bsic, tick, fn_gsm, src == 1 ? "a_a5fn" : src == 2 ? "tick+dec" : src == 3 ? "calage BSP" :
 ↪  "TICK NU", n);
+}
+
 /* [2026-10-20] MODE TRX : PUBLICATION DEPUIS LA PAGE DE L'EXTCODE. Le firmware trx remplit tx[p] (cmd,
  * pointeur de donnees) pour chaque slot p de sa fenetre TX (prim_bts.c patche : slot p = TS (tx_start + p)),
  * et l'extcode du DSP les module ; la ROM emulee ne joue qu'un burst par trame (le script TSP) et le
@@ -1962,21 +2041,58 @@ static void servir(int fd, C54xState *dsp, uint16_t *api_ram, long insns, bool v
     printf("pont : ARM connecte, API RAM %u mots, %ld insn/trame\n", CALYPSO_API_WORDS, insns);
     fflush(stdout);

+    /* [2026-10-21 18:35] PONT_RESET arrive pendant l'attente du PONT_GO (trame en deux phases) : il
+     * etait JETE (« message 1 recu en attente de GO, ignore »). Le firmware TI relance le DSP en cours
+     * de route (fn ~290) et attend alors LEAD_READY sans fin : l'ARM restait dans dsp_power_on+0x34.
+     * Le reset est desormais garde et applique des la fin de la trame. */
+    unsigned reset_differe = 0;
+    uint32_t reset_differe_dl = 0;
     while (!g_stop) {
         if (g_pause) pause_cooperative(dsp);
-        struct pollfd pfd = { .fd = fd, .events = POLLIN };
-        if (poll(&pfd, 1, 200) <= 0) {
-            continue;
-        }
-        ssize_t n = recv(fd, &m, sizeof(m), 0);
-        if (n != (ssize_t)sizeof(m)) {
-            printf("pont : ARM deconnecte (%zd)\n", n);
-            break;
+        if (reset_differe) {
+            reset_differe--;
```

```
+            memset(&m, 0, sizeof m);
+            m.type = PONT_RESET;
+            m.a = reset_differe_dl;
+            printf("pont : reset garde pendant l'attente du GO, applique maintenant\n");
+        } else {
+            struct pollfd pfd = { .fd = fd, .events = POLLIN };
+            if (poll(&pfd, 1, 200) <= 0) {
+                continue;
+            }
+            ssize_t n = recv(fd, &m, sizeof(m), 0);
+            if (n != (ssize_t)sizeof(m)) {
+                printf("pont : ARM deconnecte (%zd)\n", n);
+                break;
+            }
         }
         switch (m.type) {
-        case PONT_RESET:
-            /* Same as qosmo-dsp: c54x_reset, running, boot driven by the ticks, d_dsp_page=0 */
+        case PONT_RESET: {
+            /* [2026-10-21 18:30] LE bootCode TI. Sur le Calypso C035 l'API est aussi l'espace PROGRAMME
+             * 0xE000-0xFFFF du DSP (leadapi.h BASE_API_LEAD 0xE000). Le firmware TI (Magnetite,
+             * LA_InitialLeadBoot, leadboot.c) y pose avant de relacher le reset un chargeur (0xf800), une
+             * init (0xfe00) et le VECTEUR DE RESET (0xff80 : F073 FE00 = b 0xfe00) : chaque reset passe
+             * par ce chargeur, qui affiche LEAD_READY. Ici 0xE000-0xFFFF etait la « PDROM » figee, et
+             * c54x_reset recopie une image ROM dans la fenetre API des donnees : la ROM partait a chaud
+             * (0xb3e7) au 2e reset, sans LEAD_READY, et l'ARM restait dans dsp_power_on+0x34 (attente sans
+             * fin, NO_DWNLD). Signature presente : l'API est gardee telle que l'ARM l'a ecrite (un reset du
+             * DSP ne touche pas la RAM partagee) et le DSP repart dans le chargeur, comme avec le bootCode.
+             * Firmware osmocom (pas de signature) : inchange. */
+            static uint16_t api_avant[0x2000];
+            const bool boot_ti = api_ram[0xff80 - 0xe000] == 0xF073 && api_ram[0xff81 - 0xe000] == 0xFE00;
+            if (boot_ti) memcpy(api_avant, api_ram, sizeof api_avant);
             c54x_reset(dsp);
+            if (boot_ti) {
+                /* [18:30] Executer le bootCode lui-meme (fenetre programme 0xE000) faisait planter le DSP
+                 * (SP-CORRUPT a PC 0 : modes memoire du C035 non emules). On en garde la FONCTION : DSP
+                 * pose dans le chargeur de la ROM (0xb410 : st #1,*(0xfff) puis attente du 2 et saut),
+                 * avec les registres que la ROM y met (0xb3ff-0xb403). API gardee telle que l'ARM l'a ecrite. */
+                memcpy(api_ram, api_avant, sizeof api_avant);
+                for (int i = 0; i < 0x2000; i++) dsp->data[0x0800 + i] = api_avant[i];
+                dsp->st1 = 0x2900; dsp->st0 = 0x1800; dsp->sp = 0x5AC8;
+                dsp->pc = 0xb410;
+                printf("pont : bootCode TI dans l'API (0xff80 = b 0xfe00) : API gardee, DSP dans le chargeur ROM
 ↪  0xb410\n");
+            }
             dsp->running = true;
             init_done = false;
             api_ram[(API_NDB + NDB_D_DSP_PAGE) / 2] = 0;
@@ -1984,6 +2100,7 @@ static void servir(int fd, C54xState *dsp, uint16_t *api_ram, long insns, bool v
             printf("pont : RESET #%lu (DL_STATUS=0x%04x) fn=%u pc=0x%04x\n",
                    resets, m.a, g_c54x_exe_fn, dsp->pc);
             break;
+        }
         case PONT_TICK: {
             g_c54x_exe_fn = m.a;
             g_tick_irq_trame = (m.b & CALYPSO_PONT_TICK_IRQ_TRAME) != 0;
@@ -2075,6 +2192,7 @@ static void servir(int fd, C54xState *dsp, uint16_t *api_ram, long insns, bool v
                     if (trx_page_mode()) tx_page_publier(dsp, m.a, w_page);
                     calypso_bsp_trx_ul_prothese(dsp->data, m.a, w_page);
                 }
+                apm_jamais_nul(api_ram);
                 envoyer(fd, PONT_DONE, drapeaux & ~PONT_DONE_API_IRQ, ninsn);
                 t_done_a = chrono_ms();
                 bool go = false;
@@ -2085,6 +2203,7 @@ static void servir(int fd, C54xState *dsp, uint16_t *api_ram, long insns, bool v
                     ssize_t ng = recv(fd, &g, sizeof(g), 0);
```

```
                    if (ng != (ssize_t)sizeof(g)) { printf("pont : ARM deconnecte en attente de GO (%zd)\n", ng);
 ↪  g_stop = 1; break; }
                    if (g.type == PONT_GO) { go = true; t_go = chrono_ms(); }
+                   else if (g.type == PONT_RESET) { reset_differe = 1; reset_differe_dl = g.a; }   /* voir
 ↪  reset_differe */
                    else { static unsigned nx; if (nx++ < 3) printf("pont : message %u recu en attente de GO,
 ↪  ignore\n", g.type); }
                }
                if (!go) break;
@@ -2179,8 +2298,11 @@ static void servir(int fd, C54xState *dsp, uint16_t *api_ram, long insns, bool v
            for (int k = 0; k < 40 && dsp->running; k++) {   /* 13 page pairs per 1250-symbol frame */
                /* [2026-09-29] un SCH garde parce que la DMA n'etait pas armee au
                 * depot est relivre ici, des que la ROM (phase B) l'a armee. */
-               calypso_bsp_sb_retenter();
-               if (!calypso_rhea_dma_pump(dsp)) break;
+               int relivre = calypso_bsp_sb_retenter();
+               /* [2026-10-21 22:45] dedie TI : fenetre de mesure servie / NB garde relivre (calypso_bsp.c) ;
+                * on laisse alors tourner le DSP (fin de DMA -> ISR -> fenetre suivante) dans le meme tick */
+               relivre |= calypso_bsp_nb_retenter();
+               if (!calypso_rhea_dma_pump(dsp) && !relivre) break;
                if (dsp->idle && (dsp->ifr & dsp->imr) && !(dsp->st1 & 0x800)) dsp->idle = false;
                uint32_t av2 = dsp->insn_count;
                if (!dsp->idle) c54x_run_profile(dsp, (int)insns / 4);
@@ -2241,6 +2363,21 @@ static void servir(int fd, C54xState *dsp, uint16_t *api_ram, long insns, bool v
                /* SB: the firmware reads a_serv_demod[D_TOA] from the read page
                 * and expects about 4. PONT_CAN_SB_TOA=4 pins it on both R
                 * pages, which frames the SCH correctly. */
+               /* [2026-10-21] PONT_CAN_NB_TOA=<v> (Magnetite : 3) : la L1 TI ne converge pas sur le TOA
+                * des bursts normaux (l1ctl_toa_update : ISH = (TOA+1)*4 - 16 qbits, le banc ne deplace
+                * pas le burst quand tpu_offset bouge) ; on le fixe sur les pages R portant une tache NB
+                * (d_task_d != 0). Le TOA du SB (a_sch, 23) reste natif : la synchro en depend. */
+               static int can_nb = -2;
+               if (can_nb == -2) { const char *e = calypso_getenv("PONT_CAN_NB_TOA"); can_nb = (e && *e) ?
 ↪  atoi(e) : -1;
+                                   if (can_nb >= 0) printf("  [pont] PONT_CAN_NB_TOA=%d : TOA des NB fixe\n",
 ↪  can_nb); }
+               /* init_done : pendant le boot ces mots de l'API RAM portent le code DSP que l'ARM
+                * telecharge ; y ecrire cassait le telechargement (« occupe boot », 21/10 17:01). */
+               if (can_nb >= 0 && init_done) {
+                   for (int pg = 0; pg < 2; pg++) {
+                       uint16_t *r = &api_ram[API_R_PAGE(pg) / 2];
+                       if (r[RP_D_TASK_D / 2] && r[RP_D_TASK_D / 2] <= 40) r[RP_A_SERV_DEMOD / 2 + D_TOA] =
 ↪  (uint16_t)can_nb;
+                   }
+               }
                static int can_sb = -2;
                if (can_sb == -2) { const char *e = calypso_getenv("PONT_CAN_SB_TOA"); can_sb = (e && *e) ?
 ↪  atoi(e) : -1; }
                if (can_sb >= 0) {
@@ -2306,7 +2443,7 @@ static void servir(int fd, C54xState *dsp, uint16_t *api_ram, long insns, bool v
            insns_total += ninsn;
            if (drapeaux & PONT_DONE_API_IRQ) irqs++;
            enreg_api_prendre(api_ram);
-           if (!done_envoye) { ombre_descendante(api_ram, m.a); envoyer(fd, PONT_DONE, drapeaux, ninsn); }
+           if (!done_envoye) { apm_jamais_nul(api_ram); ombre_descendante(api_ram, m.a); envoyer(fd, PONT_DONE,
 ↪  drapeaux, ninsn); }
            { double t_fin = chrono_ms(); chrono_trame(t_tick, t_done_a, t_go, t_done > 0 ? t_done : t_fin,
 ↪  t_fin, m.a); }
            if (!init_avant && init_done) {
                printf("pont : DSP boote (premier IDLE) fn=%u insn=%u\n", m.a, dsp->insn_count);
```

3.273 /opt/GSM/osmo-operator/patches/c54x_exe-rssi-air-et-arret-instance.patch

4112 octets, 64 lignes → 64 lignes

```
diff --git a/run.sh b/run.sh
index 767103e..142271b 100755
--- a/run.sh
+++ b/run.sh
@@ -64,6 +64,7 @@ if [ -r "$INSTANCE_ENV" ]; then
     printf '\033[1m[run dsp]\033[0m instance %s : reglages de %s\n' "$INSTANCE" "$INSTANCE_ENV"
 fi
 HOTE_SHM=/run/calypso-shm-hote
+AIR_DEMANDE="${AIR:-}"   # [2026-10-23] AIR pose par l appelant (sinon defaut par instance / appli, plus bas)
 if [ "$INSTANCE" != 1 ] && [ -z "${CALYPSO_NS:-}" ]; then
     command -v unshare >/dev/null || { echo "[run] ECHEC : unshare absent (util-linux)" >&2; exit 1; }
     _d="/dev/shm/calypso$INSTANCE"; mkdir -p "$_d" "$HOTE_SHM" && chmod 1777 "$_d"
@@ -102,6 +103,15 @@ if [ -n "$FIRMWARE_APP" ]; then
     FIRMWARE_BIN="${FIRMWARE_ELF%.elf}.bin"
     PONT=0
 fi
+# [2026-10-23] RSSI : l'air realiste par defaut (calypso_bsp.c CALYPSO_AIR_NIVEAUX : la cellule seulement sur son
+# ARFCN, niveaux DL/UL, bruit ailleurs) et, en BTS classique (pont_dsp.py vivant), pas d'air.py : le descendant
+ ↪  arrive
+# par la copie du pont (PONT_DSP_COPIES, udp 6762). Ici et non dans banc-rssi : un terminal ouvert avant la mise a
+# jour des alias, ou un autre lanceur, relancait rssi avec l'ancien comportement.
+if [ "$FIRMWARE_APP" = rssi ]; then
+    export CALYPSO_AIR_NIVEAUX="${CALYPSO_AIR_NIVEAUX:-1}"
+    export CALYPSO_FAUX_AIR="${CALYPSO_FAUX_AIR:-$( . /etc/osmocom/radio-plan.env 2>/dev/null; echo
 ↪  "${PLAN_ARFCN:-}")}"
+    [ -z "$AIR_DEMANDE" ] && pgrep -f 'pont/pont_dsp\.py' >/dev/null && AIR=0
+fi
 # [2026-10-20] OSMO_TRX_TS1=1 (instance 1, avec OSMO_TRX=1) : un DEUXIEME Calypso TRX pour le TS1 (TCH/F, les
 ↪  appels) :
 # l'instance OSMO_TRX_TS1_INSTANCE (3) est lancee avant le transceiver (etapes 1-3 seules, OSMO_TRX_PHONE=2 :
 ↪  firmware trx,
 # L1CTL /tmp/osmocom_l2_3, pas de transceiver, pas d'air) et le transceiver prend « -2 -L /tmp/osmocom_l2_3 »
 ↪  (telephone 2 =
@@ -694,13 +704,34 @@ arreter() {
         rm -f "$RUNDIR/$n.pid"
     done; bruit_arreter
     sleep 1
+    # [2026-10-23] ARRET COMPLET D'UNE INSTANCE N != 1 (banc-rssi --stop) : les processus sans fichier pid ou dont
+    # le pid a ete perdu (relance par-dessus une instance vivante) gardaient le port BSP (udp 6762 tenu par un
+    # c54x_exe fantome, le nouveau ne recevait rien). On balaie ce qui porte les ressources propres a l'instance.
+    if [ "$INSTANCE" != 1 ]; then
+        # Pas « $RUNDIR/ » : un simple tail -F de ses journaux, ou le shell qui a lance --stop, y passaient.
+        local motif="$DSP_SOCK|$MONITOR|$L2_SOCK|--stub $GDB_STUB |--port $GDB_TELNET "
+        local p=$$ ancetres=" $$ "
+        while p="$(awk '/^PPid:/ {print $2}' "/proc/$p/status" 2>/dev/null)" && [ -n "$p" ] && [ "$p" -gt 1 ]; do
+            ancetres="$ancetres$p "
+        done
+        local restes="" r
+        for r in $(pgrep -f -- "$motif"); do
+            case "$ancetres" in *" $r "*) ;; *) restes="$restes$r " ;; esac
+        done
+        if [ -n "${restes// /}" ]; then
+            dire "instance $INSTANCE : restes arretes (pid $restes)"
+            kill $restes 2>/dev/null; sleep 1; kill -9 $restes 2>/dev/null
+        fi
+    fi
     rm -f "$DSP_SHM" "$DSP_SOCK" "$L2_SOCK" "$MONITOR" "$RUNDIR/modem.pty" /dev/shm/qemu-asm.fifo
     # [2026-09-22] Sockets du mobile : elles ne disparaissent pas avec lui. Le
     # « sap » etait deja oublie, et « ms_data » (tch-data, mobile_pont.cfg)
     # arrive avec la meme faiblesse : un fichier reste fait echouer le bind du
     # run suivant sur EADDRINUSE, et le mobile demarre sans sa voie donnees
     # sans le dire.
-    rm -f /tmp/osmocom_sap /tmp/ms_data
+    # [2026-10-23] SEULEMENT pour l'instance 1 : ces sockets sont dans le /tmp COMMUN et appartiennent au mobile
+ ↪  de
+    # l'instance 1 ; banc-rssi --stop (instance 4) les effacait sous le mobile en marche (SAP, tch-data).
```

```
+    [ "$INSTANCE" = 1 ] && rm -f /tmp/osmocom_sap /tmp/ms_data
     # [2026-09-22] Horloge du banc (calypso_bsp.c -> pont/trx.py) : laissee en
     # place, pont.py asservirait sa premiere seconde sur la trame d'une session
     # morte et la BTS resterait figee en attendant un DSP qui n'existe plus.
```

3.274 /opt/GSM/osmo-operator/patches/c54x_exe-sonde-depot-tch-irq.patch

1411 octets, 22 lignes → 22 lignes

```
diff --git a/src/pont.c b/src/pont.c
index f4a3080..18acd61 100644
--- a/src/pont.c
+++ b/src/pont.c
@@ -1260,9 +1342,14 @@ static uint32_t jouer_trame(C54xState *dsp, long budget, bool *init_done, uint32
                 fait += c54x_run_profile(dsp, 256);
             if (jusqua_idle) {
                 static unsigned n_tch;
-                if (n_tch++ < 30)
-                    printf("  [depot_tch] fn=%u phase A : %d insn, idle=%d, fenetre armee=%d%s\n",
-                           g_c54x_exe_fn, fait, dsp->idle, calypso_rhea_dma_rx_armed(),
+                /* [2026-10-21 23:40] + irq_trame/IMR/IFR/INTM/pc : Magnetite sur TCH TS1, 3 trames puis
+                 * « 0 insn, idle=1 » et L1 TI « > COM » (d_debug=0) : IT trame non levee (QEMU) ou
+                 * bit TPU frame absent de l'IMR (ROM) ? 200 lignes pour couvrir plusieurs appels. */
+                if (n_tch++ < 200)
+                    printf("  [depot_tch] fn=%u phase A : %d insn, idle=%d, fenetre armee=%d irq_trame=%d "
+                           "IMR=%04x IFR=%04x INTM=%d pc=%04x%s\n",
+                           g_c54x_exe_fn, fait, dsp->idle, calypso_rhea_dma_rx_armed(), g_tick_irq_trame,
+                           dsp->imr, dsp->ifr, !!(dsp->st1 & 0x800), dsp->pc & 0xffff,
                            dsp->idle ? "" : "  <- BUDGET/2 ATTEINT, depot avant la fin");
             }
             fait_a = fait;
```

3.275 /opt/GSM/osmo-operator/patches/fc-magnetite-alr-arfcn-forced.patch

9123 octets, 235 lignes → 235 lignes

```
--- a/src/aci2/alr/alr_cs.c
+++ b/src/aci2/alr/alr_cs.c
@@ -44,6 +44,11 @@
 #include "alr_em.h"
 #include "cl_list.h"

+/* [2026-10-21] banc osmo-operator : ARFCN force par defaut (voir cs_rxlev_ind) */
+#ifndef ARFCN_FORCED
+#define ARFCN_FORCED 1
+#endif
+
 #if defined (_SIMULATION_)
 #define TRACING
 #endif
@@ -828,6 +833,32 @@
         /* Sort the channels based on their RxLev */
         cs_add_and_sort_channels ();

+#if ARFCN_FORCED >= 0
+        /* [2026-10-21] banc d'emulation osmo-operator : seule la cellule de l'ARFCN_FORCED est presentee
+         * a RR (les autres porteuses sont retirees de la liste triee ; ajoutee au niveau max si le
+         * balayage ne l'a pas vue). -DARFCN_FORCED=<n> pour une autre, -1 pour le comportement normal. */
+        {
+          U16 k, n = 0;
+          for (k = 0; k < mph_power_cnf->num_of_chan; k++)
+          {
```

```
+            if ((mph_power_cnf->arfcn[k] & 0x03FF) EQ ARFCN_FORCED)
+            {
+              mph_power_cnf->arfcn[n]  = mph_power_cnf->arfcn[k];
+              mph_power_cnf->rx_lev[n] = mph_power_cnf->rx_lev[k];
+              n++;
+            }
+          }
+          if (n EQ 0)
+          {
+            mph_power_cnf->arfcn[0]  = ARFCN_FORCED;
+            mph_power_cnf->rx_lev[0] = 63;
+            n = 1;
+          }
+          mph_power_cnf->num_of_chan = (U8)n;
+          TRACE_EVENT_P2 ("ARFCN_FORCED=%u : %u porteuse(s) presentee(s) a RR", ARFCN_FORCED, n);
+        }
+#endif
+
          if(!IS_EXT_MEAS_RUNNING)
          {
            /* Free MPH_POWER_REQ buffer */
--- a/src/ui3/bmi/mmiDummy.c
+++ b/src/ui3/bmi/mmiDummy.c
@@ -154,12 +154,34 @@
  *
  ******************************************************************************/

+/* [2026-10-21] banc osmo-operator : un handle de fenetre perime (petite valeur, fenetre detruite) faisait
+ * lire hdr->data en flash a ~0 (vecteur 0xEA003FFD) puis faute de donnees (DFAR 0xEA003FFD, LR dans
+ * mmi_send_event) ; le gestionnaire d'abort bouclait et tout le firmware gelait (ATI muet). On ne
+ * suit que des pointeurs en RAM (interne 0x800000-0x83FFFF, externe 0x1000000-0x17FFFFF). */
+#define MMI_EN_RAM(p) ((((ULONG)(p)) >= 0x00800000UL && ((ULONG)(p)) < 0x00840000UL) || \
+                       (((ULONG)(p)) >= 0x01000000UL && ((ULONG)(p)) < 0x01800000UL))
+
 void mmi_send_event (T_MFW_HND win, USHORT event, USHORT value, void * parameter)
 {

    if (win){
-       T_MFW_WIN     * win_data  = ((T_MFW_HDR *)win)->data;
-         T_MMI_CONTROL * control  = (T_MMI_CONTROL *)win_data->user;
+       T_MFW_WIN     * win_data;
+       T_MMI_CONTROL * control;
+
+       if (!MMI_EN_RAM(win)) {
+           TRACE_EVENT_P2("mmi_send_event: handle invalide 0x%08x (evt %d) ignore", (ULONG)win, event);
+           return;
+       }
+       win_data = ((T_MFW_HDR *)win)->data;
+       if (!MMI_EN_RAM(win_data)) {
+           TRACE_EVENT_P2("mmi_send_event: win_data invalide 0x%08x (evt %d) ignore", (ULONG)win_data, event);
+           return;
+       }
+       control = (T_MMI_CONTROL *)win_data->user;
+       if (!MMI_EN_RAM(control)) {
+           TRACE_EVENT_P2("mmi_send_event: control invalide 0x%08x (evt %d) ignore", (ULONG)control, event);
+           return;
+       }

        if (control->dialog NEQ 0)
            (control->dialog) (win, event, value, parameter);
--- a/src/ui3/mfw/mfw_nm.c
+++ b/src/ui3/mfw/mfw_nm.c
@@ -1259,6 +1259,40 @@
 }
```

```
+/* [2026-10-21] banc osmo-operator : enregistrement robuste aux touches.
+ * - BANC_REG_AUTO (defaut 1) : nm_registration() part toujours en automatique. Le mode manuel
+ *   memorise en FFS (/mmi/mmidata) sans operateur connu (LOCI vierge) laissait le MM sans
+ *   recherche (+CREG 0) et le MMI affichait "Network Forbidden".
+ * - nm_nrg_robuste() : si l'ACI repond AT_BUSY (une commande MM en cours, liste PLMN lancee au
+ *   clavier, +COPS d'un autre canal...), on l'abandonne et on relance une fois. */
+#ifndef BANC_REG_AUTO
+#define BANC_REG_AUTO 1
+#endif
+static U8 nm_reg_attente;   /* nm_registration() refusee faute de SIM active : a relancer */
+void nm_reg_en_attente (void)
+{
+    if (!nm_reg_attente)
+        return;
+    nm_reg_attente = 0;
+    TRACE_EVENT("nm_reg_en_attente: SIM active, enregistrement relance");
+    nm_registration(NM_AUTOMATIC, 0, 0);
+}
+
+static T_ACI_RETURN nm_nrg_robuste (T_ACI_NRG_RGMD rm, char *opr)
+{
+    T_ACI_RETURN r = sAT_PercentNRG(CMD_SRC_LCL, rm, NRG_SVMD_Full, NRG_FRMT_Numeric, opr);
+    if (r == AT_BUSY)
+    {
+        TRACE_EVENT("nm_nrg_robuste: AT_BUSY, abandon COPS/NRG et relance");
+        sAT_Abort(CMD_SRC_LCL, AT_CMD_COPS);
+        sAT_Abort(CMD_SRC_LCL, AT_CMD_NRG);
+        r = sAT_PercentNRG(CMD_SRC_LCL, rm, NRG_SVMD_Full, NRG_FRMT_Numeric, opr);
+    }
+    if (r != AT_EXCT && r != AT_CMPL)
+        TRACE_EVENT_P1("nm_nrg_robuste: sAT_PercentNRG -> %d", r);
+    return r;
+}
+
 /*
 +------------------------------------------------------------------------+
 | PROJECT : MMI-Framework (8417)        MODULE  : MFW_NM             |
@@ -1296,7 +1330,19 @@
       return IMEI_NOT_VALID;

     if (!limited && sim_stat != SIM_ACTIVE)
+    {
+      /* [2026-10-21] banc : relancee par nm_reg_en_attente() quand la SIM devient active */
+      nm_reg_attente = 1;
+      TRACE_EVENT("nm_registration: SIM pas encore active, enregistrement mis en attente");
       return SIM_NOT_ACTIVE;
+    }
+    nm_reg_attente = 0;
+
+#if BANC_REG_AUTO
+    if (mode == NM_MANUAL)
+        TRACE_EVENT("nm_registration: mode manuel memorise -> automatique (BANC_REG_AUTO)");
+    mode = NM_AUTOMATIC;
+#endif


     if (mode == NM_MANUAL)
@@ -1310,22 +1356,14 @@
                         to the last manual selected HPLMN cell before the power cycle, and this does not happen;
                           instead this, the sample shows up the available networks list
            Fix         :  call to nm_avail_plmn_list() is changed to call to sAT_PercentNRG() */
-      if (sAT_PercentNRG(CMD_SRC_LCL,
-                         NRG_RGMD_Manual,
-                         NRG_SVMD_Full,
-                         NRG_FRMT_Numeric,
-                         NULL) != AT_EXCT)
```

```
+        if (nm_nrg_robuste(NRG_RGMD_Manual, NULL) != AT_EXCT)
         {
             TRACE_ERROR("sAT_PercentNRG error");
         }
     }
     else
     {
-       if (sAT_PercentNRG(CMD_SRC_LCL,
-                          NRG_RGMD_Auto,
-                          NRG_SVMD_Full,
-                          NRG_FRMT_Numeric,
-                          NULL) != AT_EXCT)
+        if (nm_nrg_robuste(NRG_RGMD_Auto, NULL) != AT_EXCT)
         {
             TRACE_ERROR("sAT_PercentNRG error");
         }
@@ -1856,12 +1894,9 @@
     T_ACI_RETURN result;
     TRACE_FUNCTION("nm_select_plmn()");

-   result = sAT_PercentNRG(CMD_SRC_LCL,
-                       NRG_RGMD_Manual,
-                       NRG_SVMD_Full,
-                       NRG_FRMT_Numeric,
-                       (char *) network);
-   result = AT_FAIL;
+   /* [2026-10-21] banc : le "result = AT_FAIL;" d'origine qui suivait cet appel faisait
+    * tracer une erreur (et tester un qAT_ErrDesc perime) a chaque choix de reseau au clavier. */
+   result = nm_nrg_robuste(NRG_RGMD_Manual, (char *) network);
     if (result != AT_EXCT)
     {
         TRACE_ERROR("sAT_PercentNRG error");
--- a/src/ui3/mfw/mfw_sim.c
+++ b/src/ui3/mfw/mfw_sim.c
@@ -656,8 +656,21 @@

 */

+/* [2026-10-21] banc osmo-operator : course au boot. nm_registration() appelee avant que la SIM
+ * soit active rendait SIM_NOT_ACTIVE sans rien lancer, et personne ne la rappelait (+CREG 0,0,
+ * pas de RACH, "Network Forbidden"). Chaque changement d etat SIM passe par sim_signal() : on
+ * y relance l enregistrement en attente (mfw_nm.c, nm_reg_en_attente) quand la SIM devient active. */
+static void sim_signal_mmi (MfwEvt event, void *para);
+EXTERN void nm_reg_en_attente (void);
 void sim_signal (MfwEvt event, void *para)
 {
+    sim_signal_mmi(event, para);
+    if (sim_stat == SIM_ACTIVE)
+        nm_reg_en_attente();
+}
+
+static void sim_signal_mmi (MfwEvt event, void *para)
+{
   UBYTE temp = dspl_Enable(0);
     TRACE_FUNCTION ("sim_signal()");

--- a/src/cs/layer1/dyn_dwl_cfile/l1_dyn_dwl_async.c
+++ b/src/cs/layer1/dyn_dwl_cfile/l1_dyn_dwl_async.c
@@ -182,6 +182,18 @@
   UWORD16 temp_patch_id[MAX_NUM_OF_PATCH_IDS];
   BOOL return_flag = FALSE;

+  /* [2026-10-21] banc osmo-operator (BANC_SANS_DYN_DWNLD, defaut 1) : aucun telechargement dynamique. Le DSP
+   * emule n a pas le patch TI qui sert ce protocole : les pointeurs d_api_dwl_* du NDB etaient faux et la copie
+   * du patch (l1_dyn_dwl_apihisr.c) ecrasait la table des vecteurs ARM a 0x0 a l entree en dedie (D_DWL_C puis
+   * abort en boucle). Sans patch charge (DWNLD 0) il n y a rien a completer. L1_DYN_DSP_DWNLD 0 ne compile pas
```

```
+   * (variante audio AMR sans en-tetes). */
+  #ifndef BANC_SANS_DYN_DWNLD
+  #define BANC_SANS_DYN_DWNLD 1
+  #endif
+  #if BANC_SANS_DYN_DWNLD
+  return FALSE;
+  #endif
+
   // Primitive is processed only if it triggers a dynamic download or there is a delay
   if(delay_flag == TRUE || l1_does_the_incoming_primitive_trigger_dynamic_dwnld(SignalCode, FALSE) == TRUE)
   {
```

3.276 /opt/GSM/osmo-operator/patches/fc-magnetite-banc.patch

14892 octets, 368 lignes → 368 lignes

```
diff -r 9394305d4ff5 src/aci2/alr/alr_cs.c
--- a/src/aci2/alr/alr_cs.c Mon Oct 31 00:14:44 2022 +0000
+++ b/src/aci2/alr/alr_cs.c Fri Oct 23 19:18:45 2026 +0200
@@ -44,6 +44,11 @@
 #include "alr_em.h"
 #include "cl_list.h"

+/* [2026-10-21] banc osmo-operator : ARFCN force par defaut (voir cs_rxlev_ind) */
+#ifndef ARFCN_FORCED
+#define ARFCN_FORCED 1
+#endif
+
 #if defined (_SIMULATION_)
 #define TRACING
 #endif
@@ -828,6 +833,32 @@
         /* Sort the channels based on their RxLev */
         cs_add_and_sort_channels ();

+#if ARFCN_FORCED >= 0
+        /* [2026-10-21] banc d'emulation osmo-operator : seule la cellule de l'ARFCN_FORCED est presentee
+         * a RR (les autres porteuses sont retirees de la liste triee ; ajoutee au niveau max si le
+         * balayage ne l'a pas vue). -DARFCN_FORCED=<n> pour une autre, -1 pour le comportement normal. */
+        {
+          U16 k, n = 0;
+          for (k = 0; k < mph_power_cnf->num_of_chan; k++)
+          {
+            if ((mph_power_cnf->arfcn[k] & 0x03FF) EQ ARFCN_FORCED)
+            {
+              mph_power_cnf->arfcn[n]  = mph_power_cnf->arfcn[k];
+              mph_power_cnf->rx_lev[n] = mph_power_cnf->rx_lev[k];
+              n++;
+            }
+          }
+          if (n EQ 0)
+          {
+            mph_power_cnf->arfcn[0]  = ARFCN_FORCED;
+            mph_power_cnf->rx_lev[0] = 63;
+            n = 1;
+          }
+          mph_power_cnf->num_of_chan = (U8)n;
+          TRACE_EVENT_P2 ("ARFCN_FORCED=%u : %u porteuse(s) presentee(s) a RR", ARFCN_FORCED, n);
+        }
+#endif
+
         if(!IS_EXT_MEAS_RUNNING)
         {
           /* Free MPH_POWER_REQ buffer */
```

```
diff -r 9394305d4ff5 src/cs/layer1/dyn_dwl_cfile/l1_dyn_dwl_async.c
--- a/src/cs/layer1/dyn_dwl_cfile/l1_dyn_dwl_async.c    Mon Oct 31 00:14:44 2022 +0000
+++ b/src/cs/layer1/dyn_dwl_cfile/l1_dyn_dwl_async.c    Fri Oct 23 19:18:45 2026 +0200
@@ -182,6 +182,18 @@
   UWORD16 temp_patch_id[MAX_NUM_OF_PATCH_IDS];
   BOOL return_flag = FALSE;

+  /* [2026-10-21] banc osmo-operator (BANC_SANS_DYN_DWNLD, defaut 1) : aucun telechargement dynamique. Le DSP
+   * emule n a pas le patch TI qui sert ce protocole : les pointeurs d_api_dwl_* du NDB etaient faux et la copie
+   * du patch (l1_dyn_dwl_apihisr.c) ecrasait la table des vecteurs ARM a 0x0 a l entree en dedie (D_DWL_C puis
+   * abort en boucle). Sans patch charge (DWNLD 0) il n y a rien a completer. L1_DYN_DSP_DWNLD 0 ne compile pas
+   * (variante audio AMR sans en-tetes). */
+  #ifndef BANC_SANS_DYN_DWNLD
+  #define BANC_SANS_DYN_DWNLD 1
+  #endif
+  #if BANC_SANS_DYN_DWNLD
+  return FALSE;
+  #endif
+
   // Primitive is processed only if it triggers a dynamic download or there is a delay
   if(delay_flag == TRUE || l1_does_the_incoming_primitive_trigger_dynamic_dwnld(SignalCode, FALSE) == TRUE)
   {
diff -r 9394305d4ff5 src/ui3/bmi/mmiDummy.c
--- a/src/ui3/bmi/mmiDummy.c    Mon Oct 31 00:14:44 2022 +0000
+++ b/src/ui3/bmi/mmiDummy.c    Fri Oct 23 19:18:45 2026 +0200
@@ -154,12 +154,34 @@
  *
  ******************************************************************************************/

+/* [2026-10-21] banc osmo-operator : un handle de fenetre perime (petite valeur, fenetre detruite) faisait
+ * lire hdr->data en flash a ~0 (vecteur 0xEA003FFD) puis faute de donnees (DFAR 0xEA003FFD, LR dans
+ * mmi_send_event) ; le gestionnaire d'abort bouclait et tout le firmware gelait (ATI muet). On ne
+ * suit que des pointeurs en RAM (interne 0x800000-0x83FFFF, externe 0x1000000-0x17FFFFF). */
+#define MMI_EN_RAM(p) ((((ULONG)(p)) >= 0x00800000UL && ((ULONG)(p)) < 0x00840000UL) || \
+                       (((ULONG)(p)) >= 0x01000000UL && ((ULONG)(p)) < 0x01800000UL))
+
 void mmi_send_event (T_MFW_HND win, USHORT event, USHORT value, void * parameter)
 {

    if (win){
-       T_MFW_WIN     * win_data  = ((T_MFW_HDR *)win)->data;
-         T_MMI_CONTROL * control  = (T_MMI_CONTROL *)win_data->user;
+       T_MFW_WIN     * win_data;
+       T_MMI_CONTROL * control;
+
+       if (!MMI_EN_RAM(win)) {
+           TRACE_EVENT_P2("mmi_send_event: handle invalide 0x%08x (evt %d) ignore", (ULONG)win, event);
+           return;
+       }
+       win_data = ((T_MFW_HDR *)win)->data;
+       if (!MMI_EN_RAM(win_data)) {
+           TRACE_EVENT_P2("mmi_send_event: win_data invalide 0x%08x (evt %d) ignore", (ULONG)win_data, event);
+           return;
+       }
+       control = (T_MMI_CONTROL *)win_data->user;
+       if (!MMI_EN_RAM(control)) {
+           TRACE_EVENT_P2("mmi_send_event: control invalide 0x%08x (evt %d) ignore", (ULONG)control, event);
+           return;
+       }

        if (control->dialog NEQ 0)
            (control->dialog) (win, event, value, parameter);
diff -r 9394305d4ff5 src/ui3/bmi/mmiSmsRead.c
--- a/src/ui3/bmi/mmiSmsRead.c  Mon Oct 31 00:14:44 2022 +0000
+++ b/src/ui3/bmi/mmiSmsRead.c  Fri Oct 23 19:18:45 2026 +0200
@@ -777,6 +777,11 @@
```

```
            {
                for (i = 0; i < ((data->sms_list_data).nb_sms_stored); i++)
                    {
+#ifdef FF_MMI_SMS_DYNAMIC
+                   /* [banc 2026-10-22] deja libere par E_EXIT, ou jamais alloue (liste vide) */
+                   if (data->sms_list_data.sms_header == NULL || data->sms_list_data.sms_header[i] == NULL)
+                       continue;
+#endif
                    FREE_MEMORY( (void *)data->sms_list_data.sms_header[i], UCS2_MENU_TEXT_LEN+ 1/*SPR1242*/);
                    }
                /*SPR 2686, only de-allocate memory for the number of messages displayed*/
@@ -784,8 +789,11 @@
                FREE_MEMORY ((UBYTE *)data->menu_list_data, sizeof(ListMenuData));
            }
 #ifdef FF_MMI_SMS_DYNAMIC
-           FREE_MEMORY((void*)data->sms_list_data.sms_messages,g_max_messages * sizeof(T_MFW_SMS_MSG) );
-           FREE_MEMORY((void*)data->sms_list_data.sms_header, g_max_messages * sizeof(wstring_t));
+           /* [banc 2026-10-22] gardes : E_EXIT a pu les liberer deja (remis a NULL) */
+           if (data->sms_list_data.sms_messages != NULL)
+               FREE_MEMORY((void*)data->sms_list_data.sms_messages,g_max_messages * sizeof(T_MFW_SMS_MSG) );
+           if (data->sms_list_data.sms_header != NULL)
+               FREE_MEMORY((void*)data->sms_list_data.sms_header, g_max_messages * sizeof(wstring_t));
 #endif
            FREE_MEMORY ((void *)data, sizeof (T_SMSREAD_R));
            sms_read_win = 0;
@@ -1012,18 +1020,32 @@
        l_parent = data->parent;
        SmsRead_R_destroy(win);
        SEND_EVENT(l_parent, E_RETURN, data->id, NULL);
-
+       /* [banc 2026-10-22] break ajoute : E_EXIT relisait data/list_data deja liberes par SmsRead_R_destroy */
+       break;
+
    case E_EXIT: // Terminate exit from sms send...
    // May 27, 2005    MMI-FIX-29869 x0018858
 #ifdef FF_MMI_SMS_DYNAMIC
     {
           int i;
-       for (i=0;i<list_data->nb_sms_stored; i++)
-           {
-               FREE_MEMORY((U8*)list_data->sms_header[i], UCS2_MENU_TEXT_LEN + 1 );
-           }
-       FREE_MEMORY((U8*)list_data->sms_header, g_max_messages * sizeof(wstring_t) );
-       FREE_MEMORY((U8*)list_data->sms_messages, g_max_messages * sizeof(T_MFW_SMS_MSG));
+       /* [banc 2026-10-22] gardes + NULL : sans eux SmsRead_R_destroy liberait une 2e fois (tas MMI
+          corrompu,  Preceeding Sentinal Bytes have been corrupted ), et sms_header n'existe pas si la
+          liste est vide */
+       if (list_data->sms_header != NULL)
+       {
+           for (i=0;i<list_data->nb_sms_stored; i++)
+               {
+                   if (list_data->sms_header[i] != NULL)
+                       FREE_MEMORY((U8*)list_data->sms_header[i], UCS2_MENU_TEXT_LEN + 1 );
+                   list_data->sms_header[i] = NULL;
+               }
+           FREE_MEMORY((U8*)list_data->sms_header, g_max_messages * sizeof(wstring_t) );
+           list_data->sms_header = NULL;
+       }
+       if (list_data->sms_messages != NULL)
+           FREE_MEMORY((U8*)list_data->sms_messages, g_max_messages * sizeof(T_MFW_SMS_MSG));
+       list_data->sms_messages = NULL;
+       list_data->nb_sms_stored = 0;
    }
 #endif
        listDisplayListMenu(win, data->menu_list_data, (ListCbFunc_t)SmsRead_R_smslist_cb,0);
diff -r 9394305d4ff5 src/ui3/mfw/mfw_nm.c
```

```
--- a/src/ui3/mfw/mfw_nm.c  Mon Oct 31 00:14:44 2022 +0000
+++ b/src/ui3/mfw/mfw_nm.c  Fri Oct 23 19:18:45 2026 +0200
@@ -1259,6 +1259,40 @@
 }


+/* [2026-10-21] banc osmo-operator : enregistrement robuste aux touches.
+ * - BANC_REG_AUTO (defaut 1) : nm_registration() part toujours en automatique. Le mode manuel
+ *   memorise en FFS (/mmi/mmidata) sans operateur connu (LOCI vierge) laissait le MM sans
+ *   recherche (+CREG 0) et le MMI affichait "Network Forbidden".
+ * - nm_nrg_robuste() : si l'ACI repond AT_BUSY (une commande MM en cours, liste PLMN lancee au
+ *   clavier, +COPS d'un autre canal...), on l'abandonne et on relance une fois. */
+#ifndef BANC_REG_AUTO
+#define BANC_REG_AUTO 1
+#endif
+static U8 nm_reg_attente;   /* nm_registration() refusee faute de SIM active : a relancer */
+void nm_reg_en_attente (void)
+{
+    if (!nm_reg_attente)
+        return;
+    nm_reg_attente = 0;
+    TRACE_EVENT("nm_reg_en_attente: SIM active, enregistrement relance");
+    nm_registration(NM_AUTOMATIC, 0, 0);
+}
+
+static T_ACI_RETURN nm_nrg_robuste (T_ACI_NRG_RGMD rm, char *opr)
+{
+    T_ACI_RETURN r = sAT_PercentNRG(CMD_SRC_LCL, rm, NRG_SVMD_Full, NRG_FRMT_Numeric, opr);
+    if (r == AT_BUSY)
+    {
+        TRACE_EVENT("nm_nrg_robuste: AT_BUSY, abandon COPS/NRG et relance");
+        sAT_Abort(CMD_SRC_LCL, AT_CMD_COPS);
+        sAT_Abort(CMD_SRC_LCL, AT_CMD_NRG);
+        r = sAT_PercentNRG(CMD_SRC_LCL, rm, NRG_SVMD_Full, NRG_FRMT_Numeric, opr);
+    }
+    if (r != AT_EXCT && r != AT_CMPL)
+        TRACE_EVENT_P1("nm_nrg_robuste: sAT_PercentNRG -> %d", r);
+    return r;
+}
+
 /*
  +--------------------------------------------------------------------+
  | PROJECT : MMI-Framework (8417)        MODULE  : MFW_NM             |
@@ -1296,7 +1330,19 @@
       return IMEI_NOT_VALID;

     if (!limited && sim_stat != SIM_ACTIVE)
+    {
+      /* [2026-10-21] banc : relancee par nm_reg_en_attente() quand la SIM devient active */
+      nm_reg_attente = 1;
+      TRACE_EVENT("nm_registration: SIM pas encore active, enregistrement mis en attente");
       return SIM_NOT_ACTIVE;
+    }
+    nm_reg_attente = 0;
+
+#if BANC_REG_AUTO
+    if (mode == NM_MANUAL)
+        TRACE_EVENT("nm_registration: mode manuel memorise -> automatique (BANC_REG_AUTO)");
+    mode = NM_AUTOMATIC;
+#endif


     if (mode == NM_MANUAL)
@@ -1310,22 +1356,14 @@
                         to the last manual selected HPLMN cell before the power cycle, and this does not happen;
                           instead this, the sample shows up the available networks list
```

```
            Fix         :  call to nm_avail_plmn_list() is changed to call to sAT_PercentNRG() */
-       if (sAT_PercentNRG(CMD_SRC_LCL,
-                          NRG_RGMD_Manual,
-                          NRG_SVMD_Full,
-                          NRG_FRMT_Numeric,
-                          NULL) != AT_EXCT)
+        if (nm_nrg_robuste(NRG_RGMD_Manual, NULL) != AT_EXCT)
         {
             TRACE_ERROR("sAT_PercentNRG error");
         }
     }
     else
     {
-       if (sAT_PercentNRG(CMD_SRC_LCL,
-                          NRG_RGMD_Auto,
-                          NRG_SVMD_Full,
-                          NRG_FRMT_Numeric,
-                          NULL) != AT_EXCT)
+        if (nm_nrg_robuste(NRG_RGMD_Auto, NULL) != AT_EXCT)
         {
             TRACE_ERROR("sAT_PercentNRG error");
         }
@@ -1856,12 +1894,9 @@
     T_ACI_RETURN result;
     TRACE_FUNCTION("nm_select_plmn()");

-   result = sAT_PercentNRG(CMD_SRC_LCL,
-                       NRG_RGMD_Manual,
-                       NRG_SVMD_Full,
-                       NRG_FRMT_Numeric,
-                       (char *) network);
-   result = AT_FAIL;
+   /* [2026-10-21] banc : le "result = AT_FAIL;" d'origine qui suivait cet appel faisait
+    * tracer une erreur (et tester un qAT_ErrDesc perime) a chaque choix de reseau au clavier. */
+   result = nm_nrg_robuste(NRG_RGMD_Manual, (char *) network);
     if (result != AT_EXCT)
     {
         TRACE_ERROR("sAT_PercentNRG error");
diff -r 9394305d4ff5 src/ui3/mfw/mfw_sim.c
--- a/src/ui3/mfw/mfw_sim.c Mon Oct 31 00:14:44 2022 +0000
+++ b/src/ui3/mfw/mfw_sim.c Fri Oct 23 19:18:45 2026 +0200
@@ -656,8 +656,21 @@

 */

+/* [2026-10-21] banc osmo-operator : course au boot. nm_registration() appelee avant que la SIM
+ * soit active rendait SIM_NOT_ACTIVE sans rien lancer, et personne ne la rappelait (+CREG 0,0,
+ * pas de RACH, "Network Forbidden"). Chaque changement d etat SIM passe par sim_signal() : on
+ * y relance l enregistrement en attente (mfw_nm.c, nm_reg_en_attente) quand la SIM devient active. */
+static void sim_signal_mmi (MfwEvt event, void *para);
+EXTERN void nm_reg_en_attente (void);
 void sim_signal (MfwEvt event, void *para)
 {
+    sim_signal_mmi(event, para);
+    if (sim_stat == SIM_ACTIVE)
+        nm_reg_en_attente();
+}
+
+static void sim_signal_mmi (MfwEvt event, void *para)
+{
   UBYTE temp = dspl_Enable(0);
     TRACE_FUNCTION ("sim_signal()");

diff -r 9394305d4ff5 src/ui3/mfw/mfw_sms.c
--- a/src/ui3/mfw/mfw_sms.c Mon Oct 31 00:14:44 2022 +0000
+++ b/src/ui3/mfw/mfw_sms.c Fri Oct 23 19:18:45 2026 +0200
```

```
@@ -241,6 +241,8 @@

 //May 27, 2005    MMI-FIX-29869 x0018858 - Added a Checking variable.
 static int Counter_first = 0;
+/* [banc 2026-10-22] sms_read_msg_info() appele avant l'allocation de msg_info (voir Check_max_sms_allowed) */
+static int liste_avant_alloc = 0;
 #else
 static T_MFW_SMS_MSG msg_info[MAX_MESSAGES]; /* message table */
 #endif
@@ -348,8 +350,30 @@
                 {
                     msg_info = (T_MFW_SMS_MSG*)ALLOC_MEMORY(g_max_messages*sizeof(T_MFW_SMS_MSG));

+                    /* [banc 2026-10-22] Le tableau sortait de ALLOC_MEMORY NON initialise. rAT_sms_ready() ->
+                     sms_read_msg_info() passe souvent AVANT cette allocation (premier evenement SMS de l'ecran de
+                     veille) : sa remise a index -1 tombait sur msg_info NULL et le tableau alloue ensuite gardait
+                     des dechets. Index != -1 = entrees fantomes  [P]  dans Messages > Read ; en ouvrir une lancait
+                       sms_read sur un index bidon, sans reponse :  Please wait  infini. */
+                    for (i=0; i<g_max_messages; i++)
+                    {
+                        memset(&msg_info[i], 0, sizeof(T_MFW_SMS_MSG));
+                        msg_info[i].index = -1;
+                        msg_info[i].first_index = -1;
+                        msg_info[i].addr.ton = MFW_TON_UNKNOWN;
+                        msg_info[i].addr.npi = MFW_NPI_UNKNOWN;
+                        msg_info[i].concat_status = MFW_SMS_NO_CONC;
+                        msg_info[i].mem = MFW_MEM_NOTPRESENT;
+                        msg_info[i].first_mem = MFW_MEM_NOTPRESENT;
+                    }
                 }
             Counter_first ++;
+           /* la liste lue avant l'allocation est perdue : la relire maintenant */
+           if (liste_avant_alloc)
+           {
+               liste_avant_alloc = 0;
+               sms_read_msg_info();
+           }
         }

 }
@@ -853,6 +877,15 @@
     TRACE_FUNCTION("sms_read_msg_info()");

 #ifdef FF_MMI_SMS_DYNAMIC
+    if (msg_info == NULL)
+    {
+        /* [banc 2026-10-22] trop tot : Check_max_sms_allowed() relira la liste apres l'allocation */
+        liste_avant_alloc = 1;
+        return;
+    }
+#endif
+
+#ifdef FF_MMI_SMS_DYNAMIC
     for (i=0; i<g_max_messages; i++)
 #else
     for (i=0; i<MAX_MESSAGES; i++)
```

3.277 /opt/GSM/osmo-operator/patches/gapk-pq-alsa-output-probe.patch

8936 octets, 221 lignes → 221 lignes

```
diff --git a/src/pq_alsa.c b/src/pq_alsa.c
index 50479e6..05efbda 100644
--- a/src/pq_alsa.c
```

```
+++ b/src/pq_alsa.c
@@ -20,6 +20,7 @@

 #include <errno.h>
 #include <stdint.h>
+#include <stdlib.h>    /* getenv() — vanne GAPK_ALSA_PROBE */
 #include <talloc.h>

 #include <osmocom/gapk/logging.h>
@@ -51,14 +52,96 @@ pq_cb_alsa_input(void *_state, uint8_t *out, const uint8_t *in, unsigned int in_
     return rv * sizeof(uint16_t);
 }

+/* SONDE GAPK_ALSA_PROBE — le PCM qui part vers ALSA est-il muet ?
+ *
+ * Pourquoi elle existe (mesures du 2026-08-09, Calypso emule sous QEMU) : le
+ * mobile recoit a sa porte 300/300 trames FR DISTINCTES, cadrage 49 o et
+ * signature 0xd correctes (releve dans osmocon.log, hdlc_recv dlci=5), la meme
+ * libosmogapk decode ces memes trames en audio reel quand on l'appelle par le
+ * binaire osmo-gapk autonome (RMS 391, 98,2 % d'echantillons non nuls), et
+ * pourtant la sortie ALSA du mobile est du ZERO NUMERIQUE EXACT (crete 0 sur
+ * 48000 echantillons). Aucune erreur n'est journalisee, car
+ * osmo_gapk_pq_execute ne se plaint que d'un retour negatif.
+ *
+ * Cette sonde tranche en un appel : le PCM remis a snd_pcm_writei est-il nul
+ * (le defaut est en amont, dans la chaine) ou non nul (le defaut est en aval,
+ * dans ALSA/PulseAudio) ?
+ *
+ * Compteurs CUMULATIFS, jamais un taux : un taux tire d'un journal bufferise a
+ * deja produit deux chiffres faux dans cette enquete. La sonde s'annonce au
+ * premier passage dans les deux cas — une sonde muette rend le silence
+ * indecidable. Diagnostic pur : aucun effet sur le flux quand elle est eteinte.
+ */
+static void pq_alsa_out_probe(void *_state, const uint8_t *in, unsigned int in_len,
+                              int rv, int rv_avant_reprise)
+{
+   struct pq_state_alsa *state = _state;
+   static int on = -1;
+   static unsigned long long n, muettes, ecrites_court, epipe, autres_err;
+   static unsigned long long pic_max;
+   unsigned int i;
+   unsigned long long pic = 0;
+   snd_pcm_state_t st;
+   snd_pcm_sframes_t dispo, retard = 0;
+
+   if (on < 0) {
+       const char *e = getenv("GAPK_ALSA_PROBE");
+       on = (e && *e == '1') ? 1 : 0;
+       LOGPGAPK(LOGL_NOTICE, "SONDE GAPK_ALSA_PROBE %s\n",
+                on ? "ACTIVE" : "inactive");
+   }
+   if (!on)
+       return;
+
+   for (i = 0; i + 1 < in_len; i += 2) {
+       int16_t s = (int16_t)(in[i] | (in[i + 1] << 8));
+       unsigned long long a = (s < 0) ? (unsigned long long)-(long long)s : (unsigned long long)s;
+       if (a > pic)
+           pic = a;
+   }
+   n++;
+   if (pic == 0)
+       muettes++;
+   if (pic > pic_max)
+       pic_max = pic;
+   if (rv >= 0 && (unsigned int)rv != in_len / 2)
```

```
+       ecrites_court++;
+   /* L'ECHEC EST RECUPERE EN SILENCE par pq_cb_alsa_output : sur -EPIPE il
+    * fait snd_pcm_prepare() puis reecrit, et ne renvoie que le rv d'apres.
+    * Un flux qui sous-alimente en permanence boucle donc XRUN -> prepare ->
+    * write sans que rien ne le dise, et PulseAudio n'a jamais rien en file
+    * (symptome observe : « Buffer Latency: 0 usec » constant). C'est ce que
+    * ce compteur rend visible. */
+   if (rv_avant_reprise == -EPIPE)
+       epipe++;
+   else if (rv_avant_reprise < 0)
+       autres_err++;
+
+   st = snd_pcm_state(state->pcm_handle);
+   dispo = snd_pcm_avail_update(state->pcm_handle);
+   if (snd_pcm_delay(state->pcm_handle, &retard) < 0)
+       retard = -1;
+
+   if (n <= 3 || (n % 250) == 0)
+       LOGPGAPK(LOGL_NOTICE, "GAPK-ALSA-PROBE : %llu blocs, in_len=%u "
+            "(%u ech), rendus par ALSA=%d -- blocs MUETS : %llu, "
+            "ecritures courtes : %llu, XRUN recuperes : %llu, "
+            "autres erreurs : %llu, crete max vue : %llu, "
+            "etat=%s avail=%ld delay=%ld\n",
+            n, in_len, in_len / 2, rv, muettes, ecrites_court,
+            epipe, autres_err, pic_max,
+            snd_pcm_state_name(st), (long)dispo, (long)retard);
+}
+
 static int
 pq_cb_alsa_output(void *_state, uint8_t *out, const uint8_t *in, unsigned int in_len)
 {
    struct pq_state_alsa *state = _state;
    unsigned int num_samples = in_len/2;
-   int rv;
+   int rv, rv_avant_reprise;

    rv = snd_pcm_writei(state->pcm_handle, in, num_samples);
+   rv_avant_reprise = rv;
    if (rv == -EPIPE) {
        /* Recover from buffer underrun */
        snd_pcm_prepare(state->pcm_handle);
@@ -66,6 +149,8 @@ pq_cb_alsa_output(void *_state, uint8_t *out, const uint8_t *in, unsigned int in
        rv = snd_pcm_writei(state->pcm_handle, in, num_samples);
    }

+   pq_alsa_out_probe(state, in, in_len, rv, rv_avant_reprise);
+
    return rv == num_samples ? 0 : -1;
 }

@@ -129,12 +214,101 @@ pq_queue_alsa_op(struct osmo_gapk_pq *pq, const char *alsa_dev, unsigned int blk
    if (rc < 0)
        goto out_free_par;

+   /* DIMENSIONNEMENT DE LA PERIODE ET DU TAMPON — sans ça, l'ecouteur est MUET.
+    *
+    * Ces deux appels manquaient. snd_pcm_hw_params() prenait donc le maximum
+    * offert par le greffon ALSA-pulse, mesure dans le conteneur :
+    *   HW  period_size=2048  buffer_size=2097152  periods=1024   (a 8 kHz :
+    *       une periode de 256 ms, un tampon de 262 SECONDES)
+    *   SW  start_threshold=1  avail_min=2048
+    * Or la voix arrive par blocs de 160 trames = 20 ms. Le flux demarre des le
+    * premier echantillon (start_threshold=1), sous-alimente bien avant que
+    * pulse ne tire une periode complete, ALSA declare XRUN, et la reprise
+    * silencieuse de pq_cb_alsa_output appelle snd_pcm_prepare() — qui sur ce
+    * greffon fait pa_stream_flush() et JETTE ce qui etait en file.
```

```
+    *
+    * MESURE IN VIVO (archive 20260809-030334, appel ETABLI de bout en bout :
+    * premier bloc a 02:54:17, instant exact de CALL_DELIVERED -> ACTIVE) :
+    *   24250 blocs, XRUN recuperes : 24249  =  99,996 %
+    *   crete max vue : 15432   -> c'etait bien de la voix
+    *   delay epingle a ~155 trames = un seul bloc jamais consomme
+    * Cote sortie : parec sur gsm_audio.monitor rendait un ZERO EXACT.
+    *
+    * ⚠ La variable discriminante immediate est la CADENCE, pas la taille :
+    * a 20 ms/bloc on mesure 0 XRUN, a 28 ms/bloc on en mesure 99,7 %. Le
+    * defaut du dimensionnement ne CAUSE pas l'XRUN, il ne laisse AUCUNE MARGE
+    * — la moindre gigue d'arrivee d'une trame DL suffit alors a tout jeter.
+    * Une periode d'un bloc et huit periodes de tampon donnent 160 ms de coussin.
+    *
+    * Mesure apres correctif, meme banc : 0,1 % -> 73,0 % d'echantillons non
+    * nuls sur gsm_audio.monitor, crete 20000 = exactement l'amplitude injectee.
+    *
+    * GAPK_ALSA_NO_SIZING=1 retablit l'ancien comportement (aucun dimensionnement)
+    * pour refaire l'A/B sans reconstruire. Ce n'est PAS une bequille : le
+    * correctif est le defaut, la vanne n'existe que pour la mesure. */
+   {
+       const char *no_sz = getenv("GAPK_ALSA_NO_SIZING");
+       if (!(no_sz && *no_sz == '1')) {
+           snd_pcm_uframes_t period = blk_len / 2; /* blk_len en octets -> trames */
+           snd_pcm_uframes_t buffer = period * 8;  /* 8 blocs = 160 ms de marge */
+           int dir = 0;
+
+           if (period == 0)
+               period = 160;
+           rc = snd_pcm_hw_params_set_period_size_near(state->pcm_handle,
+                                hw_params, &period, &dir);
+           if (rc < 0)
+               goto out_free_par;
+           buffer = period * 8;
+           rc = snd_pcm_hw_params_set_buffer_size_near(state->pcm_handle,
+                                hw_params, &buffer);
+           if (rc < 0)
+               goto out_free_par;
+       }
+   }
+
    rc = snd_pcm_hw_params(state->pcm_handle, hw_params);
    if (rc < 0)
        goto out_free_par;

    snd_pcm_hw_params_free(hw_params);

+   /* SW params : demarrer seulement quand la moitie du tampon est remplie, pour
+    * que le puits ne parte pas en famine des le premier bloc. Sans ce reglage,
+    * start_threshold vaut 1 trame (defaut pose par snd_pcm_hw_params) et le flux
+    * demarre a vide. Non fatal : si ca echoue on continue avec les defauts, mais
+    * on le DIT — une degradation silencieuse rendrait le diagnostic impossible. */
+   {
+       const char *no_sz = getenv("GAPK_ALSA_NO_SIZING");
+       if (!(no_sz && *no_sz == '1')) {
+           snd_pcm_sw_params_t *sw = NULL;
+           snd_pcm_uframes_t period = 0, buffer = 0;
+
+           if (snd_pcm_get_params(state->pcm_handle, &buffer, &period) == 0 &&
+               snd_pcm_sw_params_malloc(&sw) == 0) {
+               if (snd_pcm_sw_params_current(state->pcm_handle, sw) == 0) {
+                   snd_pcm_sw_params_set_start_threshold(state->pcm_handle,
+                                        sw, buffer / 2);
+                   snd_pcm_sw_params_set_avail_min(state->pcm_handle,
+                                    sw, period);
+                   if (snd_pcm_sw_params(state->pcm_handle, sw) < 0)
```

```
+                       LOGPGAPK(LOGL_ERROR, "ALSA '%s': sw_params refuses, "
+                            "on garde les defauts (start_threshold=1)\n",
+                            alsa_dev);
+                }
+                snd_pcm_sw_params_free(sw);
+            }
+            LOGPGAPK(LOGL_NOTICE, "ALSA '%s': periode=%lu trames tampon=%lu trames "
+                 "(%lu ms de marge a 8 kHz)\n", alsa_dev,
+                 (unsigned long)period, (unsigned long)buffer,
+                 (unsigned long)(buffer / 8));
+        } else {
+            LOGPGAPK(LOGL_NOTICE, "ALSA '%s': GAPK_ALSA_NO_SIZING=1 -- "
+                 "dimensionnement laisse au greffon (ancien comportement)\n",
+                 alsa_dev);
+        }
+    }
+
     item = osmo_gapk_pq_add_item(pq);
     if (!item) {
         rc = -ENOMEM;
```

3.278 /opt/GSM/osmo-operator/patches/grgsm-optional-qtgui-import.patch

729 octets, 16 lignes → 16 lignes

```
--- a/python/__init__.py
+++ b/python/__init__.py
@@ -54,7 +54,11 @@
     #from fcch_burst_tagger import fcch_burst_tagger
     #from sch_detector import sch_detector
     #from fcch_detector import fcch_detector
     from .clock_offset_corrector_tagged import clock_offset_corrector_tagged
-    from .multiarfcns_receiver import multiarfcns_receiver
+    try:
+        # multiarfcns_receiver tire qtgui (GUI Qt) - optionnel, inutile au decodage
+        from .multiarfcns_receiver import multiarfcns_receiver
+    except ImportError:
+        pass
     from .gsm_input import gsm_input
     from .gsm_bcch_ccch_demapper import gsm_bcch_ccch_demapper
     from .gsm_bcch_ccch_sdcch4_demapper import gsm_bcch_ccch_sdcch4_demapper
```

3.279 /opt/GSM/osmo-operator/patches/grgsm-receiver-publish-bsic-fn.patch

1993 octets, 36 lignes → 36 lignes

```
diff --git a/lib/receiver/receiver_impl.cc b/lib/receiver/receiver_impl.cc
index 558e276..e0269d6 100644
--- a/lib/receiver/receiver_impl.cc
+++ b/lib/receiver/receiver_impl.cc
@@ -377,6 +377,31 @@ namespace gr
           send_burst(d_burst_nr, output_binary,
             GSMTAP_BURST_SCH, input_nr, d_c0_burst_start);

+          /* gr-gsm = le DSP : publier le BSIC/FN REELS decodes du SCH sur le
+           * port `measurements` (le "DSP" poste son resultat de sync, comme un
+           * vrai DSP ecrit dans le NDB). Consomme par tools/grgsm_relay_decode.py ->
+           * shunt -> shunt_dispatch_sb (remplace SHUNT_CANNED_BSIC 63). */
+          {
+            int bsic = ((ncc & 0x7) << 3) | (bcc & 0x7);
+            /* TOA REEL mesure : residu de timing du burst SCH (d_c0_burst_start
+             * vs position nominale GUARD_PERIOD), en qbits @OSR (1 sample OSR =
```

```
+             * 1 qbit), recale sur la base firmware 23 ("on time"). gr-gsm aligne
+             * bien -> ~23. Le shunt l'utilise si TOA hors CALYPSO_CANNED. */
+            int toa = 23 + (int)(d_c0_burst_start - (long)floor((GUARD_PERIOD) * d_OSR));
+            pmt::pmt_t msg = pmt::make_tuple(pmt::mp("sch"),
+              pmt::from_long(bsic),
+              pmt::from_long((long)d_burst_nr.get_frame_nr()),
+              pmt::from_long((long)toa));
+            message_port_pub(pmt::mp("measurements"), msg);
+            /* Et sur stdout : le CLI grgsm_decode avale le port measurements,
+             * donc on imprime aussi le SCH pour que si_bridge.py (qui parse le
+             * stdout de grgsm_decode) le forwarde au shunt (UDP 4731 -> feed_sb).
+             * Prefixe distinctif "SCHBSIC " (ne collisionne pas avec les hex SI). */
+            printf("SCHBSIC %d %u %d\n", bsic,
+                   (unsigned)d_burst_nr.get_frame_nr(), toa);
+            fflush(stdout);
+          }
+
           /**
            * Decoding was successful, now
            * compute offset from burst_start,
```

3.280 /opt/GSM/osmo-operator/patches/magnetite-parole-ul-au-go.patch

1521 octets, 28 lignes → 28 lignes

```
--- a/c54x_exe/src/montant.c
+++ b/c54x_exe/src/montant.c
@@ -356,6 +356,25 @@
         if (g.facch++ < (unsigned long)journal())
             printf("  [montant] FACCH UL fn=%u (au GO)\n", fn);
     }
+    /* [2026-10-22] LA PAROLE MONTANTE AU GO. a_du (NDB_A_DU_1) souffre du meme mal que a_fu : la ROM
+     * pose B_BLUD puis l'efface des la phase B de la trame ; capture_tch_ul (montant_scruter, fin de
+     * trame) arrivait trop tard, prendre_ul() rendait false -> publier_parole() jamais appelee, SHM parole
+     * absent, pont ul=0 (aucune voix montante). On la capte ici au GO, symetrique a la FACCH ci-dessus.
+     * L'anti-doublon de prendre_ul() (dedup par contenu, vu[]) empeche le double envoi quand
+     * capture_tch_ul la retente en fin de trame. MONTANT_DU_AU_GO=0 coupe. */
+    static int du = -1;
+    if (du < 0) { const char *e = calypso_getenv("MONTANT_DU_AU_GO"); du = !(e && *e == '0'); }
+    if (du) {
+        uint8_t fr[FR_BYTES];
+        if (prendre_ul(api_ram, NDB_A_DU_1, fr, FR_BYTES)) {
+            noter_ul(fn, 1, TCHT_DSP_TASK, fr, FR_BYTES);   /* brute, format TI (a_du) */
+            parole_ti_vers_fr(fr);
+            noter_ul(fn, 0, TCHT_DSP_TASK, fr, FR_BYTES);   /* convertie FR, ce que recoit le pont */
+            publier_parole(fr, fn);
+            if (g.parole++ < (unsigned long)journal())
+                printf("  [montant] parole UL fn=%u (au GO)\n", fn);
+        }
+    }
 }

 static bool capture_tch_ul(uint16_t *api_ram, uint16_t task_u, uint32_t fn)
```

3.281 /opt/GSM/osmo-operator/patches/magnetite-sms-msginfo-doublefree.patch

5219 octets, 128 lignes → 128 lignes

```
diff -r 9394305d4ff5 src/ui3/bmi/mmiSmsRead.c
--- a/src/ui3/bmi/mmiSmsRead.c  Mon Oct 31 00:14:44 2022 +0000
+++ b/src/ui3/bmi/mmiSmsRead.c  Thu Oct 22 16:08:11 2026 +0200
@@ -777,6 +777,11 @@
```

```
            {
                for (i = 0; i < ((data->sms_list_data).nb_sms_stored); i++)
                    {
+#ifdef FF_MMI_SMS_DYNAMIC
+                   /* [banc 2026-10-22] deja libere par E_EXIT, ou jamais alloue (liste vide) */
+                   if (data->sms_list_data.sms_header == NULL || data->sms_list_data.sms_header[i] == NULL)
+                       continue;
+#endif
                    FREE_MEMORY( (void *)data->sms_list_data.sms_header[i], UCS2_MENU_TEXT_LEN+ 1/*SPR1242*/);
                    }
                /*SPR 2686, only de-allocate memory for the number of messages displayed*/
@@ -784,8 +789,11 @@
                FREE_MEMORY ((UBYTE *)data->menu_list_data, sizeof(ListMenuData));
            }
 #ifdef FF_MMI_SMS_DYNAMIC
-           FREE_MEMORY((void*)data->sms_list_data.sms_messages,g_max_messages * sizeof(T_MFW_SMS_MSG) );
-           FREE_MEMORY((void*)data->sms_list_data.sms_header, g_max_messages * sizeof(wstring_t));
+           /* [banc 2026-10-22] gardes : E_EXIT a pu les liberer deja (remis a NULL) */
+           if (data->sms_list_data.sms_messages != NULL)
+               FREE_MEMORY((void*)data->sms_list_data.sms_messages,g_max_messages * sizeof(T_MFW_SMS_MSG) );
+           if (data->sms_list_data.sms_header != NULL)
+               FREE_MEMORY((void*)data->sms_list_data.sms_header, g_max_messages * sizeof(wstring_t));
 #endif
            FREE_MEMORY ((void *)data, sizeof (T_SMSREAD_R));
            sms_read_win = 0;
@@ -1012,18 +1020,32 @@
        l_parent = data->parent;
        SmsRead_R_destroy(win);
        SEND_EVENT(l_parent, E_RETURN, data->id, NULL);
-
+       /* [banc 2026-10-22] break ajoute : E_EXIT relisait data/list_data deja liberes par SmsRead_R_destroy */
+       break;
+
    case E_EXIT: // Terminate exit from sms send...
    // May 27, 2005    MMI-FIX-29869 x0018858
 #ifdef FF_MMI_SMS_DYNAMIC
     {
           int i;
-       for (i=0;i<list_data->nb_sms_stored; i++)
-           {
-               FREE_MEMORY((U8*)list_data->sms_header[i], UCS2_MENU_TEXT_LEN + 1 );
-           }
-       FREE_MEMORY((U8*)list_data->sms_header, g_max_messages * sizeof(wstring_t) );
-       FREE_MEMORY((U8*)list_data->sms_messages, g_max_messages * sizeof(T_MFW_SMS_MSG));
+       /* [banc 2026-10-22] gardes + NULL : sans eux SmsRead_R_destroy liberait une 2e fois (tas MMI
+          corrompu,  Preceeding Sentinal Bytes have been corrupted ), et sms_header n'existe pas si la
+          liste est vide */
+       if (list_data->sms_header != NULL)
+       {
+           for (i=0;i<list_data->nb_sms_stored; i++)
+               {
+                   if (list_data->sms_header[i] != NULL)
+                       FREE_MEMORY((U8*)list_data->sms_header[i], UCS2_MENU_TEXT_LEN + 1 );
+                   list_data->sms_header[i] = NULL;
+               }
+           FREE_MEMORY((U8*)list_data->sms_header, g_max_messages * sizeof(wstring_t) );
+           list_data->sms_header = NULL;
+       }
+       if (list_data->sms_messages != NULL)
+           FREE_MEMORY((U8*)list_data->sms_messages, g_max_messages * sizeof(T_MFW_SMS_MSG));
+       list_data->sms_messages = NULL;
+       list_data->nb_sms_stored = 0;
    }
 #endif
        listDisplayListMenu(win, data->menu_list_data, (ListCbFunc_t)SmsRead_R_smslist_cb,0);
diff -r 9394305d4ff5 src/ui3/mfw/mfw_sms.c
```

```
--- a/src/ui3/mfw/mfw_sms.c Mon Oct 31 00:14:44 2022 +0000
+++ b/src/ui3/mfw/mfw_sms.c Thu Oct 22 16:08:11 2026 +0200
@@ -241,6 +241,8 @@

 //May 27, 2005    MMI-FIX-29869 x0018858 - Added a Checking variable.
 static int Counter_first = 0;
+/* [banc 2026-10-22] sms_read_msg_info() appele avant l'allocation de msg_info (voir Check_max_sms_allowed) */
+static int liste_avant_alloc = 0;
 #else
 static T_MFW_SMS_MSG msg_info[MAX_MESSAGES]; /* message table */
 #endif
@@ -348,8 +350,30 @@
                 {
                     msg_info = (T_MFW_SMS_MSG*)ALLOC_MEMORY(g_max_messages*sizeof(T_MFW_SMS_MSG));

+                    /* [banc 2026-10-22] Le tableau sortait de ALLOC_MEMORY NON initialise. rAT_sms_ready() ->
+                     sms_read_msg_info() passe souvent AVANT cette allocation (premier evenement SMS de l'ecran de
+                     veille) : sa remise a index -1 tombait sur msg_info NULL et le tableau alloue ensuite gardait
+                     des dechets. Index != -1 = entrees fantomes  [P]  dans Messages > Read ; en ouvrir une lancait
+                       sms_read sur un index bidon, sans reponse :  Please wait  infini. */
+                    for (i=0; i<g_max_messages; i++)
+                    {
+                        memset(&msg_info[i], 0, sizeof(T_MFW_SMS_MSG));
+                        msg_info[i].index = -1;
+                        msg_info[i].first_index = -1;
+                        msg_info[i].addr.ton = MFW_TON_UNKNOWN;
+                        msg_info[i].addr.npi = MFW_NPI_UNKNOWN;
+                        msg_info[i].concat_status = MFW_SMS_NO_CONC;
+                        msg_info[i].mem = MFW_MEM_NOTPRESENT;
+                        msg_info[i].first_mem = MFW_MEM_NOTPRESENT;
+                    }
                 }
             Counter_first ++;
+           /* la liste lue avant l'allocation est perdue : la relire maintenant */
+           if (liste_avant_alloc)
+           {
+               liste_avant_alloc = 0;
+               sms_read_msg_info();
+           }
         }

 }
@@ -853,6 +877,15 @@
     TRACE_FUNCTION("sms_read_msg_info()");

 #ifdef FF_MMI_SMS_DYNAMIC
+    if (msg_info == NULL)
+    {
+        /* [banc 2026-10-22] trop tot : Check_max_sms_allowed() relira la liste apres l'allocation */
+        liste_avant_alloc = 1;
+        return;
+    }
+#endif
+
+#ifdef FF_MMI_SMS_DYNAMIC
     for (i=0; i<g_max_messages; i++)
 #else
     for (i=0; i<MAX_MESSAGES; i++)
```

3.282 /opt/GSM/osmo-operator/patches/osmo-bts-force-rand-toy.patch

3024 octets, 84 lignes → 84 lignes

```
diff --git a/src/common/rsl.c b/src/common/rsl.c
index 7931a176..f8bf39ac 100644
--- a/src/common/rsl.c
+++ b/src/common/rsl.c
@@ -22,6 +22,7 @@
  */

 #include <stdio.h>
+#include <stdlib.h>
 #include <errno.h>
 #include <netdb.h>
 #include <stdbool.h>
@@ -3641,6 +3642,59 @@ static int rsl_reject_unknown_lchan(struct msgb *msg)
  * selecting message
  */

+/* forcer le RAND d'une AUTHENTICATION REQUEST (MM) descendante.
+ *
+ * La valeur vient de la variable d'environnement RAND, lue UNE fois :
+ *   RAND non defini -> on ne touche a rien (comportement normal du reseau) ;
+ *   RAND=0          -> RAND tout a zero sur 16 octets (defi fixe du toy,
+ *                      defi fixe connu) ;
+ *   RAND=<hexa>     -> jusqu'a 16 octets lus en hexa (espaces admis),
+ *                      completes a droite par des zeros.
+ * La PRESENCE de RAND est le seul interrupteur : sans elle le RAND du reseau
+ * passe intact. C'est donc « seulement quand precise, mais force quand precise ».
+ * "0" est traite a part car osmo_hexparse() refuse un nombre impair de quartets. */
+static void deka_force_auth_rand(struct msgb *msg)
+{
+   static int inited;
+   static bool forced;
+   static uint8_t rand_val[16];
+   struct gsm48_hdr *gh;
+   struct gsm48_auth_req *ar;
+
+   if (!inited) {
+       const char *env = getenv("RAND");
+       inited = 1;
+       if (env != NULL) {
+           forced = true;
+           if (env[0] == '0' && env[1] == '\0') {
+               memset(rand_val, 0, sizeof(rand_val));
+           } else if (osmo_hexparse(env, rand_val, sizeof(rand_val)) < 0) {
+               /* osmo_hexparse() a deja remis rand_val a zero ;
+                * on force donc un RAND nul et on le signale. */
+               LOGP(DRSL, LOGL_ERROR, "RAND=\"%s\" illisible "
+                    "(hexa <= 16 octets attendu) -> RAND=0\n", env);
+           }
+           LOGP(DRSL, LOGL_NOTICE, "RAND force a %s dans toute "
+                "AUTH REQ descendante\n",
+                osmo_hexdump_nospc(rand_val, sizeof(rand_val)));
+       }
+   }
+   if (!forced)
+       return;
+
+   if (msgb_l3len(msg) < sizeof(*gh) + sizeof(*ar))
+       return;
+   gh = (struct gsm48_hdr *) msgb_l3(msg);
+   if ((gh->proto_discr & 0x0f) != GSM48_PDISC_MM)
+       return;
+   /* bit 6 (0x40) du type porte le numero de sequence MM : on l'ignore. */
+   if ((gh->msg_type & 0x3f) != GSM48_MT_MM_AUTH_REQ)
+       return;
+
+   ar = (struct gsm48_auth_req *) gh->data;
```

```
+   memcpy(ar->rand, rand_val, sizeof(ar->rand));
+}
+
 static int rsl_rx_rll(struct gsm_bts_trx *trx, struct msgb *msg)
 {
    struct abis_rsl_rll_hdr *rh = msgb_l2(msg);
@@ -3680,6 +3734,11 @@ static int rsl_rx_rll(struct gsm_bts_trx *trx, struct msgb *msg)

    LOGPLCHAN(lchan, DRLL, LOGL_DEBUG, "Rx RLL %s Abis -> LAPDm\n", rsl_msg_name(rh->c.msg_type));

+   /* sur une DATA REQ (L3 descendant vers le mobile), reecrire le
+    * RAND de l'AUTH REQUEST si l'environnement RAND est pose. */
+   if (rh->c.msg_type == RSL_MT_DATA_REQ)
+       deka_force_auth_rand(msg);
+
    /* make copy of RLL header, as the message will be free'd in case of erroneous return */
    rh2 = *rh;
    /* exception: RLL messages are _NOT_ freed as they are now
```

3.283 /opt/GSM/osmo-operator/patches/osmo-hlr-force-rand-toy.patch

761 octets, 25 lignes → 25 lignes

```
diff --git a/src/auc.c b/src/auc.c
index 3ec0893..9fa763c 100644
--- a/src/auc.c
+++ b/src/auc.c
@@ -18,6 +18,7 @@
  */

 #include <string.h>
+#include <stdlib.h>
 #include <inttypes.h>

 #include <osmocom/core/utils.h>
@@ -110,6 +111,12 @@ int auc_compute_vectors(struct osmo_auth_vector *vec, unsigned int num_vec,
                 "bytes: rc=%d\n", sizeof(rand), rc);
            goto out;
        }
+       /* gate RAND (getenv) - force le defi du vecteur a 0
+        * (COMP128v1 toy) pour que SRES/Kc soient calcules sur RAND=0
+        * cote HLR ; sinon osmo-msc envoie 0 mais le SRES attendu est
+        * calcule sur un autre RAND -> mismatch -> LU reject. */
+       if (getenv("RAND"))
+           memset(rand, 0, sizeof(rand));
        DBGP("vector [%u]: rand = %s\n", i, hexb(rand));

        if (aud3g) {
```

3.284 /opt/GSM/osmo-operator/patches/osmo-msc-force-rand-toy.patch

853 octets, 18 lignes → 18 lignes

```
diff --git a/src/libmsc/gsm_04_08.c b/src/libmsc/gsm_04_08.c
index de0a69373..b28bd6746 100644
--- a/src/libmsc/gsm_04_08.c
+++ b/src/libmsc/gsm_04_08.c
@@ -638,6 +638,13 @@ int gsm48_tx_mm_auth_req(struct msc_a *msc_a, uint8_t *rand,
    struct gsm48_hdr *gh = (struct gsm48_hdr *) msgb_put(msg, sizeof(*gh));
    struct gsm48_auth_req *ar = (struct gsm48_auth_req *) msgb_put(msg, sizeof(*ar));

+   /* gate de forcage du RAND. RAND pose dans l'environnement ->
```

```
+    * on force le defi (0 sur 16 octets dans notre cas ; changer
+    * l'initialiseur pour un autre RAND). RAND absent -> comportement normal. */
+   static uint8_t forced_rand[16] = { 0 };
+   if (getenv("RAND"))
+       rand = forced_rand;
+
    DEBUGP(DMM, "Tx AUTH REQ (rand = %s)\n", osmo_hexdump_nospc(rand, 16));
    if (autn)
        DEBUGP(DMM, "   AUTH REQ (autn = %s)\n", osmo_hexdump_nospc(autn, 16));
```

3.285 /opt/GSM/osmo-operator/patches/osmo-trx-ipc-ts-frame-align.patch

1892 octets, 29 lignes → 29 lignes

```
diff --git a/Transceiver52M/device/ipc/IPCDevice.cpp b/Transceiver52M/device/ipc/IPCDevice.cpp
index 1d2b89a..871075a 100644
--- a/Transceiver52M/device/ipc/IPCDevice.cpp
+++ b/Transceiver52M/device/ipc/IPCDevice.cpp
@@ -1073,6 +1073,24 @@ bool IPCDevice::flush_recv(size_t num_pkts)
            read = ipc_shm_read(shm_io_rx_streams[i], (uint16_t *)&tmp.front(), 4096 / 2, &tmps, 3);
    }
    ts_initial = tmps + read;
+   /* FIX LU 2026-06-05 : aligner ts_initial sur une frontiere de TRAME TDMA
+    * (8 slots * 625 samples @ 4 SPS = 5000). Le device IPC commit des buffers a
+    * des timestamps multiples de 2500 (CALYPSO_SHM_BUFSIZE), donc ts_initial%5000
+    * valait 0 OU 2500 -> quand 2500, device-TN0 mappe sur osmo-TN4 et la RACH
+    * (placee a l'offset 0 = device-TN0) ratait le slot TS0/RACH -> NOPE/-110.
+    * Arrondir a la trame met les deux grilles en phase. (5000 = 8*625 @ rx-sps 4) */
+   ts_initial -= ts_initial % 5000;
+   /* FIX SDCCH RUN-INVARIANCE 2026-06-09 : aligner ts_initial sur une frontiere de
+    * MULTITRAME 102 (couvre CCCH+SDCCH/4) en plus de la trame. mClock demarre a FN 0
+    * (CALYPSO_START_FN=0) MAIS l'echantillon a ts_initial est etiquete FN 0 -> Q =
+    * osmo_trx_fn - internal_fn = -(ts_initial/5000). Le drain SHM au handshake
+    * OPEN->START est une course de demarrage -> ts_initial/5000 (donc Q mod 102)
+    * variait run-a-run -> le bloc SDCCH/4 SS0 UL tombait a un %51 different chaque run
+    * -> SABM decodee seulement parfois. En forcant ts_initial%(102*5000)==0 -> Q==0
+    * mod 102 a chaque run -> osmo_trx_fn==internal_fn (mod 102) -> gate fixe cote
+    * qemu_wrap (CALYPSO_UL_SDCCH_OFS=15) run-invariant. 102*5000=510000 ; gHyperframe
+    * en echantillons est multiple exact de 510000 -> wrap d'hypertrame non perturbe. */
+   ts_initial -= ts_initial % (102 * 5000);

    LOGC(DDEV, INFO) << "Initial timestamp " << ts_initial << std::endl;
    return true;
```

3.286 /opt/GSM/osmo-operator/patches/osmo-trx-rach-per-ra-table.patch

10975 octets, 231 lignes → 231 lignes

```
diff --git a/Transceiver52M/Makefile.am b/Transceiver52M/Makefile.am
index 0b63b16..fa260f5 100644
--- a/Transceiver52M/Makefile.am
+++ b/Transceiver52M/Makefile.am
@@ -73,7 +73,8 @@ COMMON_LDADD = \
    $(FFTWF_LIBS) \
    $(LIBOSMOCORE_LIBS) \
    $(LIBOSMOCTRL_LIBS) \
-   $(LIBOSMOVTY_LIBS)
+   $(LIBOSMOVTY_LIBS) \
+   $(LIBOSMOCODING_LIBS)

 if ENABLE_MS_TRX
 AM_CPPFLAGS += -I$(top_srcdir)/osmocom-bb/src/host/trxcon/include/
```

```
diff --git a/Transceiver52M/Transceiver.cpp b/Transceiver52M/Transceiver.cpp
index f5ddd9b..9c6b76a 100644
--- a/Transceiver52M/Transceiver.cpp
+++ b/Transceiver52M/Transceiver.cpp
@@ -24,6 +24,7 @@
 */

 #include <stdio.h>
+#include <stdlib.h>     // getenv/atoi (debug SDCCH UL Calypso)
 #include <netinet/in.h>
 #include <iomanip>      // std::setprecision
 #include <fstream>
@@ -258,8 +259,13 @@ bool Transceiver::init()
     mStates[i].init(filler, cfg->tx_sps, txFullScale, cfg->rtsc, cfg->rach_delay);
   }

-  /* Randomize the central clock */
-  GSM::Time startTime(random() % gHyperframe, 0);
+  /* Horloge centrale : random par defaut. CALYPSO_START_FN=N force un FN de
+   * depart DETERMINISTE -> tue la run-variance tn=0/tn=5 de l'injection UL Calypso
+   * (mClock demarrait a un FN aleatoire -> l'etiquette TN glissait). */
+  unsigned _start_fn;
+  { const char *e = getenv("CALYPSO_START_FN");
+    _start_fn = e ? (unsigned)(atol(e) % gHyperframe) : (unsigned)(random() % gHyperframe); }
+  GSM::Time startTime(_start_fn, 0);
   mRadioInterface->getClock()->set(startTime);
   mTransmitDeadlineClock = startTime;
   mLastClockUpdateTime = startTime;
@@ -741,6 +747,13 @@ int Transceiver::pullRadioVector(size_t chan, struct trx_ul_burst_ind *bi)
   burst = radio_burst->getVector(max_i);
   avg = sqrt(avg / radio_burst->chans());

+  /* ENERGY-DBG 2026-06-05 : logge TOUT slot avec de l'energie (un burst presente),
+   * quel que soit le type/TN, pour trouver OU la RACH injectee atterrit vraiment
+   * dans la grille osmo-trx (avant le filtre type==IDLE). Pas de cap : rare. */
+  if (avg > 800.0)
+    LOGCHAN(chan, DTRXDUL, NOTICE) << "ENERGY fn=" << burstTime.FN()
+      << " tn=" << burstTime.TN() << " type=" << type << " avg=" << avg << " max=" << max;
+
   if (type == IDLE) {
     /* Update noise levels */
     state->mNoises.insert(avg);
@@ -756,6 +769,14 @@ int Transceiver::pullRadioVector(size_t chan, struct trx_ul_burst_ind *bi)

   max_toa = (type == RACH || type == EXT_RACH) ?
             mMaxExpectedDelayAB : mMaxExpectedDelayNB;
+  /* DEBUG SDCCH UL (Calypso) : osmo-bts n'envoie pas SETMAXDLYNB -> mMaxExpectedDelayNB=0
+   * -> fenetre TOA du burst normal trop etroite pour le burst injecte par qemu_wrap.
+   * CALYPSO_NB_MAXDLY=N (symboles) elargit la fenetre TOA late du burst normal. */
+  if (type != RACH && type != EXT_RACH) {
+    static int nb_maxdly = -1;
+    if (nb_maxdly < 0) { const char *e = getenv("CALYPSO_NB_MAXDLY"); nb_maxdly = e ? atoi(e) : 0; }
+    if (nb_maxdly > 0) max_toa = nb_maxdly;
+  }

   if (cfg->use_va) {
     // shifted burst copy to make the old demod and detection happy
@@ -766,6 +787,15 @@ int Transceiver::pullRadioVector(size_t chan, struct trx_ul_burst_ind *bi)
   /* Detect normal or RACH bursts */
   rc = detectAnyBurst(*shvec_ptr, mTSC, BURST_THRESH, cfg->rx_sps, type, max_toa, &ebp);
+  if ((type == RACH || type == EXT_RACH) && avg > 800.0)
+    LOGCHAN(chan, DTRXDUL, NOTICE) << "RACH-DET fn=" << burstTime.FN()
+      << " rc=" << rc << " toa=" << ebp.toa << " tsc=" << ebp.tsc;
+  /* DEBUG SDCCH UL (Calypso) : sonde symetrique a RACH-DET pour le burst NORMAL,
+   * limitee a TN0 (le slot CCCH+SDCCH4) pour ne pas flooder. rc>0 = NB detecte,
```

```
+   * toa = decalage mesure -> dit si c'est detection (offset) ou demod. */
+  if ((type != RACH && type != EXT_RACH) && avg > 800.0 && burstTime.TN() == 0)
+    LOGCHAN(chan, DTRXDUL, NOTICE) << "NB-DET fn=" << burstTime.FN()
+      << " rc=" << rc << " toa=" << ebp.toa << " tsc=" << ebp.tsc << " avg=" << avg;
   if (rc <= 0) {
     if (rc == -SIGERR_CLIP) {
       LOGCHAN(chan, DTRXDUL, INFO) << "Clipping detected on received RACH or Normal Burst";
@@ -790,6 +820,33 @@ int Transceiver::pullRadioVector(size_t chan, struct trx_ul_burst_ind *bi)
   bi->tsc = ebp.tsc;
   bi->ci = ebp.ci;

+  /* SONDE SDCCH (Calypso) : |soft| moyen des bits DATA (3..60, 87..144) vs
+   * MIDAMBLE (61..86) sur les bursts NORMAUX tn=0. Discrimine la cause du BER
+   * 456/456 : data_re~0 + data_im>>data_re => bits data sur l'axe imaginaire
+   * (rotation pi/2 fausse, jetee par .real()) ; data_re~mid_re => data OK ailleurs ;
+   * data_re~0 & data_im~0 => pas de signal data. */
+  if ((type != RACH && type != EXT_RACH) && avg > 800.0 && burstTime.TN() == 0 && rxBurst) {
+    double dabs = 0, mabs = 0; int nd = 0, nm = 0;
+    int Nsb = (int)rxBurst->size();
+    for (int i = 0; i < Nsb && i < 148; i++) {
+      float v = rxBurst->begin()[i];   /* SoftVector = float (re deja extrait) */
+      if (i >= 61 && i <= 86) { mabs += fabsf(v); nm++; }
+      else if (i >= 3 && i <= 144) { dabs += fabsf(v); nd++; }
+    }
+    /* bits durs démodulés aux positions TSC (61..86) : référence CONNUE = TSC7
+     * 11101111000100101110111100. Match=correct ; complément=inversion ;
+     * décalé=alignement ; garbage=démod faux. Idem 1ers bits data (3..12). */
+    char tscb[32]; int tj = 0;
+    for (int i = 61; i <= 86 && tj < 31; i++) tscb[tj++] = (rxBurst->begin()[i] > 0) ? '1' : '0';
+    tscb[tj] = 0;
+    char datb[16]; int dj = 0;
+    for (int i = 3; i <= 12 && dj < 15; i++) datb[dj++] = (rxBurst->begin()[i] > 0) ? '1' : '0';
+    datb[dj] = 0;
+    LOGCHAN(chan, DTRXDUL, NOTICE) << "SOFT-DBG fn=" << burstTime.FN()
+      << " data_abs=" << (nd ? dabs / nd : 0.0) << " mid_abs=" << (nm ? mabs / nm : 0.0)
+      << " tsc=" << tscb << " dat0=" << datb;
+  }
+
   /* EDGE demodulator returns 444 (gSlotLen * 3) bits */
   if (rxBurst->size() == EDGE_BURST_NBITS) {
     bi->modulation = MODULATION_8PSK;
diff --git a/Transceiver52M/sigProcLib.cpp b/Transceiver52M/sigProcLib.cpp
index 5fac365..b34851a 100644
--- a/Transceiver52M/sigProcLib.cpp
+++ b/Transceiver52M/sigProcLib.cpp
@@ -35,6 +35,8 @@

 extern "C" {
 #include <osmocom/core/panic.h>
+#include <osmocom/core/bits.h>
+#include <osmocom/coding/gsm0503_coding.h>
 #include "convolve.h"
 #include "scale.h"
 #include "mult.h"
@@ -2148,6 +2150,96 @@ bool sigProcLibSetup()
   generateRACHSequence(&gRACHSequences[1], gRACHSynchSequenceTS1, 1);
   generateRACHSequence(&gRACHSequences[2], gRACHSynchSequenceTS2, 1);

+  /* DUMP RACH ref 2026-06-05 (OUTIL DE PREUVE, PAS LE FINAL) : access-burst
+   * [8 tail][41 sync TS0][36 data][3 tail] module par le modulateur EXACT d'osmo-trx
+   * (Laurent, sps=4) -> /root/rach_ref.cs16, rejoue par le device (sa modulation maison
+   * ne correle pas). data : ZERO si CALYPSO_UL_RA non defini (PORTE 1 = rc>0, la sync
+   * seule suffit) ; sinon gsm0503_rach_ext_encode(RA,BSIC) (PORTE 2 = CHAN RQD). RA/BSIC
+   * viennent de l'ENV (config, PAS de hardcode). Le FINAL = device encode+modulate le
+   * RA REEL (plombe du BSP) + BSIC reel, dynamiquement. */
+  {
```

```
+    static const char SYNC[] = "01001011011111111100110011010101000111100";
+    char bs[89];
+    for (int i = 0; i < 88; i++) bs[i] = '0';
+    for (int i = 0; i < 41; i++) bs[8 + i] = SYNC[i];
+    const char *era = getenv("CALYPSO_UL_RA");
+    if (era && *era) {
+      const char *ebs = getenv("CALYPSO_UL_BSIC");
+      uint16_t ra = (uint16_t) strtol(era, NULL, 0);
+      uint8_t bsic = (uint8_t)(ebs && *ebs ? atoi(ebs) : 0);
+      ubit_t coded[40]; memset(coded, 0, sizeof(coded));
+      gsm0503_rach_ext_encode(coded, ra, bsic, false);   /* 36 bits codes (RA + parite BSIC) */
+      for (int i = 0; i < 36; i++) bs[49 + i] = coded[i] ? '1' : '0';
+      LOG(NOTICE) << "[rach-ref] RA=" << (int) ra << " BSIC=" << (int) bsic << " (PORTE 2, env)";
+    } else {
+      LOG(NOTICE) << "[rach-ref] sync seule, data=0 (PORTE 1 ; set CALYPSO_UL_RA pour PORTE 2)";
+    }
+    bs[88] = 0;
+    BitVector ab(bs);
+    signalVector *rb = modulateBurst(ab, 68, 4, false);
+    if (rb) {
+      FILE *f = fopen("/root/rach_ref.cs16", "wb");
+      if (f) {
+        float *p = (float *) rb->begin();
+        for (size_t i = 0; i < rb->size(); i++) {
+          float I = p[2 * i] * 20000.0f, Q = p[2 * i + 1] * 20000.0f;
+          if (I > 32767.0f) I = 32767.0f; else if (I < -32768.0f) I = -32768.0f;
+          if (Q > 32767.0f) Q = 32767.0f; else if (Q < -32768.0f) Q = -32768.0f;
+          int16_t s[2] = { (int16_t) I, (int16_t) Q };
+          fwrite(s, sizeof(int16_t), 2, f);
+        }
+        fclose(f);
+        LOG(NOTICE) << "[rach-ref] dumped " << rb->size() << " IQ -> /root/rach_ref.cs16";
+      }
+      delete rb;
+    }
+  }
+
+  /* TABLE per-RA 2026-06-05 (NO-HARDCODE) : la vraie RA du mobile (d_rach@0x0474,
+   * plombee via /dev/shm/calypso_rach) varie a chaque burst. On pre-genere ICI la
+   * modulation EXACTE d'osmo-trx (Laurent sps=4, la seule qui correle son propre
+   * detecteur) pour chaque RA 0x00..0x0f a BSIC reel -> /root/rach_ref_RA<nn>.cs16.
+   * Le device (ul_drain) selectionne le ref de la vraie RA recue. BSIC = celui du
+   * BTS (env CALYPSO_UL_BSIC, defaut 7 ; colore la parite). Pas de RA hardcodee :
+   * c'est un codebook, la vraie RA en selectionne l'entree. */
+  {
+    static const char SYNC[] = "01001011011111111100110011010101000111100";
+    const char *ebs = getenv("CALYPSO_UL_BSIC");
+    uint8_t bsic = (uint8_t)(ebs && *ebs ? atoi(ebs) : 7);
+    int ntab = 0;
+    for (int ra = 0; ra < 16; ra++) {
+      char bs[89];
+      for (int i = 0; i < 88; i++) bs[i] = '0';
+      for (int i = 0; i < 41; i++) bs[8 + i] = SYNC[i];
+      ubit_t coded[40]; memset(coded, 0, sizeof(coded));
+      gsm0503_rach_ext_encode(coded, (uint16_t) ra, bsic, false);
+      for (int i = 0; i < 36; i++) bs[49 + i] = coded[i] ? '1' : '0';
+      bs[88] = 0;
+      BitVector ab(bs);
+      signalVector *rb = modulateBurst(ab, 68, 4, false);
+      if (!rb) continue;
+      char path[64];
+      snprintf(path, sizeof(path), "/root/rach_ref_RA%02x.cs16", ra);
+      FILE *f = fopen(path, "wb");
+      if (f) {
+        float *p = (float *) rb->begin();
+        for (size_t i = 0; i < rb->size(); i++) {
```

```
+          float I = p[2 * i] * 20000.0f, Q = p[2 * i + 1] * 20000.0f;
+          if (I > 32767.0f) I = 32767.0f; else if (I < -32768.0f) I = -32768.0f;
+          if (Q > 32767.0f) Q = 32767.0f; else if (Q < -32768.0f) Q = -32768.0f;
+          int16_t s[2] = { (int16_t) I, (int16_t) Q };
+          fwrite(s, sizeof(int16_t), 2, f);
+        }
+        fclose(f);
+        ntab++;
+      }
+      delete rb;
+    }
+    LOG(NOTICE) << "[rach-ref] table per-RA : " << ntab
+                << " refs (RA 0x00..0x0f, BSIC=" << (int) bsic
+                << ") -> /root/rach_ref_RA*.cs16";
+  }
+
    generateSCHSequence(1);
    generateDummyMidamble(1);
```

3.287 /opt/GSM/osmo-operator/patches/osmocom-bb-clck-gen-frame-tolerance.patch

1078 octets, 23 lignes → 23 lignes

```
diff --git a/src/target/trx_toolkit/clck_gen.py b/src/target/trx_toolkit/clck_gen.py
index f769c3e9..8c253ea2 100755
--- a/src/target/trx_toolkit/clck_gen.py
+++ b/src/target/trx_toolkit/clck_gen.py
@@ -104,8 +104,16 @@ class CLCKGen:
            t = time.monotonic_ns()
            dt = (t_next - t)
            if dt < 0:
-               log.warning("CLCKGen: time overrun by %dus; resetting the clock" % (dt * ns // us))
-               t_next = time.monotonic_ns()
+               # [osmo-operator 2026-09-08] Only rebase the clock when we are
+               # late by more than one full TDMA frame. A sub-frame overrun
+               # (0.3..2 ms on a loaded host) used to reset t_next as well,
+               # shifting the timebase by that much every time: osmo-bts-trx
+               # then logged "N FN slower/faster than TRX, compensating" all
+               # day long and the speech path lost frames (audible dropouts
+               # at TCH start). Below one frame we simply catch up.
+               if dt < -t_tick:
+                   log.warning("CLCKGen: time overrun by %dus; resetting the clock" % (dt * ns // us))
+                   t_next = time.monotonic_ns()
                dt = 0

            if self._breaker.wait(dt * ns):
```

3.288 /opt/GSM/osmo-operator/patches/osmocom-bb-fbsb-timeout-scan-to.patch

5064 octets, 109 lignes → 109 lignes

```
osmocom-bb : rendre reglables les deux budgets d'une tentative de synchro.

Sous emulation QEMU-Calypso, l'interpreteur c54x fait tourner le tick TDMA a
~10 Hz au lieu de 217 (facteur ~96). Les deux budgets qu'osmocom fixe en dur
sont alors trop courts d'autant, et le mobile boucle :

  gsm322_sync_to_cell() emet L1CTL_RESET_REQ puis un FBSB_REQ avec un timeout
  litteral de 100 TRAMES (461 ms de temps GSM, correct sur silicium, ~10 s de
  temps reel ici). Le firmware abandonne, renvoie un FBSB_CONF d'echec, la
  couche 3 loggue « Channel sync error » / « Cell selection failed » et rappelle
```

```
  gsm322_sync_to_cell() — donc re-emet le RESET. Le DSP emule n'obtient jamais
  une fenetre assez longue pour acquerir FB+SB.

  support.c fixe scan_to = 4 s, sans aucune commande VTY : impossible a regler
  depuis mobile.cfg.

Ce correctif n'introduit AUCUNE valeur fabriquee : ni detection, ni resultat
DSP, ni cellule d'API. Il empeche seulement le banc d'interrompre son propre
sujet. Si le DSP ne trouve rien, il ne trouvera toujours rien — sans etre coupe.

  CALYPSO_FBSB_TIMEOUT=<trames>    defaut 100   (plafond 65535, champ uint16)
  CALYPSO_SCAN_TO=<secondes>       defaut 4     (plafond 255, champ uint8)

Les deux doivent etre accordes : SCAN_TO doit depasser la duree MUR de
FBSB_TIMEOUT trames. A ~10 Hz, 2000 trames (~200 s) va avec SCAN_TO=240.

Absentes, les deux variables laissent le comportement osmocom d'origine.
diff --git a/src/host/layer23/src/common/support.c b/src/host/layer23/src/common/support.c
index 3e053713..a1c1c8bf 100644
--- a/src/host/layer23/src/common/support.c
+++ b/src/host/layer23/src/common/support.c
@@ -89,7 +89,24 @@ void gsm_support_init(struct osmocom_ms *ms)
     sup->ch_cap = GSM_CAP_SDCCH_TCHF_TCHH;
     sup->min_rxlev_dbm = -106; // TODO
     sup->sync_to = 6; /* how long to wait sync (0.9 s) */
+    /* [2026-08-03] Reglable : sous emulation le DSP est ~96x trop lent, et 4 s
+     * de mur ne couvrent qu'une fraction du temps GSM necessaire. Sans commande
+     * VTY jusqu'ici, d'ou l'impossibilite de le regler depuis mobile.cfg.
+     * CALYPSO_SCAN_TO=<secondes> ; absente = 4 = comportement d'origine.
+     * Champ uint8_t -> plafonne a 255. Instrument de banc, pas une bequille :
+     * aucune valeur radio n'est fabriquee. */
     sup->scan_to = 4; /* how long to wait for all sysinfos (>=4 s) */
+    {
+        const char *e = getenv("CALYPSO_SCAN_TO");
+        if (e && *e) {
+            long v = strtol(e, NULL, 0);
+            if (v < 1) v = 1;
+            if (v > 255) v = 255;
+            sup->scan_to = (uint8_t)v;
+            printf("[support] CALYPSO_SCAN_TO=%u s (defaut 4)\n",
+                   sup->scan_to);
+        }
+    }
     sup->dsc_max = 90; /* the specs defines 90 */

     /* codec */
diff --git a/src/host/layer23/src/mobile/gsm322.c b/src/host/layer23/src/mobile/gsm322.c
index 3d0cc77d..698648c8 100644
--- a/src/host/layer23/src/mobile/gsm322.c
+++ b/src/host/layer23/src/mobile/gsm322.c
@@ -493,10 +493,44 @@ static int gsm322_sync_to_cell(struct gsm322_cellsel *cs,

     cs->sync_pending = 1;
     l1ctl_tx_reset_req(ms, L1CTL_RES_T_FULL);
-    return l1ctl_tx_fbsb_req(ms, cs->arfcn,
-            L1CTL_FBSB_F_FB01SB, 100, 0,
-            cs->ccch_mode,
-            cs->list[cs->arfci].rxlev);
+
+    /* [2026-08-03] BUDGET D'UNE TENTATIVE FBSB, en TRAMES TDMA.
+     *
+     * Etait le litteral 100 : 100 trames = 461 ms de temps GSM, parfaitement
+     * correct sur silicium. Mais sous emulation le tick TDMA du DSP tourne a
+     * ~10 Hz au lieu de 217 (interpreteur c54x ~96x trop lent), donc ces
+     * 100 trames valent ~10 s de temps REEL. Le firmware abandonne, envoie un
+     * FBSB_CONF d'echec, la couche 3 loggue « Channel sync error » puis
```

```
+    * « Cell selection failed », et gsm322_sync_to_cell() est rappelee — ce qui
+    * re-emet le L1CTL_RESET_REQ ci-dessus. C'est CETTE boucle qui empeche le
+    * DSP natif d'avoir une seule fenetre assez longue pour acquerir FB+SB.
+    *
+    * CE N'EST PAS UNE BEQUILLE : on ne fabrique aucun resultat DSP, aucune
+    * detection, aucune valeur d'API. On empeche seulement le banc d'interrompre
+    * son propre sujet toutes les ~8 s. Le juge reste intact : si le DSP ne
+    * trouve rien, il ne trouvera toujours rien, simplement sans etre coupe.
+    *
+    * CALYPSO_FBSB_TIMEOUT=<trames> ; absente = 100 = comportement d'origine.
+    * Champ uint16 cote protocole -> plafonne a 65535. En natif, viser large :
+    * 20000 trames ~= 30 min de mur a 10 Hz, soit ~92 s de temps GSM simule. */
+   {
+       static int fbsb_to = -1;
+       if (fbsb_to < 0) {
+           const char *e = getenv("CALYPSO_FBSB_TIMEOUT");
+           long v = (e && *e) ? strtol(e, NULL, 0) : 100;
+           if (v < 1) v = 1;
+           if (v > 65535) v = 65535;
+           fbsb_to = (int)v;
+           if (fbsb_to != 100)
+               LOGP(DCS, LOGL_NOTICE, "FBSB timeout force a %d trames "
+                    "(defaut 100) — fenetre de synchro allongee pour le "
+                    "DSP emule ; aucun resultat n'est fabrique\n", fbsb_to);
+       }
+       return l1ctl_tx_fbsb_req(ms, cs->arfcn,
+               L1CTL_FBSB_F_FB01SB, (uint16_t)fbsb_to, 0,
+               cs->ccch_mode,
+               cs->list[cs->arfci].rxlev);
+   }
 }

 /* this is called whenever the serving cell is unselectied */
```

3.289 /opt/GSM/osmo-operator/patches/osmocom-bb-firmware-gprs.patch

20467 octets, 635 lignes → 633 lignes (2 groupes compactés)

```
diff --git a/src/target/firmware/include/layer1/prim.h b/src/target/firmware/include/layer1/prim.h
index 4e34bb23..ea2715ba 100644
--- a/src/target/firmware/include/layer1/prim.h
+++ b/src/target/firmware/include/layer1/prim.h
@@ -25,6 +25,7 @@ void l1a_rach_req(uint16_t offset, uint8_t combined, uint8_t ra, uint8_t uic);
 /* Primitives raw scheduling sets */
 extern const struct tdma_sched_item nb_sched_set[];
 extern const struct tdma_sched_item nb_sched_set_ul[];
+extern const struct tdma_sched_item pdtch_sched_set_ul[];

 extern const struct tdma_sched_item tch_sched_set[];
 extern const struct tdma_sched_item tch_a_sched_set[];
diff --git a/src/target/firmware/layer1/Makefile b/src/target/firmware/layer1/Makefile
index 0a24e054..b1765b44 100644
--- a/src/target/firmware/layer1/Makefile
+++ b/src/target/firmware/layer1/Makefile
@@ -5,5 +5,5 @@ LIB_layer1_SRCS=avg.c agc.c afc.c toa.c sync.c tdma_sched.c tpu_window.c init.c
        l23_api.c mframe_sched.c sched_gsmtime.c async.c rfch.c

 LIB_layer1_SRCS += prim_pm.c prim_rach.c prim_tx_nb.c prim_rx_nb.c prim_fbsb.c \
-          prim_freq.c prim_utils.c prim_tch.c
+          prim_freq.c prim_utils.c prim_tch.c l1gprs_fw.c prim_pdtch.c

diff --git a/src/target/firmware/layer1/l23_api.c b/src/target/firmware/layer1/l23_api.c
index f4309e2a..2519a59c 100644
--- a/src/target/firmware/layer1/l23_api.c
```

```
+++ b/src/target/firmware/layer1/l23_api.c
@@ -44,6 +44,7 @@
 #include <calypso/dsp.h>

 #include <l1ctl_proto.h>
+#include <layer1/l1gprs_fw.h>

 /* the size we will allocate struct msgb* for HDLC */
 #define L3_MSG_HEAD 4
@@ -300,6 +301,26 @@ static void l1ctl_rx_dm_est_req(struct msgb *msg)

     /* figure out which MF tasks to enable */
     l1a_mftask_set(chan_nr2mf_task_mask(ul->chan_nr, NEIGH_MODE_PM));
+
+   /* [2026-10-15] PDCH : annoncer l intervalle TOUT DE SUITE. Sur le banc
+    * (QEMU + c54x_exe), le BSP ne joue l intervalle du PDCH qu une fois que le
+    * tap QEMU a vu un L1CTL_GPRS_DL_BLOCK_IND de ce TN. Le premier n arrivait
+    * qu apres le premier bloc PDTCH decode - lui-meme joue depuis TS0, donc
+    * perdu : les premiers blocs apres l assignation (commandes, polls)
+    * manquaient, T3190/T3168 et TBF abandonne. Une indication vide (meme
+    * forme qu un bloc en erreur, cf. l1gprs de trxcon) arme le BSP des
+    * l etablissement du PDCH. */
+   if (l1s.dedicated.type == GSM_DCHAN_PDCH) {
+       struct l1ctl_gprs_dl_block_ind *ind;
+       struct msgb *m = l1ctl_msgb_alloc(L1CTL_GPRS_DL_BLOCK_IND);
+       ind = (struct l1ctl_gprs_dl_block_ind *) msgb_put(m, sizeof(*ind));
+       memset(ind, 0, sizeof(*ind));
+       ind->hdr.fn = htonl(l1s.current_time.fn);
+       ind->hdr.tn = ul->chan_nr & 0x7;
+       ind->usf = 0xff;
+       l1_queue_for_l2(m);
+       printf("l1gprs: PDCH TN=%u etabli, annonce\n", ul->chan_nr & 0x7);
+   }
 }

 /* receive a L1CTL_DM_FREQ_REQ from L23 */
@@ -351,6 +372,7 @@ static void l1ctl_rx_crypto_req(struct msgb *msg)
 /* receive a L1CTL_DM_REL_REQ from L23 */
 static void l1ctl_rx_dm_rel_req(struct msgb *msg)
 {
+   l1gprs_fw_reset();  /* PDCH: drop TBFs and queued uplink blocks */
     printd("L1CTL_DM_REL_REQ\n");
     l1a_mftask_set(0);
     l1s.dedicated.type = GSM_DCHAN_NONE;
@@ -719,6 +741,19 @@ void l1a_l23_handler(void)
     case L1CTL_SIM_REQ:
         l1ctl_sim_req(msg);
         break;
+   case L1CTL_GPRS_UL_TBF_CFG_REQ:
+       l1gprs_fw_ul_tbf_cfg(msg);
+       break;
+   case L1CTL_GPRS_DL_TBF_CFG_REQ:
+       l1gprs_fw_dl_tbf_cfg(msg);
+       break;
+   case L1CTL_GPRS_UL_BLOCK_REQ:
+       l1gprs_fw_ul_block_req(msg);
+       /* queued for prim_pdtch.c (or freed there on error) */
+       goto exit_nofree;
+   default:
+       printf("l1a_l23_cb: unknown L1CTL msg_type 0x%02x\n", l1h->msg_type);
+       break;
     }

 exit_msgbfree:
diff --git a/src/target/firmware/layer1/mframe_sched.c b/src/target/firmware/layer1/mframe_sched.c
index 478fd20a..8ad23838 100644
```

```
--- a/src/target/firmware/layer1/mframe_sched.c
+++ b/src/target/firmware/layer1/mframe_sched.c
@@ -46,6 +46,7 @@ struct mframe_sched_item {
 #define NB_QUAD_FH_DL  NB_QUAD_DL
 #define NB_QUAD_UL nb_sched_set_ul
 #define NB_QUAD_FH_UL  NB_QUAD_UL
+#define PDTCH_QUAD_UL  pdtch_sched_set_ul
 #define NEIGH_PM   neigh_pm_sched_set

 /* BCCH Normal */
@@ -312,7 +313,11 @@ static const struct mframe_sched_item mf_gprs_pdtch[] = {
    { .sched_set = NB_QUAD_DL, .modulo = 13, .frame_nr = ⟨1⟩ },  ×3
    ⟨⟩ = (0) (4) (8)
-   /* NOTE: receive only task */
+   /* Uplink block periods, same frame numbers (3-TS offset in the TPU):
+    * prim_pdtch.c only transmits when L23 queued a block for that Fn. */
+   { .sched_set = PDTCH_QUAD_UL, .modulo = 13, .frame_nr = ⟨1⟩ },  ×3
    ⟨⟩ = (0) (4) (8)
    { .sched_set = NULL }
 };

diff --git a/src/target/firmware/layer1/prim_rx_nb.c b/src/target/firmware/layer1/prim_rx_nb.c
index d6dd82a5..6cee98ce 100644
--- a/src/target/firmware/layer1/prim_rx_nb.c
+++ b/src/target/firmware/layer1/prim_rx_nb.c
@@ -48,6 +48,7 @@
 #include <layer1/agc.h>

 #include <l1ctl_proto.h>
+#include <layer1/l1gprs_fw.h>

 struct l1s_rxnb_state {
    struct l1s_meas_hdr meas[4];
@@ -153,7 +154,16 @@ static int l1s_nb_resp(__unused uint8_t p1, uint8_t burst_id, uint16_t p3)
        /* copy actual data, skipping the information block [0,1,2] */
        dsp_memcpy_from_api(rxnb.di->data, &dsp_api.ndb->a_cd[3], 23, 0);

-       l1_queue_for_l2(rxnb.msg);
+       if (mf_task_id == MF_TASK_GPRS_PDTCH) {
+           /* PDCH: a CS-1 block (same coding as xCCH) goes to the
+            * GPRS state machine, not up as L1CTL_DATA_IND. */
+           l1gprs_fw_dl_block(rx_time.fn, tn, rxnb.di->data,
+                      rxnb.dl->fire_crc < 2,
+                      rxnb.dl->rx_level, num_biterr, rxnb.dl->snr);
+           msgb_free(rxnb.msg);
+       } else {
+           l1_queue_for_l2(rxnb.msg);
+       }
        rxnb.msg = NULL; rxnb.dl = NULL; rxnb.di = NULL;

        /* clear downlink task */
diff --git a/src/target/firmware/include/layer1/l1gprs_fw.h b/src/target/firmware/include/layer1/l1gprs_fw.h
new file mode 100644
index 00000000..2688e3f5
--- /dev/null
+++ b/src/target/firmware/include/layer1/l1gprs_fw.h
@@ -0,0 +1,33 @@
+/* l1gprs_fw.h - GPRS (PDCH) support in the Calypso layer1 firmware.
+ *
+ * A static-allocation port of src/shared/l1gprs.c (trxcon/virtphy) to the
+ * target: TBF registry, DL block filtering by TFI, RTS generation, and the
+ * uplink block queue consumed by prim_pdtch.c.  CS-1 only: the DSP xCCH
+ * coder/decoder (DDL/DUL tasks) is bit-exact with GPRS CS-1.
+ */
+#pragma once
+
```

```
+#include <stdint.h>
+#include <stdbool.h>
+
+struct msgb;
+
+/* L1A side (l23_api.c, main loop) */
+void l1gprs_fw_reset(void);
+int  l1gprs_fw_ul_tbf_cfg(struct msgb *msg);   /* L1CTL_GPRS_UL_TBF_CFG_REQ */
+int  l1gprs_fw_dl_tbf_cfg(struct msgb *msg);   /* L1CTL_GPRS_DL_TBF_CFG_REQ */
+int  l1gprs_fw_ul_block_req(struct msgb *msg); /* L1CTL_GPRS_UL_BLOCK_REQ: takes ownership of msg */
+
+/* L1S side (TDMA interrupt) */
+/* A downlink PDTCH block was decoded (or failed: crc_ok == 0, data ignored). */
+void l1gprs_fw_dl_block(uint32_t fn, uint8_t tn, const uint8_t *data,
+           bool crc_ok, uint8_t rx_lev, uint16_t num_biterr, uint16_t snr);
+/* Uplink block to transmit at exactly (fn, tn), or NULL.  Stale blocks are dropped. */
+struct msgb *l1gprs_fw_ul_dequeue(uint32_t fn, uint8_t tn);
+/* The block for (fn, tn) went out: L1CTL_GPRS_UL_BLOCK_CNF to L23. */
+void l1gprs_fw_ul_block_cnf(uint32_t fn, uint8_t tn, const uint8_t *data);
+
+/* Frame-number advance of the RTS.ind relative to the DL block that carried
+ * the USF (13 = next-but-one block period; 13 maps a block start onto a
+ * block start of the 52-multiframe).  Polls (RRBP) are at N+13 at least. */
+#define L1GPRS_FW_RTS_ADVANCE  13
diff --git a/src/target/firmware/layer1/l1gprs_fw.c b/src/target/firmware/layer1/l1gprs_fw.c
new file mode 100644
index 00000000..301a88f4
--- /dev/null
+++ b/src/target/firmware/layer1/l1gprs_fw.c
@@ -0,0 +1,324 @@
+/* l1gprs_fw.c - GPRS (PDCH) layer-1 state for the Calypso firmware.
+ *
+ * (C) 2026 osmo-operator bench. GPLv2+.
+ * Port of src/shared/l1gprs.c (C) 2022-2024 sysmocom, Vadim Yanitskiy,
+ * to static allocation and to the firmware's L1A/L1S split.
+ */
+#include <stdint.h>
+#include <stdio.h>
+#include <string.h>
+#include <errno.h>
+
+#include <arpa/inet.h>
+#include <asm/system.h>
+
+#include <osmocom/core/msgb.h>
+#include <osmocom/core/linuxlist.h>
+#include <osmocom/gsm/gsm_utils.h>
+
+#include <layer1/l23_api.h>
+#include <layer1/l1gprs_fw.h>
+#include <l1ctl_proto.h>
+
+#define MAX_TBF        8
+#define TDMA_FN_INVALID    0xffffffff
+#define CS1_BLOCK_LEN  23
+
+struct fw_tbf {
+   bool used;
+   bool uplink;
+   bool pending;       /* waiting for start_fn */
+   uint8_t ref;
+   uint8_t slotmask;
+   uint8_t dl_tfi;
+   uint32_t start_fn;
+};
+
```

```
+static struct fw_tbf tbfs[MAX_TBF];
+/* Uplink blocks queued by L23 (L1CTL_GPRS_UL_BLOCK_REQ), fn in cb[0], tn in cb[1] */
+static struct llist_head ul_queue = LLIST_HEAD_INIT(ul_queue);
+
+/* 0 when a == b, >0 when a is after b, <0 when a is before b (hyperframe wrap) */
+static int fn_cmp(uint32_t a, uint32_t b)
+{
+   uint32_t d = (a + GSM_MAX_FN - b) % GSM_MAX_FN;
+   if (d == 0)
+       return 0;
+   return (d < GSM_MAX_FN / 2) ? 1 : -1;
+}
+
+void l1gprs_fw_reset(void)
+{
+   unsigned long flags;
+   struct msgb *msg;
+
+   local_firq_save(flags);
+   memset(tbfs, 0, sizeof(tbfs));
+   while ((msg = msgb_dequeue(&ul_queue)))
+       msgb_free(msg);
+   local_irq_restore(flags);
+}
+
+static struct fw_tbf *find_tbf(bool uplink, uint8_t ref, bool pending)
+{
+   int i;
+   for (i = 0; i < MAX_TBF; i++) {
+       if (!tbfs[i].used || tbfs[i].uplink != uplink)
+           continue;
+       if (tbfs[i].ref != ref || tbfs[i].pending != pending)
+           continue;
+       return &tbfs[i];
+   }
+   return NULL;
+}
+
+static struct fw_tbf *alloc_tbf(void)
+{
+   int i;
+   for (i = 0; i < MAX_TBF; i++)
+       if (!tbfs[i].used)
+           return &tbfs[i];
+   return NULL;
+}
+
+/* Called from both contexts; the caller holds the IRQ lock */
+static void tbf_cfg(bool uplink, uint8_t ref, uint8_t slotmask, uint8_t dl_tfi, uint32_t start_fn)
+{
+   struct fw_tbf *t;
+
+   if (slotmask == 0x00) {
+       /* release */
+       while ((t = find_tbf(uplink, ref, false)) || (t = find_tbf(uplink, ref, true)))
+           t->used = false;
+       printf("l1gprs: %cL-TBF#%u released\n", uplink ? 'U' : 'D', ref);
+       return;
+   }
+   if (start_fn != TDMA_FN_INVALID) {
+       t = find_tbf(uplink, ref, true);
+       if (!t)
+           t = alloc_tbf();
+       if (!t) {
+           puts("l1gprs: no free TBF slot (pending)\n");
+           return;
```

```
+       }
+       *t = (struct fw_tbf) { .used = true, .uplink = uplink, .pending = true, .ref = ref,
+                       .slotmask = slotmask, .dl_tfi = dl_tfi, .start_fn = start_fn };
+       printf("l1gprs: %cL-TBF#%u pending slotmask=0x%02x start_fn=%u\n",
+              uplink ? 'U' : 'D', ref, slotmask, start_fn);
+       return;
+   }
+   t = find_tbf(uplink, ref, false);
+   if (!t)
+       t = alloc_tbf();
+   if (!t) {
+       puts("l1gprs: no free TBF slot\n");
+       return;
+   }
+   *t = (struct fw_tbf) { .used = true, .uplink = uplink, .pending = false, .ref = ref,
+                   .slotmask = slotmask, .dl_tfi = dl_tfi, .start_fn = TDMA_FN_INVALID };
+   printf("l1gprs: %cL-TBF#%u active slotmask=0x%02x tfi=%u\n",
+          uplink ? 'U' : 'D', ref, slotmask, dl_tfi);
+}
+
+/* Promote pending TBFs whose starting time has come (L1S context) */
+static void check_pending(uint32_t fn)
+{
+   int i;
+   for (i = 0; i < MAX_TBF; i++) {
+       struct fw_tbf *t = &tbfs[i], *a;
+       if (!t->used || !t->pending)
+           continue;
+       if (fn_cmp(fn, t->start_fn) < 0)
+           continue;
+       /* merge into an existing active TBF of the same ref, or promote */
+       a = find_tbf(t->uplink, t->ref, false);
+       if (a) {
+           a->slotmask = t->slotmask;
+           a->dl_tfi = t->dl_tfi;
+           t->used = false;
+       } else {
+           t->pending = false;
+       }
+       printf("l1gprs: %cL-TBF#%u becomes active (fn=%u)\n", t->uplink ? 'U' : 'D', t->ref, fn);
+   }
+}
+
+static unsigned int pdch_count(uint8_t tn, bool pending, uint32_t *dl_tfi_mask)
+{
+   unsigned int n = 0;
+   int i;
+   for (i = 0; i < MAX_TBF; i++) {
+       struct fw_tbf *t = &tbfs[i];
+       if (!t->used || t->pending != pending)
+           continue;
+       if (!(t->slotmask & (1 << tn)))
+           continue;
+       n++;
+       if (dl_tfi_mask && !t->uplink)
+           *dl_tfi_mask |= (1 << t->dl_tfi);
+   }
+   return n;
+}
+
+int l1gprs_fw_ul_tbf_cfg(struct msgb *msg)
+{
+   struct l1ctl_hdr *l1h = (struct l1ctl_hdr *) msg->data;
+   struct l1ctl_gprs_ul_tbf_cfg_req *req = (void *) l1h->data;
+   unsigned long flags;
+
```

```
+   if (msg->len < sizeof(*l1h) + sizeof(*req))
+       return -EINVAL;
+   local_firq_save(flags);
+   tbf_cfg(true, req->tbf_ref, req->slotmask, 0, ntohl(req->start_fn));
+   local_irq_restore(flags);
+   return 0;
+}
+
+int l1gprs_fw_dl_tbf_cfg(struct msgb *msg)
+{
+   struct l1ctl_hdr *l1h = (struct l1ctl_hdr *) msg->data;
+   struct l1ctl_gprs_dl_tbf_cfg_req *req = (void *) l1h->data;
+   unsigned long flags;
+
+   if (msg->len < sizeof(*l1h) + sizeof(*req))
+       return -EINVAL;
+   if (req->dl_tfi > 31)
+       return -EINVAL;
+   local_firq_save(flags);
+   tbf_cfg(false, req->tbf_ref, req->slotmask, req->dl_tfi, ntohl(req->start_fn));
+   local_irq_restore(flags);
+   return 0;
+}
+
+int l1gprs_fw_ul_block_req(struct msgb *msg)
+{
+   struct l1ctl_hdr *l1h = (struct l1ctl_hdr *) msg->data;
+   struct l1ctl_gprs_ul_block_req *req = (void *) l1h->data;
+   unsigned long flags;
+   uint32_t fn;
+
+   if (msg->len < sizeof(*l1h) + sizeof(*req) + CS1_BLOCK_LEN) {
+       printf("l1gprs: short UL BLOCK.req (len=%u)\n", msg->len);
+       msgb_free(msg);
+       return -EINVAL;
+   }
+   fn = ntohl(req->hdr.fn);
+   if (req->hdr.tn > 7) {
+       msgb_free(msg);
+       return -EINVAL;
+   }
+   msg->l3h = req->data;
+   msg->cb[0] = fn;
+   msg->cb[1] = req->hdr.tn;
+   local_firq_save(flags);
+   msgb_enqueue(&ul_queue, msg);
+   local_irq_restore(flags);
+   return 0;
+}
+
+struct msgb *l1gprs_fw_ul_dequeue(uint32_t fn, uint8_t tn)
+{
+   struct msgb *msg, *tmp;
+
+   llist_for_each_entry_safe(msg, tmp, &ul_queue, list) {
+       int c = fn_cmp((uint32_t) msg->cb[0], fn);
+       if (c < 0) {
+           /* the ship has sailed */
+           printf("l1gprs: drop stale UL block fn=%u (now %u)\n", (uint32_t) msg->cb[0], fn);
+           llist_del(&msg->list);
+           msgb_free(msg);
+           continue;
+       }
+       if (c == 0 && (uint8_t) msg->cb[1] == tn) {
+           llist_del(&msg->list);
+           return msg;
```

```
+       }
+   }
+   return NULL;
+}
+
+void l1gprs_fw_ul_block_cnf(uint32_t fn, uint8_t tn, const uint8_t *data)
+{
+   struct l1ctl_gprs_ul_block_cnf *cnf;
+   struct msgb *msg = l1ctl_msgb_alloc(L1CTL_GPRS_UL_BLOCK_CNF);
+
+   cnf = (void *) msgb_put(msg, sizeof(*cnf));
+   cnf->fn = htonl(fn);
+   cnf->tn = tn;
+   if (data)
+       memcpy(msgb_put(msg, CS1_BLOCK_LEN), data, CS1_BLOCK_LEN);
+   l1_queue_for_l2(msg);
+}
+
+/* Does the (CS-1) downlink block address one of our DL TBFs? Control blocks
+ * without optional octets always pass (3GPP TS 44.060 10.2.1 / 10.3.1). */
+static bool filter_dl_block(uint32_t dl_tfi_mask, const uint8_t *data)
+{
+   uint8_t dl_tfi;
+   switch (data[0] >> 6) {
+   case 0x00:  /* RLC data block */
+       dl_tfi = (data[1] >> 1) & 0x1f;
+       break;
+   case 0x02:  /* control block with optional octets */
+       dl_tfi = (data[2] >> 1) & 0x1f;
+       break;
+   case 0x01:  /* control block, no optional octets */
+       return true;
+   default:
+       return false;
+   }
+   return (dl_tfi_mask & (1 << dl_tfi)) != 0;
+}
+
+void l1gprs_fw_dl_block(uint32_t fn, uint8_t tn, const uint8_t *data,
+           bool crc_ok, uint8_t rx_lev, uint16_t num_biterr, uint16_t snr)
+{
+   struct l1ctl_gprs_dl_block_ind *ind;
+   struct l1ctl_gprs_rts_ind *rts;
+   uint32_t dl_tfi_mask = 0;
+   uint8_t usf = 0xff;
+   struct msgb *msg;
+   uint32_t ber10k;
+
+   check_pending(fn);
+
+   if (pdch_count(tn, false, &dl_tfi_mask) == 0) {
+       /* nothing configured on this PDCH (yet): L23 does not want to hear about it */
+       return;
+   }
+
+   ber10k = (uint32_t) num_biterr * 10000 / 456;
+   if (ber10k > 10000)
+       ber10k = 10000;
+
+   msg = l1ctl_msgb_alloc(L1CTL_GPRS_DL_BLOCK_IND);
+   ind = (void *) msgb_put(msg, sizeof(*ind));
+   ind->hdr.fn = htonl(fn);
+   ind->hdr.tn = tn;
+   ind->meas.ber10k = htons(ber10k);
+   ind->meas.ci_cb = htons((int16_t) (snr > 300 ? 300 : snr));
+   ind->meas.rx_lev = rx_lev;
```

```
+   ind->usf = 0xff;
+   if (crc_ok && data) {
+       if ((fn % 104) == 12) {
+           /* PTCCH/D: pass as-is */
+           memcpy(msgb_put(msg, CS1_BLOCK_LEN), data, CS1_BLOCK_LEN);
+       } else {
+           usf = data[0] & 0x07;
+           ind->usf = usf;
+           if (filter_dl_block(dl_tfi_mask, data))
+               memcpy(msgb_put(msg, CS1_BLOCK_LEN), data, CS1_BLOCK_LEN);
+       }
+   }
+   l1_queue_for_l2(msg);
+
+   /* Ready-to-send for the uplink block L1GPRS_FW_RTS_ADVANCE frames after
+    * this one.  The USF is the one just decoded: on the bench a lone UL TBF
+    * gets its USF in every block, and polls (RRBP) are fn-exact anyway. */
+   msg = l1ctl_msgb_alloc(L1CTL_GPRS_RTS_IND);
+   rts = (void *) msgb_put(msg, sizeof(*rts));
+   rts->fn = htonl((fn + L1GPRS_FW_RTS_ADVANCE) % GSM_MAX_FN);
+   rts->tn = tn;
+   rts->usf = usf;
+   l1_queue_for_l2(msg);
+}
diff --git a/src/target/firmware/layer1/prim_pdtch.c b/src/target/firmware/layer1/prim_pdtch.c
new file mode 100644
index 00000000..bac76f3a
--- /dev/null
+++ b/src/target/firmware/layer1/prim_pdtch.c
@@ -0,0 +1,115 @@
+/* prim_pdtch.c - PDTCH uplink (CS-1, four normal bursts, DUL task) for the
+ * Calypso firmware.  Mirror of prim_tx_nb.c, but a block is only sent when
+ * L23 queued one for exactly this frame number (L1CTL_GPRS_UL_BLOCK_REQ);
+ * otherwise the uplink block period is left silent.
+ *
+ * (C) 2026 osmo-operator bench. GPLv2+. */
+#include <stdint.h>
+#include <stdio.h>
+#include <string.h>
+
+#include <defines.h>
+#include <debug.h>
+#include <memory.h>
+#include <byteorder.h>
+#include <osmocom/gsm/gsm_utils.h>
+#include <osmocom/core/msgb.h>
+#include <calypso/dsp_api.h>
+#include <calypso/irq.h>
+#include <calypso/tpu.h>
+#include <calypso/tsp.h>
+#include <calypso/dsp.h>
+#include <calypso/timer.h>
+#include <comm/sercomm.h>
+
+#include <layer1/sync.h>
+#include <layer1/agc.h>
+#include <layer1/tdma_sched.h>
+#include <layer1/mframe_sched.h>
+#include <layer1/tpu_window.h>
+#include <layer1/l23_api.h>
+#include <layer1/rfch.h>
+#include <layer1/prim.h>
+#include <layer1/l1gprs_fw.h>
+
+#include <l1ctl_proto.h>
+
```

```
+static struct {
+   uint8_t active;     /* a block is being transmitted in this period */
+   uint32_t fn;        /* its first frame number */
+   uint8_t tn;
+   uint8_t data[23];
+} pdtch_ul;
+
+static int l1s_pdtch_tx_resp(__unused uint8_t p1, uint8_t burst_id, __unused uint16_t p3)
+{
+   if (!pdtch_ul.active)
+       return 0;
+
+   putchart('p');
+   dsp_api.r_page_used = 1;
+
+   if (burst_id == 3) {
+       pdtch_ul.active = 0;
+       l1gprs_fw_ul_block_cnf(pdtch_ul.fn, pdtch_ul.tn, pdtch_ul.data);
+   }
+   return 0;
+}
+
+static int l1s_pdtch_tx_cmd(__unused uint8_t p1, uint8_t burst_id, __unused uint16_t p3)
+{
+   uint16_t arfcn;
+   uint8_t tsc, tn;
+
+   rfch_get_params(&l1s.next_time, &arfcn, &tsc, &tn);
+
+   if (burst_id == 0) {
+       uint16_t *info_ptr = dsp_api.ndb->a_cu;
+       struct msgb *msg;
+
+       msg = l1gprs_fw_ul_dequeue(l1s.next_time.fn, tn);
+       if (!msg) {
+           pdtch_ul.active = 0;
+           return 0;
+       }
+       putchart('P');
+       pdtch_ul.active = 1;
+       pdtch_ul.fn = l1s.next_time.fn;
+       pdtch_ul.tn = tn;
+       memcpy(pdtch_ul.data, msg->l3h, sizeof(pdtch_ul.data));
+       msgb_free(msg);
+
+       /* Fill data block Header */
+       info_ptr[0] = (1 << B_BLUD);     // 1st word: Set B_BLU bit.
+       info_ptr[1] = 0;                 // 2nd word: cleared.
+       info_ptr[2] = 0;                 // 3rd word: cleared.
+       /* Copy the actual data after the header */
+       dsp_memcpy_to_api(&info_ptr[3], pdtch_ul.data, 23, 0);
+   } else if (!pdtch_ul.active) {
+       return 0;
+   }
+
+   l1s_tx_apc_helper(arfcn);
+
+   /* DUL_DSP_TASK, four normal bursts: the ROM's xCCH coding is CS-1 */
+   dsp_load_tch_param(&l1s.next_time, SIG_ONLY_MODE, SDCCH_4, 0, 0, 0, tn);
+
+   dsp_load_tx_task(
+       dsp_task_iq_swap(DUL_DSP_TASK, arfcn, 1),
+       burst_id, tsc
+   );
+
+   l1s_tx_win_ctrl(arfcn | ARFCN_UPLINK, L1_TXWIN_NB, 0, 3);
```

```
+
+   return 0;
+}
+
+const struct tdma_sched_item pdtch_sched_set_ul[] = {
+   SCHED_ITEM_DT(l1s_pdtch_tx_cmd, 3, 2, 0),                      SCHED_END_FRAME(),
+   SCHED_ITEM_DT(l1s_pdtch_tx_cmd, 3, 2, 1),                      SCHED_END_FRAME(),
+   SCHED_ITEM(l1s_pdtch_tx_resp, -4, 2, 0),    SCHED_ITEM_DT(l1s_pdtch_tx_cmd, 3, 2, 2),   SCHED_END_FRAME(),
+   SCHED_ITEM(l1s_pdtch_tx_resp, -4, 2, 1),    SCHED_ITEM_DT(l1s_pdtch_tx_cmd, 3, 2, 3),   SCHED_END_FRAME(),
+                       SCHED_ITEM(l1s_pdtch_tx_resp, -4, 2, 2),    SCHED_END_FRAME(),
+                       SCHED_ITEM(l1s_pdtch_tx_resp, -4, 2, 3),    SCHED_END_FRAME(),
+   SCHED_END_SET()
+};
```

3.290 /opt/GSM/osmo-operator/patches/osmocom-bb-firmware-map-ld.patch

1449 octets, 24 lignes → 24 lignes

```
osmocom-bb firmware : -Wl,-Map,<fichier> au lieu de -Wl,-Map <fichier>

[2026-10-24] Le lien des .elf passe par gcc ($(LD) = $(CC)). « -Wl,-Map board/x.map »
ne transmet a ld QUE « -Map » ; le chemin du .map reste un argument de gcc, qui le prend
pour un fichier d entree et s arrete avant ld :
  arm-elf-gcc: board/compal_e88/hello_world.highram.map : No such file or directory
  make[1]: *** [Makefile.inc:139: board/compal_e88/hello_world.highram.elf] Error 1
Avec la virgule, option et fichier vont ensemble a ld. tools/osmo-firmwares-install.sh
(_fw_bb_corriger) le faisait par sed pour le supplement --gnuarm ; le patch couvre aussi
un « make firmware » tape a la main dans osmocom-bb/src.

diff --git a/src/target/firmware/Makefile.inc b/src/target/firmware/Makefile.inc
index 86e8f3ce..17549064 100644
--- a/src/target/firmware/Makefile.inc
+++ b/src/target/firmware/Makefile.inc
@@ -99,7 +99,7 @@ $(1)_$(2)_$(3)_OBJS+=board/$(2)/$(1).$(3).manifest.o
 # define compilation rule, also generates map file
 board/$(2)/$(1).$(3).elf board/$(2)/$(1).$(3).map: $$($(1)_$(2)_$(3)_OBJS) $$($(1)_$(2)_$(3)_LIBS)
 ↪  $$(ENV_$(3)_LDS)
    $$(Q_LD)$(CROSS_COMPILE)$(LD) $(LDFLAGS) -T $$(ENV_$(3)_LDS) -Bstatic \
-       -Wl,-Map board/$(2)/$(1).$(3).map -o board/$(2)/$(1).$(3).elf \
+       -Wl,-Map,board/$(2)/$(1).$(3).map -o board/$(2)/$(1).$(3).elf \
        -Wl,--start-group $$($(1)_$(2)_$(3)_OBJS) $$($(1)_$(2)_$(3)_LIBS) -Wl,--end-group

 # define size rule
```

3.291 /opt/GSM/osmo-operator/patches/osmocom-bb-force-rand-toy.patch

1210 octets, 30 lignes → 30 lignes

```
diff --git a/src/host/layer23/src/common/subscriber.c b/src/host/layer23/src/common/subscriber.c
index f6aa05d5..53645b04 100644
--- a/src/host/layer23/src/common/subscriber.c
+++ b/src/host/layer23/src/common/subscriber.c
@@ -16,6 +16,7 @@
  */

 #include <stdint.h>
+#include <stdlib.h>
 #include <errno.h>
 #include <string.h>
 #include <arpa/inet.h>
@@ -265,6 +266,17 @@ int gsm_subscr_generate_kc(struct osmocom_ms *ms, uint8_t key_seq, const uint8_t
    struct gsm_subscriber *subscr = &ms->subscr;
```

```
    struct osmobb_l23_subscr_sim_auth_resp_sig_data sd;
    int rc;
+   static const uint8_t forced_rand[16];
+
+   /* force RAND=0 for the fixed COMP128v1 toy table
+    * (see /root/deka/crack_toy.py) instead of the network's RAND.
+    * FORCE_RAND is a presence-only switch: its value is never read, so
+    * this can only ever force an all-zero RAND, never an arbitrary one.
+    * Unset (default): rand is untouched, normal behavior. */
+   if (getenv("FORCE_RAND")) {
+       LOGP(DMM, LOGL_NOTICE, "FORCE_RAND set: using RAND=0 (toy mode)\n");
+       rand = forced_rand;
+   }

    if (no_sim || subscr->sim_type == GSM_SIM_TYPE_NONE || !subscr->sim_valid) {
        LOGP(DMM, LOGL_INFO, "Sending dummy authentication response\n");
```

3.292 /opt/GSM/osmo-operator/patches/osmocom-bb-osmocon-kc-filter.patch

2335 octets, 62 lignes → 62 lignes

```
diff --git a/src/host/osmocon/osmocon.c b/src/host/osmocon/osmocon.c
index 547f49fe..d8921885 100644
--- a/src/host/osmocon/osmocon.c
+++ b/src/host/osmocon/osmocon.c
@@ -1281,6 +1281,57 @@ static int un_tool_read(struct osmo_fd *fd, unsigned int flags)
        c += rc;
    }

+   /* === FILTRE Kc en amont du firmware (chiffrement A5) ===
+    * Capture L1CTL_CRYPTO_REQ (mobile -> L1) et ecrit /dev/shm/calypso_kc
+    * pour qemu_wrap (cipher UL) + si_bridge/grgsm (decipher DL).
+    * Layout fichier (32o): [0..3]seq LE [4]algo [5]key_len=8 [6..13]Kc [14]key_seq(CKSN)
+    * NE TOUCHE PAS au firmware. */
+   if (length >= 10 && buf[0] == 0x15 /* L1CTL_CRYPTO_REQ */) {
+       uint8_t algo = buf[8];
+       uint8_t klen = buf[9];
+       unsigned i;
+       fprintf(stderr, "[osmocon] CRYPTO_REQ FILTRE: len=%u algo=%u key_len=%u key=",
+           length, algo, klen);
+       for (i = 10; i < length && i < 32; i++)
+           fprintf(stderr, "%02x", buf[i]);
+       fprintf(stderr, "\n");
+       if (algo) {
+           static uint32_t kc_seq = 0;
+           uint8_t kc[32];
+           /* key_seq (CKSN) : 9e octet quand key_len=9, sinon inconnu */
+           uint8_t key_seq = (klen == 9 && length >= 19) ? buf[18] : 0xff;
+           FILE *kf;
+           memset(kc, 0, sizeof(kc));
+           kc_seq++;
+           memcpy(kc, &kc_seq, 4);
+           kc[4] = algo;
+           kc[5] = 8;                  /* on stocke toujours 8 octets de Kc */
+           memcpy(kc + 6, &buf[10], 8);
+           kc[14] = key_seq;
+           kf = fopen("/dev/shm/calypso_kc", "wb");
+           if (kf) { fwrite(kc, 1, sizeof(kc), kf); fclose(kf); }
+           fprintf(stderr, "[osmocon] Kc ecrit (seq=%u algo=%u cksn=0x%02x)\n",
+               kc_seq, algo, key_seq);
+           if (klen != 8)
+               buf[9] = 8;         /* force key_len=8 (inert: corruption en aval) */
+       } else {
+           /* algo=0 : pas de chiffrement -> efface le Kc */
```

```
+           uint8_t zero[32];
+           FILE *kf;
+           memset(zero, 0, sizeof(zero));
+           kf = fopen("/dev/shm/calypso_kc", "wb");
+           if (kf) { fwrite(zero, 1, sizeof(zero), kf); fclose(kf); }
+       }
+   } else if (length >= 1 && (buf[0] == 0x05 /* DM_EST_REQ */ ||
+                  buf[0] == 0x12 /* DM_REL_REQ */)) {
+       /* nouveau canal dedie ou release -> repart en clair */
+       uint8_t zero[32];
+       FILE *kf;
+       memset(zero, 0, sizeof(zero));
+       kf = fopen("/dev/shm/calypso_kc", "wb");
+       if (kf) { fwrite(zero, 1, sizeof(zero), kf); fclose(kf); }
+   }
+
    hdlc_send_to_phone(con->server->dlci, buf, length);

    return 0;
```

3.293 /opt/GSM/osmo-operator/patches/osmocom-bb-plmn-a6-reinsert.patch

574 octets, 14 lignes → 14 lignes

```
--- a/src/host/layer23/src/mobile/gsm322.c
+++ b/src/host/layer23/src/mobile/gsm322.c
@@ -1373,6 +1373,11 @@

    memset(&plmn->plmn, 0, sizeof(plmn->plmn));

+   /* back to the switched-on state: PLMN_SEARCH_END is only handled in A0.
+    * Left in A6 (SIM re-inserted without RPLMN, e.g. SAP reconnect) the
+    * search ends unhandled and the MS stays in PLMN search forever. */
+   new_a_state(plmn, GSM322_A0_NULL);
+
    /* initiate search at cell selection */
    LOGP(DSUM, LOGL_INFO, "Search for network\n");
    LOGP(DPLMN, LOGL_INFO, "Switch on, no RPLMN, start PLMN search "
```

3.294 /opt/GSM/osmo-operator/patches/osmocom-bb-sim-plmn-from-bcd.patch

774 octets, 20 lignes → 20 lignes

```
--- a/src/host/layer23/src/common/subscriber.c
+++ b/src/host/layer23/src/common/subscriber.c
@@ -930,7 +930,7 @@
        plmn = talloc_zero(ms, struct gsm_sub_plmn_list);
        if (!plmn)
            return -ENOMEM;
-       osmo_plmn_to_bcd(&data[0], &plmn->plmn);
+       osmo_plmn_from_bcd(&data[0], &plmn->plmn);
        llist_add_tail(&plmn->entry, &subscr->plmn_list);

        LOGP(DMM, LOGL_INFO, "received PLMN selector (mcc-mnc=%s) from SIM\n",
@@ -1027,7 +1027,7 @@
        na = talloc_zero(ms, struct gsm_sub_plmn_na);
        if (!na)
            return -ENOMEM;
-       osmo_plmn_to_bcd(&data[0], &na->plmn);
+       osmo_plmn_from_bcd(&data[0], &na->plmn);
        LOGP(DMM, LOGL_INFO, "received Forbidden PLMN %s from SIM\n",
```

```
            osmo_plmn_name(&na->plmn));
        na->cause = -1; /* must have a value, but SIM stores no cause */
```

3.295 /opt/GSM/osmo-operator/patches/osmocom-bb-transceiver-dsp-error-ratelimit.patch

8832 octets, 266 lignes → 266 lignes

```
diff --git a/src/target/firmware/layer1/prim_bts.c b/src/target/firmware/layer1/prim_bts.c
index 91b18dc..910277c 100644
--- a/src/target/firmware/layer1/prim_bts.c
+++ b/src/target/firmware/layer1/prim_bts.c
@@ -139,7 +139,13 @@ l1s_bts_resp(uint8_t p1, uint8_t p2, uint16_t p3)
    struct gsm_time rx_time;

    /* Time of the command */
-   gsm_fn2gsmtime(&rx_time, l1s.current_time.fn - 2);
+   /* [2026-10-20] Banc QEMU : BTS_RX_LAG = 3 au lieu de 2. Le burst montant du mobile emule
+    * traverse l'air (anneau -> air.py -> UDP -> BSP) et arrive ~1 trame apres l'instant ou la
+    * fenetre RX de sa trame s'ouvre : 3 bursts sur 4 d'un bloc SDCCH manquaient. On lit donc le
+    * burst de la trame X une trame plus tard et on le marque X (le BSP de c54x_exe le joue a
+    * tick(X) + 1 : CALYPSO_BSP_TRX_DELTA = -2). Silicium : remettre 2. */
+#define BTS_RX_LAG 12  /* [2026-10-21] 12 : ~10 trames de marge, latence aller-retour de l air + rattrapage
 ↪  Magnetite (delta -11 cote c54x_exe, run.sh) */
+   gsm_fn2gsmtime(&rx_time, l1s.current_time.fn - BTS_RX_LAG);
        /* We're shifted in time since we RX in the 'next' frame */


@@ -165,9 +171,11 @@ l1s_bts_resp(uint8_t p1, uint8_t p2, uint16_t p3)
        bi->tn = l1s.bts.rx_start;

        /* Data (cut to 8 bits */
-       bi->toa = db->rx[1].cmd;
-       if (bi->toa > 68)
+       /* [2026-10-19] la garde testait bi->toa, tronque a 8 bits : 0x7719 passait
+        * (0x19 = 25) et j = 0xee32 sortait de la fenetre API (data abort). */
+       if (db->rx[1].cmd > 68)
            goto exit;
+       bi->toa = db->rx[1].cmd;
        for (i=0,j=(db->rx[1].cmd)<<1; i<2*88; i++,j++)
            bi->iq[i] = iq[j] >> 8;

@@ -261,6 +269,48 @@ l1s_bts_resp(uint8_t p1, uint8_t p2, uint16_t p3)
        }
    }

+   /* [2026-10-20] banc QEMU : le deuxieme intervalle (rx_start + 1), resultat pose par c54x_exe
+    * dans rx[2] / data[72..] (meme format que d[] ci-dessus : toa, pm, -, energie, -, bits) */
+   if (l1s.bts.rx_num > 1 && db->rx[2].cmd == DSP_EXT_RX_CMD_NB)
+   {
+       uint16_t *d = &db->data[72];
+       int rssi = agc_inp_dbm8_by_pm(d[1] >> 3) / 8;
+       int16_t toa = (int16_t)d[0] - 3;
+
+       db->rx[2].cmd = 0;
+       if (d[3] > 0x1000) {
+           struct msgb *msg;
+           struct l1ctl_bts_burst_nb_ind *bi;
+           int i;
+
+           msg = l1ctl_msgb_alloc(L1CTL_BTS_BURST_NB_IND);
+           if (!msg)
+               goto exit;
+
```

```
+           bi = (struct l1ctl_bts_burst_nb_ind *) msgb_put(msg, sizeof(*bi));
+           bi->fn = htonl(rx_time.fn);
+           bi->tn = (l1s.bts.rx_start + 1) & 7;
+           if (toa > -32 && toa < 32)
+               bi->toa = toa;
+           if (rssi < -110)
+               bi->rssi = -110;
+           else if (rssi < 0)
+               bi->rssi = rssi;
+           memset(bi->data, 0x00, sizeof(bi->data));
+           for (i=0; i<116; i++)
+           {
+               int sbit = 0x0008 << ((3 - (i & 3)) << 2);
+               int sword = i >> 2;
+               int dbit = 1 << (7 - (i & 7));
+               int dbyte = i >> 3;
+
+               if (d[5+sword] & sbit)
+                   bi->data[dbyte] |= dbit;
+           }
+           l1_queue_for_l2(msg);
+       }
+   }
+
 exit:
    /* We're done with this */
    dsp_api.r_page_used = 1;
@@ -289,15 +339,31 @@ l1s_bts_cmd(uint8_t p1, uint8_t p2, uint16_t p3)

    db->rx[0].cmd = 0;
    db->rx[0].data = 0x000;
+   /* [2026-10-19] rx[1] (offset du burst dans data[]) n'etait jamais remis a
+    * zero : sur le banc QEMU (c54x_exe) la page garde une copie de code ROM
+    * tant que le DSP n'y a rien ecrit, et l1s_bts_resp indexait data[] avec. */
+   db->rx[1].cmd = 0;
+   db->rx[1].data = 0x000;
+   /* [2026-10-20] banc QEMU : rx[2] est le resultat du DEUXIEME intervalle (rx_start + 1, le TCH),
+    * pose par c54x_exe (prothese montante TS1) dans data[72..] ; remis a zero a chaque trame */
+   db->rx[2].cmd = 0;
+   db->rx[2].data = 0x000;

    t3 = l1s.next_time.t3;

    if (t3 != 2)
    {
        /* We're really a frame in advance since we RX in the next frame ! */
-       t3 = t3 - 1;
-       fn_mod_26 = (l1s.next_time.fn + 2715648 - 1) % 26;
-       fn_mod_102 = (l1s.next_time.fn + 2715648 - 1) % 102;
+       /* [2026-10-21] banc QEMU : avec BTS_RX_LAG > 2 le burst joue dans cette fenetre est celui de
+        * BTS_RX_LAG - 2 trames plus tot (l1s_bts_resp le marque current_fn - BTS_RX_LAG). Le choix AB/NB
+        * se faisait sur la trame native : un RACH (t3 14..36) tombait sur une trame SDCCH -> traite en
+        * NB, jamais d'AB vers le transceiver (osmo-bts rach:rcvd = 0). On classe donc la trame recue. */
+       {
+           uint32_t fn_rx = (l1s.next_time.fn + 2715648 - 1 - (BTS_RX_LAG - 2)) % 2715648;
+           t3 = fn_rx % 51;
+           fn_mod_26 = fn_rx % 26;
+           fn_mod_102 = fn_rx % 102;
+       }

        /* Select which type of burst */
        switch (l1s.bts.type[l1s.bts.rx_start]) {
@@ -365,64 +431,69 @@ l1s_bts_cmd(uint8_t p1, uint8_t p2, uint16_t p3)
    /* TX side */
    /* ------- */
```

```
-   // FIXME put that in a loop over all TX bursts
-   static uint8_t SLOT, tn;
-   if (l1s.bts.tx_start == 5) {
-       SLOT = 3;
-       tn = 0;
-   } else {
-       SLOT = 0;
-       tn = 1;
-   }
+   /* [2026-10-20] Tous les slots de la fenetre TX, pas un seul : le slot p de la fenetre ouverte
+    * au slot TPU (tx_start + 5) & 7 part sur le TS descendant (tx_start + p) & 7 (le TX du Calypso
+    * est en avance de 3 slots). Sur le banc QEMU c54x_exe ne joue pas la fenetre multiple de la
+    * ROM : il publie lui-meme les bursts a partir de tx[p] (cmd + donnees) de cette page.
+    * Les donnees NB du slot p sont dans data[8 p .. 8 p + 7]. */
+   {
+   int p, nslots = l1s.bts.tx_num ? l1s.bts.tx_num : 1;
+   if (nslots > 8) nslots = 8;

    /* Reset all commands to dummy burst */
    for (i=0; i<8; i++)
        db->tx[i].cmd = DSP_EXT_TX_CMD_DUMMY;

-   /* Get the next burst */
-   type = trx_get_burst(l1s.next_time.fn, tn, data);
-
-   /* Program the TX commands */
-   if (t3 == 2 && (l1s.bts.type[tn] >> 1) != 2) {
-       // FIXME use dummy until TSC will be set individually
-       db->tx[SLOT].cmd = DSP_EXT_TX_CMD_DUMMY;
-   } else
-   switch (type) {
-   case BURST_FB:
-       db->tx[SLOT].cmd = DSP_EXT_TX_CMD_FB;
-       break;
-
-   case BURST_SB:
-       db->tx[SLOT].cmd = DSP_EXT_TX_CMD_SB;
-
-       sb = sb_build(
-           l1s.bts.bsic,                   /* BSIC */
-           l1s.next_time.t1,               /* T1   */
-           l1s.next_time.t2,               /* T2   */
-           (l1s.next_time.t3 - 1) / 10     /* T3'  */
-       );
-
-       db->sch[0] = sb;
-       db->sch[1] = sb >> 16;
-
-       break;
-
-   case BURST_DUMMY:
-       db->tx[SLOT].cmd = DSP_EXT_TX_CMD_DUMMY;
-       break;
-
-   case BURST_NB:
-       db->tx[SLOT].cmd = DSP_EXT_TX_CMD_NB;
-       db->tx[SLOT].data = sizeof(struct dsp_ext_db) / sizeof(uint16_t);
-
-       for (i=0; i<7; i++)
-           db->data[i] = (data[i<<1] << 8) | data[(i<<1)+1];
-       db->data[7] = data[14] << 8;
-       break;
-
-   default:
-       db->tx[SLOT].cmd = DSP_EXT_TX_CMD_DUMMY;
-       break;
```

```
+   for (p = 0; p < nslots; p++) {
+       uint8_t tn = (l1s.bts.tx_start + p) & 7;
+
+       /* Get the next burst */
+       type = trx_get_burst(l1s.next_time.fn, tn, data);
+
+       /* Program the TX commands */
+       if (t3 == 2 && (l1s.bts.type[tn] >> 1) != 2) {
+           // FIXME use dummy until TSC will be set individually
+           db->tx[p].cmd = DSP_EXT_TX_CMD_DUMMY;
+           continue;
+       }
+       switch (type) {
+       case BURST_FB:
+           db->tx[p].cmd = DSP_EXT_TX_CMD_FB;
+           break;
+
+       case BURST_SB:
+           db->tx[p].cmd = DSP_EXT_TX_CMD_SB;
+
+           sb = sb_build(
+               l1s.bts.bsic,                   /* BSIC */
+               l1s.next_time.t1,               /* T1   */
+               l1s.next_time.t2,               /* T2   */
+               (l1s.next_time.t3 - 1) / 10     /* T3'  */
+           );
+
+           db->sch[0] = sb;
+           db->sch[1] = sb >> 16;
+
+           break;
+
+       case BURST_DUMMY:
+           db->tx[p].cmd = DSP_EXT_TX_CMD_DUMMY;
+           break;
+
+       case BURST_NB:
+           db->tx[p].cmd = DSP_EXT_TX_CMD_NB;
+           db->tx[p].data = sizeof(struct dsp_ext_db) / sizeof(uint16_t) + 8 * p;
+
+           for (i=0; i<7; i++)
+               db->data[8 * p + i] = (data[i<<1] << 8) | data[(i<<1)+1];
+           db->data[8 * p + 7] = data[14] << 8;
+           break;
+
+       default:
+           db->tx[p].cmd = DSP_EXT_TX_CMD_DUMMY;
+           break;
+       }
+   }
    }

    /* Enable the task */
diff --git a/src/target/firmware/layer1/sync.c b/src/target/firmware/layer1/sync.c
index 0cf4f10..7201b23 100644
--- a/src/target/firmware/layer1/sync.c
+++ b/src/target/firmware/layer1/sync.c
@@ -270,8 +270,20 @@ static void l1_sync(void)
    /* Update AFC */
    afc_load_dsp();

+   /* [2026-10-21] banc QEMU : le DSP emule leve d_error_status a CHAQUE trame ; une ligne
+    * console par trame (21 octets x 216/s) prend les msgb du meme pool (32) que les bursts
+    * vers transceiver, et au premier retard de l'hote _talloc_zero() boucle a l'infini
+    * (BTS : « No more clock from transceiver »). On n'affiche que les changements de
+    * valeur, et un rappel toutes les 10000 occurrences. */
```

```
    if (dsp_api.ndb->d_error_status) {
-       printf("DSP Error Status: %u\n", dsp_api.ndb->d_error_status);
+       static uint16_t last_err;
+       static uint32_t n_err;
+       uint16_t err = dsp_api.ndb->d_error_status;
+
+       n_err++;
+       if (err != last_err || n_err % 10000 == 0)
+           printf("DSP Error Status: %u (%lu fois)\n", err, (unsigned long)n_err);
+       last_err = err;
        dsp_api.ndb->d_error_status = 0;
    }
```

3.296 /opt/GSM/osmo-operator/patches/osmocom-bb-transceiver-qemu-bench.patch

30756 octets, 853 lignes → 853 lignes

```
diff --git a/src/host/layer23/src/transceiver/app.h b/src/host/layer23/src/transceiver/app.h
index 7b8e25c..5e70dd6 100644
--- a/src/host/layer23/src/transceiver/app.h
+++ b/src/host/layer23/src/transceiver/app.h
@@ -26,6 +26,7 @@


 #include <stdint.h>
+#include <time.h>

 #include "l1ctl_link.h"

@@ -52,7 +53,24 @@ struct app_state

    /* Options */
    uint16_t arfcn_sync;
+   int no_sync;    /* [2026-10-19] mode BTS sans synchro sur une cellule (QEMU, pas d'air) */
+
+   /* [2026-10-21] telephone 2 (TS1, -2) : son compteur de trames est independant de celui du
+    * telephone 1 (chacun part de 0 a son boot). On mesure D = fn1 - fn2 sur les DATA_IND des deux
+    * (fn1 extrapole a l'instant du DATA_IND du 2 par le temps ecoule, 4.615 ms par trame), et on
+    * renumerote : BURST_REQ vers le 2 avec fn - D, indications du 2 avec fn + D. D est publie dans
+    * fn_delta_file pour l'air (pont/air.py) qui tague de meme les bursts du telephone 2. */
+   struct { uint32_t fn; struct timespec t; int valid; } clk[2];
+   int fn_delta, fn_delta_valid;
+   int fn_samples[64]; unsigned fn_n_samples, fn_n_total;
+   const char *fn_delta_file;
 };

+#define TRX_GSM_MAX_FN 2715648u
+/* fn du telephone 2 -> numerotation du telephone 1 (la BTS), et retour */
+#define TRX_FN_P2_TO_BTS(as, fn) ((uint32_t)(((int64_t)(fn) + (as)->fn_delta + TRX_GSM_MAX_FN) % TRX_GSM_MAX_FN))
+#define TRX_FN_BTS_TO_P2(as, fn) ((uint32_t)(((int64_t)(fn) - (as)->fn_delta + 2 * (int64_t)TRX_GSM_MAX_FN) %
 ↪  TRX_GSM_MAX_FN))
+int app_fn_delta_sample(struct app_state *as, int idx, uint32_t fn);
+

 #endif /* __TRX_APP_H__ */
diff --git a/src/host/layer23/src/transceiver/l1ctl.c b/src/host/layer23/src/transceiver/l1ctl.c
index 5bcabe3..e9af802 100644
--- a/src/host/layer23/src/transceiver/l1ctl.c
+++ b/src/host/layer23/src/transceiver/l1ctl.c
@@ -21,7 +21,9 @@
  * along with this program.  If not, see <http://www.gnu.org/licenses/>.
  */
```

```
+#include <stdio.h>
 #include <stdlib.h>
+#include <time.h>
 #include <string.h>

 #include <osmocom/core/select.h>
@@ -147,6 +149,75 @@ l1ctl_tx_bts_mode(struct l1ctl_link *l1l, uint8_t enabled, uint8_t *type,
     return l1l_send(l1l, msg);
 }

+/* [2026-10-21] mesure de D = fn1 - fn2 (voir app.h). idx 0 : telephone 1, 1 : telephone 2. */
+static void
+_fn_delta_write(struct app_state *as)
+{
+   char tmp[256];
+   FILE *f;
+
+   if (!as->fn_delta_file)
+       return;
+   snprintf(tmp, sizeof(tmp), "%s.tmp", as->fn_delta_file);
+   f = fopen(tmp, "w");
+   if (!f)
+       return;
+   fprintf(f, "%d\n", as->fn_delta);
+   fclose(f);
+   rename(tmp, as->fn_delta_file);
+}
+
+int
+app_fn_delta_sample(struct app_state *as, int idx, uint32_t fn)
+{
+   struct timespec now;
+   int64_t d, el_ns, fn1;
+   unsigned i, best = 0, best_n = 0;
+
+   clock_gettime(CLOCK_MONOTONIC, &now);
+   as->clk[idx].fn = fn;
+   as->clk[idx].t = now;
+   as->clk[idx].valid = 1;
+   if (idx != 1 || !as->clk[0].valid)
+       return 0;
+
+   /* fn du telephone 1 a cet instant : son dernier fn + les trames ecoulees depuis */
+   el_ns = (int64_t)(now.tv_sec - as->clk[0].t.tv_sec) * 1000000000LL + (now.tv_nsec - as->clk[0].t.tv_nsec);
+   if (el_ns < 0 || el_ns > 2000000000LL)
+       return 0;
+   fn1 = (int64_t)as->clk[0].fn + (el_ns + 2307692) / 4615385;
+   d = ((fn1 - (int64_t)fn) % TRX_GSM_MAX_FN + TRX_GSM_MAX_FN) % TRX_GSM_MAX_FN;
+   if (d > TRX_GSM_MAX_FN / 2)
+       d -= TRX_GSM_MAX_FN;
+   as->fn_samples[as->fn_n_samples++] = (int)d;
+   as->fn_n_total++;
+   if (as->fn_n_samples < (as->fn_delta_valid ? 64u : 16u))
+       return 0;
+
+   /* le mode des echantillons */
+   for (i = 0; i < as->fn_n_samples; i++) {
+       unsigned j, n = 0;
+       for (j = 0; j < as->fn_n_samples; j++)
+           if (as->fn_samples[j] == as->fn_samples[i])
+               n++;
+       if (n > best_n) { best_n = n; best = i; }
+   }
+   d = as->fn_samples[best];
+   as->fn_n_samples = 0;
+
```

```
+   if (!as->fn_delta_valid) {
+       as->fn_delta = (int)d; as->fn_delta_valid = 1;
+       LOGP(DL1C, LOGL_NOTICE, "Phone 2 frame offset: fn1 - fn2 = %d (%u/%u samples) -> written to %s\n",
+            as->fn_delta, best_n, 16, as->fn_delta_file ? as->fn_delta_file : "(nowhere)");
+       _fn_delta_write(as);
+   } else if (d - as->fn_delta >= 2 || as->fn_delta - d >= 2) {
+       LOGP(DL1C, LOGL_NOTICE, "Phone 2 frame offset drifted: %d -> %d (%u/64 samples)\n", as->fn_delta, (int)d,
 ↪  best_n);
+       as->fn_delta = (int)d;
+       _fn_delta_write(as);
+   }
+   return 0;
+}
+
 int
 l1ctl_tx_bts_burst_req(struct l1ctl_link *l1l,
                        uint32_t fn, uint8_t tn, struct burst_data *burst)
@@ -157,6 +228,13 @@ l1ctl_tx_bts_burst_req(struct l1ctl_link *l1l,
     if (!l1l->sync)
         return -EIO;

+   /* [2026-10-21] telephone 2 : dans SA numerotation de trames (fn - D) ; rien tant que D est inconnu */
+   if (l1l == &l1l->as->l1l[1]) {
+       if (!l1l->as->fn_delta_valid)
+           return 0;
+       fn = TRX_FN_BTS_TO_P2(l1l->as, fn);
+   }
+
     msg = _l1ctl_alloc(L1CTL_BTS_BURST_REQ);
     if (!msg)
         return -1;
@@ -184,7 +262,7 @@ _l1ctl_rx_bts_burst_nb_ind(struct l1ctl_link *l1l, struct msgb *msg)
 {
     struct l1ctl_bts_burst_nb_ind *bi;
     uint32_t fn;
-   int rc, i;
+   int rc = 0, i;
     sbit_t data[148], steal[2];
     float toa;
     int8_t rssi;
@@ -200,7 +278,13 @@ _l1ctl_rx_bts_burst_nb_ind(struct l1ctl_link *l1l, struct msgb *msg)

     fn = ntohl(bi->fn);

-   LOGP(DL1C, LOGL_INFO, "Normal Burst Indication (fn=%d)\n", fn);
+   /* [2026-10-21] telephone 2 : vers la numerotation de la BTS (fn + D) */
+   if (l1l == &l1l->as->l1l[1]) {
+       if (!l1l->as->fn_delta_valid) { rc = 0; goto exit; }
+       fn = TRX_FN_P2_TO_BTS(l1l->as, fn);
+   }
+
+   LOGP(DL1C, LOGL_INFO, "Normal Burst Indication (fn=%d tn=%d)\n", fn, bi->tn);

     toa = bi->toa * 1.0F;

@@ -244,6 +328,10 @@ _l1ctl_rx_bts_burst_ab_ind(struct l1ctl_link *l1l, struct msgb *msg)
     }

     fn = ntohl(bi->fn);
+   if (l1l == &l1l->as->l1l[1]) {
+       if (!l1l->as->fn_delta_valid) { rc = 0; goto exit; }
+       fn = TRX_FN_P2_TO_BTS(l1l->as, fn);
+   }

     rc = gsm_ab_ind_process(l1l->as, bi, data, &toa);
     if (rc < 0)
```

```
@@ -265,7 +353,7 @@ _l1ctl_rx_data_ind(struct l1ctl_link *l1l, struct msgb *msg)
 {
    struct l1ctl_info_dl *dl;
    struct l1ctl_data_ind *di;
-   int rc;
+   int rc = 0;

    dl = (struct l1ctl_info_dl *) msg->l1h;
    msg->l2h = dl->payload;
@@ -299,6 +387,8 @@ _l1ctl_rx_data_ind(struct l1ctl_link *l1l, struct msgb *msg)
    /* forward clock of fist l1ctl_link instance only */
    if (l1l == &l1l->as->l1l[0])
        rc = trx_clk_ind(l1l->trx, ntohl(dl->frame_nr));
+   /* [2026-10-21] et la mesure du decalage de trames entre les deux telephones */
+   app_fn_delta_sample(l1l->as, l1l == &l1l->as->l1l[0] ? 0 : 1, ntohl(dl->frame_nr));

 exit:
    msgb_free(msg);
@@ -380,6 +470,16 @@ l1ctl_recv(void *data, struct msgb *msg)

    case L1CTL_RESET_IND:
    case L1CTL_RESET_CONF:
+       if (l1l->as->no_sync) {
+           /* [2026-10-19] pas de cellule a ecouter : le compteur de trames
+            * du telephone fait foi, mode BTS des que la BTS a dit POWERON */
+           LOGP(DAPP, LOGL_INFO, "Reset received: --no-sync, BTS mode without cell sync.\n");
+           l1l->sync = 1;
+           if (l1l->trx->power)
+               l1ctl_tx_bts_mode(l1l, 1, l1l->trx->type, l1l->trx->bsic, l1l->trx->arfcn, l1l->trx->gain,
 ↪  l1l->tx_mask, l1l->rx_mask, l1l->trx->handover);
+           msgb_free(msg);
+           break;
+       }
        LOGP(DAPP, LOGL_INFO, "Reset received: Starting sync.\n");
        l1l->sync = 0;
        l1ctl_tx_fbsb_req(l1l, l1l->as->arfcn_sync, L1CTL_FBSB_F_FB01SB, 100, 0, CCCH_MODE_NONE);
diff --git a/src/host/layer23/src/transceiver/main.c b/src/host/layer23/src/transceiver/main.c
index a895ffc..86c8c5c 100644
--- a/src/host/layer23/src/transceiver/main.c
+++ b/src/host/layer23/src/transceiver/main.c
@@ -22,6 +22,7 @@
  */

 #include <stdlib.h>
+#include <unistd.h>
 #include <string.h>
 #include <getopt.h>
 #include <sched.h>
@@ -46,6 +47,25 @@ static int arfcn_sync = 0;
 static int daemonize = 0;
 static int rt_prio = -1;
 static int second_phone = 0;
+/* [2026-10-19] Banc QEMU (osmo-operator, start-native.sh --osmo-trx) : le
+ * telephone emule n'a aucune cellule sur laquelle se caler (FBSB), on passe
+ * en mode BTS des le reset ; un seul timeslot pour commencer (--ts-mask 0x01).
+ * [2026-10-20] --ts-mask ne touche que le masque RX : le masque TX reste 0xe3
+ * (TS 5,6,7,0,1), car prim_bts.c (l1s_bts_cmd) ne sert le TS0 BCCH que si
+ * tx_start == 5 (la fenetre TX du Calypso est decalee de 3 slots : SLOT 3 de
+ * la fenetre multiple ouverte en 5 tombe sur le TS0). Avec tx_mask 0x01,
+ * tx_start valait 0 et le firmware emettait tn 1 -> dummy burst a chaque trame
+ * (c54x_exe : « DUMMY 18000, FB 0, SB 0 »). --tx-mask pour le forcer quand meme. */
+static int no_sync = 0;
+static int ts_mask = -1;
+static int tx_mask_opt = -1;
+/* [2026-10-20] second telephone (TS1, --second-phone) : son socket L1CTL (banc QEMU : /tmp/osmocom_l2_3) */
+static const char *l1ctl2_path = "/tmp/osmocom_l2.2";
```

```
+/* [2026-10-21] ou publier fn1 - fn2 du telephone 2 (lu par pont/air.py) */
+static const char *fn_delta_file = "/tmp/osmocom_fn_delta_2";
+static const char *l1ctl_path = "/tmp/osmocom_l2";
+static int base_port = 5700;
+static int bsic_opt = -1;

 static void print_help(char *argv[])
 {
@@ -61,6 +81,14 @@ static void print_help(char *argv[])
        "  -p   --arfcn-sync-pcs   The ARFCN above is PCS\n"
        "  -2   --second-phone     Use second phone for TS 1\n"
        "  -r   --realtime PRIO    Set realtime scheduler with given prio\n"
+       "  -n   --no-sync          No cell sync (FBSB): BTS mode right after the phone reset (QEMU bench)\n"
+       "  -m   --ts-mask MASK     RX timeslot mask of phone 1 (default 0x01 = TS0 only)\n"
+       "  -M   --tx-mask MASK     TX timeslot mask of phone 1 (default 0xe3 = TS 5,6,7,0,1 : tx_start 5 serves
 ↪  TS0)\n"
+       "  -l   --l1ctl PATH       L1CTL socket of phone 1 (default /tmp/osmocom_l2)\n"
+       "  -L   --l1ctl2 PATH      L1CTL socket of phone 2, TS1 (default /tmp/osmocom_l2.2, with -2)\n"
+       "  -F   --fn-delta-file P  where to publish fn1 - fn2 of phone 2 (default /tmp/osmocom_fn_delta_2, with
 ↪  -2)\n"
+       "  -P   --base-port PORT   TRX UDP base port (default 5700)\n"
+       "  -b   --bsic BSIC        BSIC of the cell (osmo-bts-trx sends SETTSC, not SETBSIC: NCC<<3|BCC)\n"
        );
 }

@@ -78,9 +106,18 @@ static void handle_options(int argc, char **argv, struct app_state *as)
            { "arfcn-sync-pcs", 0, 0, 'p' },
            { "second-phone", 0, 0, '2' },
            { "realtime", 1, 0, 'r' },
+           { "no-sync", 0, 0, 'n' },
+           { "ts-mask", 1, 0, 'm' },
+           { "tx-mask", 1, 0, 'M' },
+           { "l1ctl", 1, 0, 'l' },
+           { "l1ctl2", 1, 0, 'L' },
+           { "fn-delta-file", 1, 0, 'F' },
+           { "base-port", 1, 0, 'P' },
+           { "bsic", 1, 0, 'b' },
+           { NULL, 0, 0, 0 },
        };

-       c = getopt_long(argc, argv, "hd:e:Dsa:p2r:",
+       c = getopt_long(argc, argv, "hd:e:Dsa:p2r:nm:M:l:L:F:P:b:",
            long_options, &option_idx);

        if (c == -1)
@@ -116,6 +153,30 @@ static void handle_options(int argc, char **argv, struct app_state *as)
        case 'r':
            rt_prio = atoi(optarg);
            break;
+       case 'n':
+           no_sync = 1;
+           break;
+       case 'm':
+           ts_mask = strtol(optarg, NULL, 0) & 0xff;
+           break;
+       case 'M':
+           tx_mask_opt = strtol(optarg, NULL, 0) & 0xff;
+           break;
+       case 'l':
+           l1ctl_path = optarg;
+           break;
+       case 'L':
+           l1ctl2_path = optarg;
+           break;
+       case 'F':
+           fn_delta_file = optarg;
```

```
+           break;
+       case 'P':
+           base_port = atoi(optarg);
+           break;
+       case 'b':
+           bsic_opt = atoi(optarg) & 0x3f;
+           break;
        default:
            fprintf(stderr, "Unknow option %s\n", optarg);
            exit(0);
@@ -148,10 +209,14 @@ int main(int argc, char *argv[])
        return -1;
    }
    handle_options(argc, argv, as);
-   if (!arfcn_sync) {
-       fprintf(stderr, "Use --arfcn-sync <ARFCN>\n");
+   if (!arfcn_sync && !no_sync) {
+       fprintf(stderr, "Use --arfcn-sync <ARFCN> (or --no-sync)\n");
        exit(0);
    }
+   as->no_sync = no_sync;
+   as->fn_delta_file = second_phone ? fn_delta_file : NULL;
+   if (second_phone)
+       unlink(fn_delta_file);  /* pas de D perime pour l'air */

    /* App state init */
    printf("%d\n", as->arfcn_sync);
@@ -168,34 +233,55 @@ int main(int argc, char *argv[])
        exit(-1);

    /* TRX interface with OpenBTS */
-   as->trx[0] = trx_alloc("127.0.0.1", 5700, as, 1);
-   if (!as->trx)
+   as->trx[0] = trx_alloc("127.0.0.1", base_port, as, 1);
+   if (!as->trx[0])
        exit(-1);
+   if (bsic_opt >= 0)
+       as->trx[0]->bsic = bsic_opt;

    /* Establish l1ctl link */
-   rv = l1l_open(&as->l1l[0], "/tmp/osmocom_l2", l1ctl_recv, &as->l1l[0]);
+   rv = l1l_open(&as->l1l[0], l1ctl_path, l1ctl_recv, &as->l1l[0]);
    if (rv)
        exit(-1);
    as->l1l[0].trx = as->trx[0];
    as->l1l[0].as = as;
    as->l1l[0].tx_mask = 0xe3; /* TS 5,6,7,0,1 */
    as->l1l[0].rx_mask = 0x01; /* TS 0 */
+   if (ts_mask >= 0)
+       as->l1l[0].rx_mask = ts_mask;
+   if (tx_mask_opt >= 0)
+       as->l1l[0].tx_mask = tx_mask_opt;
    as->trx[0]->l1l[0] = &as->l1l[0];
+   /* [2026-10-20] banc QEMU : le telephone 1 sert TOUS les intervalles de ses masques (TS0 et TS1
+    * avec 0x03) : les bursts de la BTS pour ces TS lui sont transmis et ses indications tn=1 acceptees */
+   if (!second_phone) {
+       int tn;
+       for (tn = 1; tn < 8; tn++)
+           if ((as->l1l[0].tx_mask | as->l1l[0].rx_mask) & (1 << tn))
+               as->trx[0]->l1l[tn] = &as->l1l[0];
+   }
+   printf("transceiver: l1ctl %s, TRXD base port %d, tx_mask 0x%02x rx_mask 0x%02x%s\n",
+       l1ctl_path, base_port, as->l1l[0].tx_mask, as->l1l[0].rx_mask,
+       no_sync ? ", no-sync (BTS mode at reset)" : "");

    /* Reset phone */
```

```
    l1ctl_tx_reset_req(&as->l1l[0], L1CTL_RES_T_FULL);

    if (second_phone) {
        /* Establish l1ctl link */
-       rv = l1l_open(&as->l1l[1], "/tmp/osmocom_l2.2", l1ctl_recv, &as->l1l[1]);
+       rv = l1l_open(&as->l1l[1], l1ctl2_path, l1ctl_recv, &as->l1l[1]);
        if (rv)
            exit(-1);
        as->l1l[1].trx = as->trx[0];
        as->l1l[1].as = as;
        as->l1l[1].tx_mask = 0x02; /* TS 1 */
        as->l1l[1].rx_mask = 0x02; /* TS 1 */
-       as->l1l[0].tx_mask = 0xe1; /* TS 5,6,7,0 */
-       as->l1l[0].rx_mask = 0x01; /* TS 0 */
+       /* [2026-10-20] le telephone 1 garde -M / -m s'ils ont ete donnes (banc QEMU : 0x01 / 0x01) */
+       if (tx_mask_opt < 0)
+           as->l1l[0].tx_mask = 0xe1; /* TS 5,6,7,0 */
+       if (ts_mask < 0)
+           as->l1l[0].rx_mask = 0x01; /* TS 0 */
+       printf("transceiver: second phone (TS1) on l1ctl %s\n", l1ctl2_path);
        as->trx[0]->l1l[1] = &as->l1l[1];

        /* Reset phone */
diff --git a/src/host/layer23/src/transceiver/trx.c b/src/host/layer23/src/transceiver/trx.c
index 3f51928..eb38b2a 100644
--- a/src/host/layer23/src/transceiver/trx.c
+++ b/src/host/layer23/src/transceiver/trx.c
@@ -265,11 +265,51 @@ _trx_ctrl_cmd_poweron(struct trx *trx, const char *cmd, const char *args)
 static int
 _trx_ctrl_cmd_settsc(struct trx *trx, const char *cmd, const char *args)
 {
-   LOGP(DTRX, LOGL_ERROR,
-        "TRX received SETTSC command ! "
-        "OpenBTS should be configured to send SETBSIC command !\n");
+   /* [2026-10-19] osmo-bts-trx envoie SETTSC (critique) puis SETBSIC ; le TSC
+    * est le BCC du BSIC, le firmware le derive lui-meme : on acquiesce. */
+   int tsc = atoi(args);

-   return _trx_ctrl_send_resp(trx, cmd, "%d", -1);
+   if (tsc < 0 || tsc > 7)
+       return _trx_ctrl_send_resp(trx, cmd, "%d", -1);
+
+   /* osmo-bts-trx n'envoie SETBSIC qu'en mode legacy : sans --bsic, on prend
+    * NCC 0 et BCC = TSC, sinon on verifie que le BCC donne suit le TSC. */
+   if (trx->bsic == BSIC_INVAL) {
+       trx->bsic = tsc;
+       LOGP(DTRX, LOGL_NOTICE, "SETTSC %d: no BSIC known, assuming NCC 0 -> BSIC %d (use --bsic)\n", tsc,
 ↪  trx->bsic);
+   } else if ((trx->bsic & 7) != tsc) {
+       LOGP(DTRX, LOGL_ERROR, "SETTSC %d does not match BCC of BSIC %d\n", tsc, trx->bsic);
+       return _trx_ctrl_send_resp(trx, cmd, "%d", -1);
+   } else
+       LOGP(DTRX, LOGL_INFO, "SETTSC %d accepted (BSIC %d)\n", tsc, trx->bsic);
+   return _trx_ctrl_send_resp(trx, cmd, "%d", 0);
+}
+
+/* [2026-10-19] Deux commandes d'osmo-bts-trx (>= 1.3) inconnues d'OpenBTS 2013 :
+ *  - SETFORMAT <ver> : negociation du format TRXD. Nous ne parlons que la v0 ;
+ *    reponse « RSP SETFORMAT 0 <ver> » (statut = version retenue, params =
+ *    la requete, ce que cmd_matches_rsp() d'osmo-bts compare) ;
+ *  - NOMTXPOWER (critique) : puissance nominale, « RSP NOMTXPOWER 0 <dBm> ». */
+#define TRX_NOM_TX_POWER_DBM   23
+
+static int
+_trx_ctrl_cmd_setformat(struct trx *trx, const char *cmd, const char *args)
+{
```

```
+   LOGP(DTRX, LOGL_INFO, "SETFORMAT %s requested, answering TRXD version 0\n", args);
+   return _trx_ctrl_send_resp(trx, cmd, "%d %s", 0, args);
+}
+
+static int
+_trx_ctrl_cmd_nomtxpower(struct trx *trx, const char *cmd, const char *args)
+{
+   return _trx_ctrl_send_resp(trx, cmd, "%d %d", 0, TRX_NOM_TX_POWER_DBM);
+}
+
+static int
+_trx_ctrl_cmd_rfmute(struct trx *trx, const char *cmd, const char *args)
+{
+   /* RFMUTE 0|1 (non critique) : rien a couper dans QEMU, on acquiesce */
+   return _trx_ctrl_send_resp(trx, cmd, "%d", 0);
 }

 static int
@@ -441,6 +481,9 @@ static const struct trx_cmd_handler trx_handlers[] = {
    { "POWEROFF",   _trx_ctrl_cmd_poweroff },
    { "POWERON",    _trx_ctrl_cmd_poweron },
    { "SETTSC", _trx_ctrl_cmd_settsc },
+   { "SETFORMAT",  _trx_ctrl_cmd_setformat },
+   { "NOMTXPOWER", _trx_ctrl_cmd_nomtxpower },
+   { "RFMUTE", _trx_ctrl_cmd_rfmute },
    { "SETBSIC",    _trx_ctrl_cmd_setbsic },
    { "SETPOWER",   _trx_ctrl_cmd_setpower },
    { "SETRXGAIN",  _trx_ctrl_cmd_setrxgain },
diff --git a/src/target/firmware/Makefile.inc b/src/target/firmware/Makefile.inc
index 9049898..47524dc 100644
--- a/src/target/firmware/Makefile.inc
+++ b/src/target/firmware/Makefile.inc
@@ -1,17 +1,17 @@

 #### TOOLCHAIN CONFIGURATION ####

-CROSS_COMPILE?=arm-elf-
+CROSS_COMPILE?=arm-none-eabi-

 CC=gcc
-LD=ld
+LD=gcc
 AR=ar
 SIZE=size
 OBJCOPY=objcopy

 DEBUGF=dwarf-2

-CFLAGS=-mcpu=arm7tdmi $(INCLUDES)
+CFLAGS=-mcpu=arm7tdmi -fcommon $(INCLUDES)
 CFLAGS += -Wall -Wextra -Wcast-align -Wimplicit -Wunused
 CFLAGS += -Wswitch -Wredundant-decls -Wreturn-type -Wshadow -Wnested-externs
 CFLAGS += -Wbad-function-cast -Wsign-compare -Waggregate-return
@@ -22,7 +22,7 @@ CFLAGS += -g$(DEBUGF)
 #ASFLAGS=--g$(DEBUGF) $(INCLUDES) -D__ASSEMBLY__
 ASFLAGS=$(INCLUDES) -D__ASSEMBLY__

-LDFLAGS = -nostartfiles -nostdlib -nodefaultlibs --gc-sections --cref
+LDFLAGS = -nostartfiles -nostdlib -nodefaultlibs -Wl,--gc-sections -Wl,--cref

 #### QUIET OUTPUT ####

@@ -95,8 +95,8 @@ $(1)_$(2)_$(3)_OBJS+=board/$(2)/$(1).$(3).manifest.o
 # define compilation rule, also generates map file
 board/$(2)/$(1).$(3).elf board/$(2)/$(1).$(3).map: $$($(1)_$(2)_$(3)_OBJS) $$($(1)_$(2)_$(3)_LIBS)
 ↪  $$(ENV_$(3)_LDS)
```

```
    $$(Q_LD)$(CROSS_COMPILE)$(LD) $(LDFLAGS) -T $$(ENV_$(3)_LDS) -Bstatic \
-       -Map board/$(2)/$(1).$(3).map -o board/$(2)/$(1).$(3).elf \
-       --start-group $$($(1)_$(2)_$(3)_OBJS) $$($(1)_$(2)_$(3)_LIBS) --end-group
+       -Wl,-Map,board/$(2)/$(1).$(3).map -o board/$(2)/$(1).$(3).elf \
+       -Wl,--start-group $$($(1)_$(2)_$(3)_OBJS) $$($(1)_$(2)_$(3)_LIBS) -Wl,--end-group

 # define size rule
 board/$(2)/$(1).$(3).size: board/$(2)/$(1).$(3).elf
diff --git a/src/target/firmware/include/stdint.h b/src/target/firmware/include/stdint.h
index 627403f..5997f43 100644
--- a/src/target/firmware/include/stdint.h
+++ b/src/target/firmware/include/stdint.h
@@ -19,7 +19,7 @@
 #include_next <stdint.h>
 #endif

-#ifndef __int8_t_defined
+#if (!defined(__int8_t_defined) && !defined(INT8_MAX))
 typedef signed char int8_t;
 typedef unsigned char uint8_t;

diff --git a/src/target/firmware/layer1/l23_api.c b/src/target/firmware/layer1/l23_api.c
index 6841f80..9cc87f1 100644
--- a/src/target/firmware/layer1/l23_api.c
+++ b/src/target/firmware/layer1/l23_api.c
@@ -603,12 +603,22 @@ static int l1ctl_bts_mode(struct msgb *msg)

    l1s.tx_power = ms_pwr_ctl_lvl(gsm_arfcn2band(l1s.bts.arfcn), 15);

-   l1a_mftask_set(0);
+   /* [2026-10-20] BTS_MODE est RENVOYE par l'appli hote a chaque SETRXGAIN, SETSLOT,
+    * HANDOVER/NOHANDOVER (trx.c) : osmo-bts-trx en envoie un juste apres le RACH
+    * (activation du canal). Refaire l1a_mftask_set(0) + mframe_enable() en pleine
+    * marche cassait la tache de synchro (prim_bts_sync.c) : plus de DATA_IND, donc plus
+    * d'IND CLOCK, et osmo-bts-trx sortait (« No more clock from transceiver »). Deja en
+    * mode BTS : on met a jour les parametres sans toucher aux taches. */
+   static int bts_actif;
+   if (!(bm->enabled && bts_actif))
+       l1a_mftask_set(0);
    if (bm->enabled) {
        int i;

-       mframe_enable(MF_TASK_BTS_SYNC);
-       mframe_enable(MF_TASK_BTS);
+       if (!bts_actif || !(l1s.mframe_sched.tasks_tgt & (1 << MF_TASK_BTS))) {
+           mframe_enable(MF_TASK_BTS_SYNC);
+           mframe_enable(MF_TASK_BTS);
+       }
        for (i = 0; i < 8; i++) {
            l1s.bts.type[i] = bm->type[i];
            l1s.bts.handover[i] = bm->handover[i];
@@ -648,6 +658,7 @@ static int l1ctl_bts_mode(struct msgb *msg)
        mframe_enable(MF_TASK_BCCH_NORM);
    }

+   bts_actif = bm->enabled ? 1 : 0;
    printf("BTS MODE: %u %u {%u,%u,%u,%u,%u,%u,%u,%u} "
        "TX %d..%d RX %d..%d\n", l1s.bts.bsic, l1s.bts.arfcn,
        bm->type[0], bm->type[1], bm->type[2], bm->type[3],
diff --git a/src/target/firmware/layer1/prim_bts.c b/src/target/firmware/layer1/prim_bts.c
index 91b18dc..02acc7d 100644
--- a/src/target/firmware/layer1/prim_bts.c
+++ b/src/target/firmware/layer1/prim_bts.c
@@ -139,7 +139,13 @@ l1s_bts_resp(uint8_t p1, uint8_t p2, uint16_t p3)
    struct gsm_time rx_time;
```

```
    /* Time of the command */
-   gsm_fn2gsmtime(&rx_time, l1s.current_time.fn - 2);
+   /* [2026-10-20] Banc QEMU : BTS_RX_LAG = 3 au lieu de 2. Le burst montant du mobile emule
+    * traverse l'air (anneau -> air.py -> UDP -> BSP) et arrive ~1 trame apres l'instant ou la
+    * fenetre RX de sa trame s'ouvre : 3 bursts sur 4 d'un bloc SDCCH manquaient. On lit donc le
+    * burst de la trame X une trame plus tard et on le marque X (le BSP de c54x_exe le joue a
+    * tick(X) + 1 : CALYPSO_BSP_TRX_DELTA = -2). Silicium : remettre 2. */
+#define BTS_RX_LAG 4   /* [2026-10-20 soir] 4 : deux trames de marge (delta -3 cote c54x_exe) */
+   gsm_fn2gsmtime(&rx_time, l1s.current_time.fn - BTS_RX_LAG);
        /* We're shifted in time since we RX in the 'next' frame */


@@ -165,9 +171,11 @@ l1s_bts_resp(uint8_t p1, uint8_t p2, uint16_t p3)
        bi->tn = l1s.bts.rx_start;

        /* Data (cut to 8 bits */
-       bi->toa = db->rx[1].cmd;
-       if (bi->toa > 68)
+       /* [2026-10-19] la garde testait bi->toa, tronque a 8 bits : 0x7719 passait
+        * (0x19 = 25) et j = 0xee32 sortait de la fenetre API (data abort). */
+       if (db->rx[1].cmd > 68)
            goto exit;
+       bi->toa = db->rx[1].cmd;
        for (i=0,j=(db->rx[1].cmd)<<1; i<2*88; i++,j++)
            bi->iq[i] = iq[j] >> 8;

@@ -261,6 +269,48 @@ l1s_bts_resp(uint8_t p1, uint8_t p2, uint16_t p3)
        }
    }

+   /* [2026-10-20] banc QEMU : le deuxieme intervalle (rx_start + 1), resultat pose par c54x_exe
+    * dans rx[2] / data[72..] (meme format que d[] ci-dessus : toa, pm, -, energie, -, bits) */
+   if (l1s.bts.rx_num > 1 && db->rx[2].cmd == DSP_EXT_RX_CMD_NB)
+   {
+       uint16_t *d = &db->data[72];
+       int rssi = agc_inp_dbm8_by_pm(d[1] >> 3) / 8;
+       int16_t toa = (int16_t)d[0] - 3;
+
+       db->rx[2].cmd = 0;
+       if (d[3] > 0x1000) {
+           struct msgb *msg;
+           struct l1ctl_bts_burst_nb_ind *bi;
+           int i;
+
+           msg = l1ctl_msgb_alloc(L1CTL_BTS_BURST_NB_IND);
+           if (!msg)
+               goto exit;
+
+           bi = (struct l1ctl_bts_burst_nb_ind *) msgb_put(msg, sizeof(*bi));
+           bi->fn = htonl(rx_time.fn);
+           bi->tn = (l1s.bts.rx_start + 1) & 7;
+           if (toa > -32 && toa < 32)
+               bi->toa = toa;
+           if (rssi < -110)
+               bi->rssi = -110;
+           else if (rssi < 0)
+               bi->rssi = rssi;
+           memset(bi->data, 0x00, sizeof(bi->data));
+           for (i=0; i<116; i++)
+           {
+               int sbit = 0x0008 << ((3 - (i & 3)) << 2);
+               int sword = i >> 2;
+               int dbit = 1 << (7 - (i & 7));
+               int dbyte = i >> 3;
+
+               if (d[5+sword] & sbit)
```

```
+                   bi->data[dbyte] |= dbit;
+           }
+           l1_queue_for_l2(msg);
+       }
+   }
+
 exit:
    /* We're done with this */
    dsp_api.r_page_used = 1;
@@ -289,6 +339,15 @@ l1s_bts_cmd(uint8_t p1, uint8_t p2, uint16_t p3)

    db->rx[0].cmd = 0;
    db->rx[0].data = 0x000;
+   /* [2026-10-19] rx[1] (offset du burst dans data[]) n'etait jamais remis a
+    * zero : sur le banc QEMU (c54x_exe) la page garde une copie de code ROM
+    * tant que le DSP n'y a rien ecrit, et l1s_bts_resp indexait data[] avec. */
+   db->rx[1].cmd = 0;
+   db->rx[1].data = 0x000;
+   /* [2026-10-20] banc QEMU : rx[2] est le resultat du DEUXIEME intervalle (rx_start + 1, le TCH),
+    * pose par c54x_exe (prothese montante TS1) dans data[72..] ; remis a zero a chaque trame */
+   db->rx[2].cmd = 0;
+   db->rx[2].data = 0x000;

    t3 = l1s.next_time.t3;

@@ -365,64 +424,69 @@ l1s_bts_cmd(uint8_t p1, uint8_t p2, uint16_t p3)
    /* TX side */
    /* ------- */

-   // FIXME put that in a loop over all TX bursts
-   static uint8_t SLOT, tn;
-   if (l1s.bts.tx_start == 5) {
-       SLOT = 3;
-       tn = 0;
-   } else {
-       SLOT = 0;
-       tn = 1;
-   }
+   /* [2026-10-20] Tous les slots de la fenetre TX, pas un seul : le slot p de la fenetre ouverte
+    * au slot TPU (tx_start + 5) & 7 part sur le TS descendant (tx_start + p) & 7 (le TX du Calypso
+    * est en avance de 3 slots). Sur le banc QEMU c54x_exe ne joue pas la fenetre multiple de la
+    * ROM : il publie lui-meme les bursts a partir de tx[p] (cmd + donnees) de cette page.
+    * Les donnees NB du slot p sont dans data[8 p .. 8 p + 7]. */
+   {
+   int p, nslots = l1s.bts.tx_num ? l1s.bts.tx_num : 1;
+   if (nslots > 8) nslots = 8;

    /* Reset all commands to dummy burst */
    for (i=0; i<8; i++)
        db->tx[i].cmd = DSP_EXT_TX_CMD_DUMMY;

-   /* Get the next burst */
-   type = trx_get_burst(l1s.next_time.fn, tn, data);
-
-   /* Program the TX commands */
-   if (t3 == 2 && (l1s.bts.type[tn] >> 1) != 2) {
-       // FIXME use dummy until TSC will be set individually
-       db->tx[SLOT].cmd = DSP_EXT_TX_CMD_DUMMY;
-   } else
-   switch (type) {
-   case BURST_FB:
-       db->tx[SLOT].cmd = DSP_EXT_TX_CMD_FB;
-       break;
-
-   case BURST_SB:
-       db->tx[SLOT].cmd = DSP_EXT_TX_CMD_SB;
```

```
-
-       sb = sb_build(
-           l1s.bts.bsic,                   /* BSIC */
-           l1s.next_time.t1,               /* T1   */
-           l1s.next_time.t2,               /* T2   */
-           (l1s.next_time.t3 - 1) / 10     /* T3'  */
-       );
-
-       db->sch[0] = sb;
-       db->sch[1] = sb >> 16;
-
-       break;
-
-   case BURST_DUMMY:
-       db->tx[SLOT].cmd = DSP_EXT_TX_CMD_DUMMY;
-       break;
-
-   case BURST_NB:
-       db->tx[SLOT].cmd = DSP_EXT_TX_CMD_NB;
-       db->tx[SLOT].data = sizeof(struct dsp_ext_db) / sizeof(uint16_t);
-
-       for (i=0; i<7; i++)
-           db->data[i] = (data[i<<1] << 8) | data[(i<<1)+1];
-       db->data[7] = data[14] << 8;
-       break;
-
-   default:
-       db->tx[SLOT].cmd = DSP_EXT_TX_CMD_DUMMY;
-       break;
+   for (p = 0; p < nslots; p++) {
+       uint8_t tn = (l1s.bts.tx_start + p) & 7;
+
+       /* Get the next burst */
+       type = trx_get_burst(l1s.next_time.fn, tn, data);
+
+       /* Program the TX commands */
+       if (t3 == 2 && (l1s.bts.type[tn] >> 1) != 2) {
+           // FIXME use dummy until TSC will be set individually
+           db->tx[p].cmd = DSP_EXT_TX_CMD_DUMMY;
+           continue;
+       }
+       switch (type) {
+       case BURST_FB:
+           db->tx[p].cmd = DSP_EXT_TX_CMD_FB;
+           break;
+
+       case BURST_SB:
+           db->tx[p].cmd = DSP_EXT_TX_CMD_SB;
+
+           sb = sb_build(
+               l1s.bts.bsic,                   /* BSIC */
+               l1s.next_time.t1,               /* T1   */
+               l1s.next_time.t2,               /* T2   */
+               (l1s.next_time.t3 - 1) / 10     /* T3'  */
+           );
+
+           db->sch[0] = sb;
+           db->sch[1] = sb >> 16;
+
+           break;
+
+       case BURST_DUMMY:
+           db->tx[p].cmd = DSP_EXT_TX_CMD_DUMMY;
+           break;
+
+       case BURST_NB:
```

```
+           db->tx[p].cmd = DSP_EXT_TX_CMD_NB;
+           db->tx[p].data = sizeof(struct dsp_ext_db) / sizeof(uint16_t) + 8 * p;
+
+           for (i=0; i<7; i++)
+               db->data[8 * p + i] = (data[i<<1] << 8) | data[(i<<1)+1];
+           db->data[8 * p + 7] = data[14] << 8;
+           break;
+
+       default:
+           db->tx[p].cmd = DSP_EXT_TX_CMD_DUMMY;
+           break;
+       }
+   }
    }

    /* Enable the task */
diff --git a/src/target/firmware/layer1/prim_bts_sync.c b/src/target/firmware/layer1/prim_bts_sync.c
index 60cb157..3ea69e5 100644
--- a/src/target/firmware/layer1/prim_bts_sync.c
+++ b/src/target/firmware/layer1/prim_bts_sync.c
@@ -166,17 +166,17 @@ static int l1s_nb_resp(__unused uint8_t p1, __unused uint8_t p2, uint16_t p3)
    rxnb.meas[0].snr = dsp_api.db_r->a_serv_demod[D_SNR];

    /* feed computed frequency error into AFC loop */
-   if (rxnb.meas[0].snr > AFC_SNR_THRESHOLD)
-       afc_input(rxnb.meas[0].freq_err, rf_arfcn, 1);
-   else
-       afc_input(rxnb.meas[0].freq_err, rf_arfcn, 0);
+   /* [2026-10-20] banc QEMU (c54x_exe) : NI AFC NI boucle de TA. Pas de cellule a suivre
+    * (transceiver --no-sync) ; quand un burst du mobile tombait dans cette fenetre TS0
+    * (freq_err=3967 mesure) les deux boucles decalaient frequence et TPU, la tache de
+    * synchro ne rendait plus rien, plus de DATA_IND donc plus d IND CLOCK et osmo-bts-trx
+    * sortait (« No more clock from transceiver »). On garde la mesure et le DATA_IND. */

    /* feed computed TOA into TA loop */
    printf("TOA=%d snr=%d (thres %d) freq_err=%d pm=%d\n",
        rxnb.meas[0].toa_qbit, rxnb.meas[0].snr, 2560,
        rxnb.meas[0].freq_err, rxnb.meas[0].pm_dbm8 / 8);
-   for (i = 0; i < 10; i++)
-       toa_input(rxnb.meas[0].toa_qbit << 2, rxnb.meas[0].snr);
+   (void)i;

    /* Tell the RF frontend to set the gain appropriately */
    rffe_compute_gain(rxnb.meas[0].pm_dbm8/8, CAL_DSP_TGT_BB_LVL);
@@ -225,18 +225,18 @@ static int l1s_nb_resp(__unused uint8_t p1, __unused uint8_t p2, uint16_t p3)
    rxnb.meas[0].snr = d[D_SNR];

    /* feed computed frequency error into AFC loop */
-   if (rxnb.meas[0].snr > AFC_SNR_THRESHOLD)
-       afc_input(rxnb.meas[0].freq_err, rf_arfcn, 1);
-   else
-       afc_input(rxnb.meas[0].freq_err, rf_arfcn, 0);
+   /* [2026-10-20] banc QEMU (c54x_exe) : NI AFC NI boucle de TA. Pas de cellule a suivre
+    * (transceiver --no-sync) ; quand un burst du mobile tombait dans cette fenetre TS0
+    * (freq_err=3967 mesure) les deux boucles decalaient frequence et TPU, la tache de
+    * synchro ne rendait plus rien, plus de DATA_IND donc plus d IND CLOCK et osmo-bts-trx
+    * sortait (« No more clock from transceiver »). On garde la mesure et le DATA_IND. */

    /* feed computed TOA into TA loop */
    printf("arfcn=%d tn=%d TOA=%d snr=%d (thres %d) freq_err=%d pm=%d\n",
        rf_arfcn, tn,
        rxnb.meas[0].toa_qbit, rxnb.meas[0].snr, 2560,
        rxnb.meas[0].freq_err, rxnb.meas[0].pm_dbm8 / 8);
-   for (i = 0; i < 10; i++)
-       toa_input(rxnb.meas[0].toa_qbit << 2, rxnb.meas[0].snr);
+   (void)i;
```

```
    /* Tell the RF frontend to set the gain appropriately */
    rffe_compute_gain(rxnb.meas[0].pm_dbm8/8, CAL_DSP_TGT_BB_LVL);
diff --git a/src/target/firmware/solve_envs.py b/src/target/firmware/solve_envs.py
index d6414d6..da1b0e1 100755
--- a/src/target/firmware/solve_envs.py
+++ b/src/target/firmware/solve_envs.py
@@ -1,4 +1,4 @@
-#!/usr/bin/env python
+#!/usr/bin/env python3

 import sys

@@ -34,7 +34,7 @@ def main(name, board_envs, app_envs):
     envs = solve(board_envs, app_envs)

     # Result
-    print ' '.join(envs)
+    print(' '.join(envs))


 if __name__ == '__main__':
```

3.297 /opt/GSM/osmo-operator/patches/pmaports-linux-postmarketos-stable-ppp.patch

1769 octets, 30 lignes → 30 lignes

```
diff --git a/device/main/linux-postmarketos-stable/APKBUILD b/device/main/linux-postmarketos-stable/APKBUILD
index 4ae7783d82..4146fa5a7b 100644
--- a/device/main/linux-postmarketos-stable/APKBUILD
+++ b/device/main/linux-postmarketos-stable/APKBUILD
@@ -133,5 +133,5 @@ a4fa95d30c48aeca7e082e1a65a7cb296f1d4561d5ef41d6ff22008d30cb7db0f513be3bc9fa8229

 ↪  e44dd245093beb04c3242273440b20bd5d1a43ee6b1cb74231f7cbd6daa942c573e130816dd5cf645c0ecc6186ee9d13048ebe7c1e43d15e0f21ea3f952
 ↪  config-stable.ppc64le

 ↪  25dc70e3cddf21d95e3267a8fb59804d77eb90f3eac55b81a5a267e9300edf12ca6603a5ac57a7f1b35780275dac23851e003a0ebfda7c2ce5de448819c
 ↪  config-stable.riscv64

 ↪  85b208a8c6d40080f050bb5fdfc63c20b0aa3beabad3166f82aed57441afee58996aa36793cb662bf669eac71efd6f2e91f29433faf3dfa642dab4c50e6
 ↪  config-stable.x86
-
 ↪  e50bea8bcefd1d25d64f662376799c50f6553a7c4881b041751c48d71756494a4506713aca92d1dd08645a69c296a1889784355424be2486be1089181b9
 ↪  config-stable.x86_64
]
 ↪  +1335c280137718cd2b38f6e588efbf7f2af55c1b7efc82ec603372fa950863e0a78dce855ba3d210a2594ee80db9cba0076e964d94239386f04417a9ae
 ↪  config-stable.x86_64
 "
diff --git a/device/main/linux-postmarketos-stable/config-stable.x86_64
 ↪  b/device/main/linux-postmarketos-stable/config-stable.x86_64
index 653356b9de..ca0c576c81 100644
--- a/device/main/linux-postmarketos-stable/config-stable.x86_64
+++ b/device/main/linux-postmarketos-stable/config-stable.x86_64
@@ -2841,7 +2841,13 @@ CONFIG_ACPI_MDIO=y
 # CONFIG_PCS_XPCS is not set
 # end of PCS device drivers

-# CONFIG_PPP is not set
+CONFIG_PPP=m
+CONFIG_PPP_BSDCOMP=m
+CONFIG_PPP_DEFLATE=m
+CONFIG_PPP_FILTER=y
+CONFIG_PPP_MULTILINK=y
```

```
+CONFIG_PPP_ASYNC=m
+CONFIG_PPP_SYNC_TTY=m
 # CONFIG_SLIP is not set
 CONFIG_USB_NET_DRIVERS=y
 # CONFIG_USB_CATC is not set
```

3.298 /opt/GSM/osmo-operator/patches/pmbootstrap-osmo-bench-qemu.patch

16351 octets, 341 lignes → 341 lignes

```
diff --git a/pmb/chroot/shutdown.py b/pmb/chroot/shutdown.py
index b49aaf7f..a3e307b8 100644
--- a/pmb/chroot/shutdown.py
+++ b/pmb/chroot/shutdown.py
@@ -19,6 +19,11 @@ def kill_adb() -> None:
     with closing(socket.socket(socket.AF_INET, socket.SOCK_STREAM)) as sock:
         sock.settimeout(2)
         if sock.connect_ex(("127.0.0.1", port)) == 0:
+            # [2026-09-07] osmo-operator: on this bench 127.0.0.1:5038 is
+            # Asterisk's AMI, not an adb server, and the native chroot has no
+            # adb - the kill would fail with 127 and abort every build.
+            if not (Chroot.native() / "usr/bin/adb").exists():
+                return
             pmb.chroot.root(["adb", "-P", str(port), "kill-server"])


diff --git a/pmb/commands/qemu.py b/pmb/commands/qemu.py
index 72b46c60..d2aadc99 100644
--- a/pmb/commands/qemu.py
+++ b/pmb/commands/qemu.py
@@ -61,6 +61,34 @@ def which_qemu(arch: Arch) -> str:
         )


+def _host_gpu_vendors() -> set[str]:
+    """PCI vendor ids of the host's DRM cards ("0x1002" = AMD, "0x10de" = NVIDIA)."""
+    vendors: set[str] = set()
+    drm = "/sys/class/drm"
+    try:
+        for card in os.listdir(drm):
+            vp = os.path.join(drm, card, "device", "vendor")
+            if card.startswith("card") and os.path.isfile(vp):
+                with open(vp) as fh:
+                    vendors.add(fh.read().strip().lower())
+    except OSError:
+        pass
+    return vendors
+
+
+def _host_cpu_has_virt() -> bool:
+    """True when the CPU exposes VT-x (vmx) or AMD-V (svm) - i.e. it is enabled in the firmware."""
+    try:
+        with open("/proc/cpuinfo") as fh:
+            for line in fh:
+                if line.startswith("flags"):
+                    flags = line.split(":", 1)[1].split()
+                    return "vmx" in flags or "svm" in flags
+    except OSError:
+        pass
+    return False
+
+
 def command_qemu(
     config: Config,
```

```
     arch: Arch,
@@ -160,7 +188,12 @@ def command_qemu(
     else:
         command += ["stdio"]

-    command += ["-drive", f"file={img_path},format=raw,if=virtio"]
+    # [2026-09-09] bootindex=0 on the root disk: with a fresh NVRAM, OVMF still
+    # ranks its own "UEFI PXEv4" (built-in network stack, no option ROM needed)
+    # ahead of the disk and spends a while in PXE before booting. QEMU's
+    # fw_cfg boot order, which OVMF honours, puts the disk first.
+    command += ["-drive", f"file={img_path},format=raw,if=none,id=osmo-root"]
+    command += ["-device", "virtio-blk-pci,drive=osmo-root,bootindex=0"]
     if img_path_2nd:
         command += ["-drive", f"file={img_path_2nd}" + ",format=raw,if=virtio"]

@@ -169,8 +202,23 @@ def command_qemu(
     else:
         command += ["-device", "virtio-mouse-pci"]
     command += ["-device", "virtio-keyboard-pci"]
-    command += ["-netdev", f"user,id=net,hostfwd=tcp:127.0.0.1:{port_ssh}-:22"]
-    command += ["-device", "virtio-net-pci,netdev=net"]
+    command += [
+        "-netdev",
+        # Plan d'adressage explicite du reseau utilisateur (slirp). Sans ces
+        # options QEMU prend des valeurs implicites et l'invite peut se
+        # retrouver sans bail DHCP : plus d'Internet ET plus de SSH, car le
+        # hostfwd n'a alors aucune IP invite ou livrer le trafic.
+        #   net=       le sous-reseau vu par l'invite
+        #   host=      la passerelle (QEMU lui-meme)
+        #   dhcpstart= la premiere IP distribuee -> l'invite est toujours .15
+        #   dns=       le resolveur interne de QEMU, qui relaie vers l'hote
+        f"user,id=net,net=10.0.2.0/24,host=10.0.2.2,dhcpstart=10.0.2.15,"
+        f"dns=10.0.2.3,hostfwd=tcp:127.0.0.1:{port_ssh}-:22",
+    ]
+    # romfile= drops the card's PXE option ROM. With it, a freshly created
+    # NVRAM ranks "UEFI PXEv4" ahead of the disk: the firmware runs iPXE,
+    # fails to find a server, and only then boots - a detour on every start.
+    command += ["-device", "virtio-net-pci,netdev=net,romfile="]

     # Check that arch is supported by pmb qemu
     if arch not in [
@@ -200,13 +248,48 @@ def command_qemu(
         case _:
             pass

+    # [2026-09-08] OPENGL ONLY WHEN IT CAN WORK. virtio-vga-gl + gl=on hands the
+    # guest's rendering to the host's Mesa as seen from this musl chroot. On an
+    # NVIDIA host there is no driver: EGL fails, QEMU falls back to software
+    # rendering and only warns. On an AMD host Mesa DOES find radeonsi, virgl
+    # engages for real against a libdrm/LLVM that do not match the host, and
+    # QEMU aborts or segfaults at launch - the usual "crashes on AMD".
+    # OSMO_PMOS_GL=0 turns GL off (virtio-vga, gl=off) ; unset, it is off on
+    # AMD hosts and on hosts without /dev/dri ; OSMO_PMOS_GL=1 forces it on.
+    gl_env = os.environ.get("OSMO_PMOS_GL", "")
+    if gl_env == "0":
+        use_qemu_gl = False
+    elif gl_env != "1" and use_qemu_gl:
+        vendors = _host_gpu_vendors()
+        if "0x1002" in vendors:
+            logging.info("osmo: AMD GPU on the host - QEMU without OpenGL (OSMO_PMOS_GL=1 to force it)")
+            use_qemu_gl = False
+        elif not os.path.exists("/dev/dri"):
+            logging.info("osmo: no /dev/dri on the host - QEMU without OpenGL")
+            use_qemu_gl = False
+
     # Configure display
```

```
     if qemu_display != "none":
+        # [2026-09-07] The screen's shape decides the phone's shape. Phosh lays
+        # itself out from the screen it is given: tall and narrow is a phone,
+        # wide is a tablet - same image, same session, two form factors. QEMU's
+        # virtio card takes that size as xres/yres, and the guest boots
+        # straight into it, so it is picked at launch and never changes under
+        # the running session. OSMO_PMOS_RES="<width>x<height>", empty or
+        # unset for whatever the card would pick on its own.
+        res = os.environ.get("OSMO_PMOS_RES", "")
+        geom = ""
+        if res:
+            try:
+                xres, yres = (int(v) for v in res.lower().split("x", 1))
+                geom = f",xres={xres},yres={yres}"
+            except ValueError:
+                logging.warning(f"WARNING: OSMO_PMOS_RES={res} is not <width>x<height>, ignored")
         match arch:
             case Arch.x86_64:
-                command += ["-device", "virtio-vga-gl"]
+                command += ["-device", f"{'virtio-vga-gl' if use_qemu_gl else 'virtio-vga'}{geom}"]
             case Arch.aarch64 | Arch.armv7:
-                command += ["-device", "virtio-gpu-gl"]
+                command += ["-device", "virtio-gpu-gl" if use_qemu_gl else "virtio-gpu-pci"]
             case Arch.riscv64 | Arch.ppc64le | Arch.loongarch64:
                 command += ["-device", "virtio-gpu-pci"]
             case _:
@@ -221,10 +304,22 @@ def command_qemu(
                 f"if=pflash,format=raw,readonly=on,file={edk2_chroot}/usr/share/AAVMF/AAVMF_CODE.fd",
             ]
         case Arch.x86_64:
+            # OVMF.fd on its own is read-only, so the firmware has nowhere to
+            # write its variables: it records no boot entry and drops into its
+            # setup menu instead of booting. Give it the code read-only and a
+            # writable copy of the variables, one per device.
+            ovmf = edk2_chroot / "usr/share/OVMF"
+            nvram = Chroot.native() / "home/pmos/rootfs" / f"{config.device}-ovmf-vars.fd"
+            if not nvram.exists():
+                pmb.helpers.run.root(["cp", ovmf / "OVMF_VARS.fd", nvram])
+            # QEMU runs as the local user, the copy is owned by root.
+            if not os.access(nvram, os.W_OK):
+                pmb.helpers.run.root(["chmod", "666", nvram])
             command += [
                 "-drive",
-                f"if=pflash,format=raw,readonly=on,file={edk2_chroot}/usr/share/OVMF/OVMF.fd",
+                f"if=pflash,unit=0,format=raw,readonly=on,file={ovmf}/OVMF_CODE.fd",
             ]
+            command += ["-drive", f"if=pflash,unit=1,format=raw,file={nvram}"]
         case Arch.loongarch64:
             command += [
                 "-drive",
@@ -243,11 +338,29 @@ def command_qemu(

     # Kernel Virtual Machine (KVM) support
     native = pmb.parse.deviceinfo().arch.is_native()
-    if use_qemu_kvm and native and os.path.exists("/dev/kvm"):
+    # [2026-09-08] KVM ONLY WHEN IT REALLY WORKS. /dev/kvm can exist while the
+    # CPU's virtualisation is off in the firmware (SVM on AMD boards ships off
+    # more often than not) or the node is not writable by this user : QEMU then
+    # dies at once ("Could not access KVM kernel module", "failed to initialize
+    # kvm: Operation not supported"). We check the flag and the access first,
+    # and fall back to TCG with -cpu max (the guest still boots, slower).
+    # OSMO_PMOS_KVM=0 forces TCG.
+    kvm_ok = (
+        use_qemu_kvm
+        and native
+        and os.environ.get("OSMO_PMOS_KVM", "1") != "0"
```

```
+        and os.access("/dev/kvm", os.R_OK | os.W_OK)
+        and _host_cpu_has_virt()
+    )
+    if kvm_ok:
          command += ["-enable-kvm"]
          command += ["-cpu", "host"]
     else:
+        if use_qemu_kvm and native and os.path.exists("/dev/kvm"):
+            logging.info("osmo: /dev/kvm is there but unusable (virtualisation off in the firmware ? not writable
 ↪  ?) - TCG")
          logging.info("WARNING: QEMU is not using KVM and will run slower!")
+        if arch == Arch.x86_64 and not qemu_cpu:
+            command += ["-cpu", "max"]

     if qemu_cpu:
         command += ["-cpu", qemu_cpu]
@@ -255,6 +368,19 @@ def command_qemu(
     display = qemu_display
     if display != "none":
         display += ",show-cursor=on,gl=" + ("on" if use_qemu_gl else "off")
+    # [2026-10-14] A GTK WINDOW THAT FITS THE SCREEN. The phone is 720x1440 and
+    # most monitors are 1080 high: without zoom-to-fit the GTK drawing area
+    # asks for the full guest size, the window manager clamps the window and
+    # the bottom of the phone is simply cut off. zoom-to-fit scales the frame
+    # (and the pointer with it) to whatever size the window gets. GTK is also
+    # the only display that leaves the host GPU alone when GL is off: the SDL
+    # display loads this chroot's Mesa and opens /dev/dri/renderD128 whatever
+    # its renderer, which is where the on-screen artefacts came from.
+    # OSMO_PMOS_GTK_OPTS replaces the extra options (empty: none).
+    if display.startswith("gtk"):
+        gtk_opts = os.environ.get("OSMO_PMOS_GTK_OPTS", "zoom-to-fit=on")
+        if gtk_opts:
+            display += "," + gtk_opts

     # Separate -show-cursor option is deprecated. If your host qemu fails here,
     # it's old (#1995).
@@ -264,9 +390,119 @@ def command_qemu(
     if qemu_audio:
         command += ["-audio", f"{qemu_audio},model=hda"]

+    command += osmo_bench_args(qemu_audio)
+
     return (command, env)


+def osmo_bench_args(qemu_audio: str | None) -> list[str]:
+    """Extra devices that put the VM on the osmocom bench: the modem, and the
+    voice bridge.
+
+    The modem (tools/osmo-phonesim-banc.py) is a PCI serial port, not ISA and
+    not USB. The UEFI firmware treats an ISA serial port as a console and
+    mistakes the modem's replies for keypresses, so it never boots; usb-serial
+    would need ftdi_sio, which this kernel doesn't have. 8250_pci is built in,
+    so the port shows up on its own as /dev/ttyS1, which is what ModemManager
+    probes. QEMU only listens here - the modem connects afterwards, once the
+    firmware is done booting (see osmo-pmos-setup).
+
+    There are TWO sound cards, because QEMU cannot push audio into a guest's
+    speaker: of an emulated card, playback goes out to a host sink and capture
+    comes in from a host source. The call's downlink lives on the host, so it
+    can only enter through the guest's capture side - never through its
+    speaker. One card alone therefore gives the guest the network's point of
+    view (it hears the far end on its microphone), not a handset's.
+
+    So: the "bridge" card is wired to the bench's voice path - gsm_audio
+    carries the downlink (RTP from osmo-mgw), so its monitor feeds that card's
```

```
+    capture; gsm_mic carries the uplink, so it takes what that card plays. The
+    "handset" card stays on the host's default devices, the real speakers and
+    microphone. Crossing the two inside the guest (see osmo-pmos-setup) puts
+    the VM in the middle of the voice path for real.
+
+    The slots are pinned so the guest can tell them apart by name alone:
+    0000:00:12.0 is the bridge, 0000:00:13.0 the handset.
+
+    Beware the pa backend: it looks for PulseAudio's pidfile, which doesn't
+    exist when PipeWire provides the service - there is only the socket, so we
+    name it explicitly. And an unknown backend doesn't error out, it segfaults;
+    this qemu only knows none, alsa, pa, sdl and wav.
+    """
+    port = os.environ.get("OSMO_PMOS_AT_PORT", "12346")
+    args: list[str] = [
+        "-chardev",
+        f"socket,id=gsmchr,host=127.0.0.1,port={port},server=on,wait=off",
+    ]
+    # [2026-09-08] virtio-serial, not a 16550. QEMU paces its emulated 16550
+    # at the programmed baud rate (115200 -> ~11 KB/s each way), so PPP data
+    # over the modem port crawled and every page filled a minute of queue.
+    # A virtconsole port has no baud rate: it shows up as /dev/hvc0 in the
+    # guest, a tty like any other, and ModemManager probes it thanks to the
+    # udev rule osmo-pmos-setup installs. OSMO_PMOS_SERIAL=pci brings the
+    # 16550 (ttyS1) back.
+    if os.environ.get("OSMO_PMOS_SERIAL", "virtio") == "pci":
+        args += ["-device", "pci-serial,chardev=gsmchr"]
+    else:
+        # Two ports, like a real modem: hvc0 is the command port (AT, URCs,
+        # calls, SMS), hvc1 the data port where ModemManager dials ATD*99 and
+        # runs PPP - so the phone keeps an AT channel while data is up. The
+        # data port listens on OSMO_PMOS_AT_PORT + 1 (12347 by default).
+        port_data = int(port) + 1
+        args += [
+            "-chardev",
+            f"socket,id=gsmdata,host=127.0.0.1,port={port_data},server=on,wait=off",
+            "-device",
+            "virtio-serial-pci,id=osmovser",
+            "-device",
+            "virtconsole,chardev=gsmchr,name=osmo.modem",
+            "-device",
+            "virtconsole,chardev=gsmdata,name=osmo.data",
+        ]
+
+    # --audio wins if it was asked for explicitly: it adds a card of its own.
+    if qemu_audio:
+        return args
+
+    sock = Path(os.environ.get("XDG_RUNTIME_DIR", f"/run/user/{os.getuid()}")) / "pulse/native"
+    if not sock.is_socket():
+        logging.warning(f"WARNING: no PulseAudio socket ({sock}), VM without sound")
+        return args
+
+    out = os.environ.get("OSMO_PMOS_SINK", "gsm_mic")
+    src = os.environ.get("OSMO_PMOS_SOURCE", "gsm_audio.monitor")
+    # [2026-09-09] The handset card is NAMED too, when the launcher says so:
+    # left to "the host's defaults", it landed on whatever was default when
+    # QEMU connected - the raw card, since the echo canceller (osmo_hp_ec /
+    # osmo_mic_ec) only came later with osmo-pmos-setup - and then had to be
+    # moved. osmo-pmos-qemu now loads the canceller first and passes its
+    # names here (OSMO_PMOS_HP / OSMO_PMOS_MIC); unset, the defaults apply.
+    combine = f"pa,id=combine,server={sock}"
+    hp = os.environ.get("OSMO_PMOS_HP", "")
+    mic = os.environ.get("OSMO_PMOS_MIC", "")
+    if hp:
+        combine += f",out.name={hp}"
```

```
+    if mic:
+        combine += f",in.name={mic}"
+    args += [
+        # The bridge: the call itself.
+        "-audiodev",
+        f"pa,id=gsm,server={sock},out.name={out},in.name={src}",
+        "-device",
+        "intel-hda,id=hdagsm,addr=0x12",
+        "-device",
+        "hda-duplex,bus=hdagsm.0,audiodev=gsm",
+        # The handset: the host's own speakers and microphone.
+        "-audiodev",
+        combine,
+        "-device",
+        "intel-hda,id=hdacombine,addr=0x13",
+        "-device",
+        "hda-duplex,bus=hdacombine.0,audiodev=combine",
+    ]
+    return args
+
+
 def resize_image(img_size_new: str, img_path: Path) -> None:
     """
     Truncates an image to a specific size. The value must be larger than the
```

3.299 /opt/GSM/osmo-operator/patches/qosmo-calypso-air-realiste.patch

18944 octets, 308 lignes → 308 lignes

```
diff --git a/hw/arm/calypso/l1-dsp/c54x_mem.c b/hw/arm/calypso/l1-dsp/c54x_mem.c
index cac9a37e..14b2ef5c 100644
--- a/hw/arm/calypso/l1-dsp/c54x_mem.c
+++ b/hw/arm/calypso/l1-dsp/c54x_mem.c
@@ -1140,6 +1140,7 @@ void data_write(C54xState *s, uint16_t addr, uint16_t val)
         if (rssi_on < 0) rssi_on = calypso_gate("CALYPSO_PM_RSSI", 1);
         if (rssi_on && ((addr >= 0x0834 && addr <= 0x0836) || (addr >= 0x0848 && addr <= 0x084A)))
             val = calypso_bsp_rssi_apm_suivante(NULL);   /* [2026-10-22] fenetre suivante, dans l'ordre
 ↪  (calypso_bsp.c) */
+        if (addr == 0x0831 || addr == 0x0845) val = calypso_bsp_air_dpm(val);   /* [2026-10-23] D_PM, air
 ↪  realiste */
         /* [2026-10-19] moniteur mailbox aussi sur le chemin rapide (voir data_read) */
         calypso_mbx(MBX_DSP_WR, addr, val, s->data[addr], s->pc, 0, s->insn_count);
         data_write_locked(s, addr, val);
@@ -1344,6 +1345,7 @@ void data_write(C54xState *s, uint16_t addr, uint16_t val)
                             (addr >= 0x0848 && addr <= 0x084A))) {
                     val = calypso_bsp_rssi_apm_suivante(NULL);   /* [2026-10-22] fenetre suivante, dans l'ordre
 ↪  (calypso_bsp.c) */
                 }
+                if (addr == 0x0831 || addr == 0x0845) val = calypso_bsp_air_dpm(val);   /* [2026-10-23] D_PM, air
 ↪  realiste */
             }
         }
         if (sent && addr == 0x08f8) {
diff --git a/hw/arm/calypso/l1-dsp/calypso_bsp.c b/hw/arm/calypso/l1-dsp/calypso_bsp.c
index 10d1b0fc..c8a4eb62 100644
--- a/hw/arm/calypso/l1-dsp/calypso_bsp.c
+++ b/hw/arm/calypso/l1-dsp/calypso_bsp.c
@@ -225,7 +225,7 @@ static int pm_limite_qbit(int rang)
 {
     return rang == 0 ? -1 : 2500;    /* -1 : seulement les trames precedentes */
 }
-static void bsp_pm_mav_noter(const int16_t *iq, int n)
+static void bsp_pm_mav_noter(const int16_t *iq, int n, double k_air)
 {
```

```
     if (g_pm_mav_tick != g_tick_courant) {
         if (g_pm_mav_n > 0) { memcpy(g_pm_mav_der, g_pm_mav, sizeof g_pm_mav); g_pm_mav_der_n = g_pm_mav_n; }
@@ -235,7 +235,7 @@ static void bsp_pm_mav_noter(const int16_t *iq, int n)
     int lim = pm_limite_qbit(g_pm_mav_n);      /* rang de cette fenetre dans la trame (avant ajout) */
     uint64_t acc = 0;
     for (int i = 0; i < n; i++) { int v = iq[i]; acc += (v < 0) ? (uint32_t)(-v) : (uint32_t)v; }
-    uint32_t mav = (uint32_t)(acc / (uint32_t)n);
+    uint32_t mav = (uint32_t)(acc / (uint32_t)n * k_air);
     uint16_t m16 = mav > 0xffff ? 0xffff : (uint16_t)mav;
     if (m16 >= 5000) g_pm_mav_cellule = m16;   /* dernier vrai burst vu dans une fenetre de mesure */
     g_pm_der_mav = m16; g_pm_der_gain = faux_air_gain(g_tick_courant, lim); g_pm_der_ok = 1;
@@ -255,6 +255,34 @@ static void bsp_pm_mav_noter(const int16_t *iq, int n)
     if (g_pm_fifo_w - g_pm_fifo_r > BSP_PM_FIFO_N) g_pm_fifo_r = g_pm_fifo_w - BSP_PM_FIFO_N;
 }
 static uint16_t bsp_apm_de_mav(double mav, int gain_db);
+static int air_niveaux_on(void);
+/* [2026-10-23] AGC de l'air realiste (CALYPSO_AIR_AGC=1, defaut) : le niveau d'un burst de la cellule (dBm) ne
+ * change plus son amplitude I/Q -- un vrai recepteur la ramene a pleine echelle par son AGC, et a l'amplitude
+ * 3000 le DSP ne decodait plus le BCCH (appli rssi : synchro OK mais MCC/MNC/LAC jamais lus, run du 23/10
+ * 18:55 ; sans l'air realiste : MCC 001 01 LAC 1 cell 6001). Le niveau ne porte que sur la MESURE de puissance :
+ * air_choisir() range le facteur d'amplitude par (fn, TN), calypso_bsp_rx_burst() l'applique au MAV de la
+ * fenetre de mesure (bsp_pm_mav_noter). */
+#define AIR_PM_RING 4096u
+static struct { uint32_t fn; float k; } g_air_pm[AIR_PM_RING][8];
+static double air_pm_k(uint32_t fn, unsigned tn)
+{
+    const typeof(g_air_pm[0][0]) *e = &g_air_pm[fn % AIR_PM_RING][tn & 7];
+    return (e->fn == fn && e->k > 0) ? e->k : 1.0;
+}
+static double g_air_k_der = 1.0;   /* facteur du dernier burst livre au DSP (calypso_bsp_rx_burst) */
+static int g_air_dpm = -1;          /* D_PM du modele pour ce burst (meme echelle que a_pm), -1 inconnu */
+static int faux_air_gain(uint32_t tick, int limite);
+/* D_PM (a_serv_demod[1], 1/64 dB a la bande de base, layer1/prim_rx_nb.c : agc_inp_dbm8_by_pm(D_PM >> 3)) : le
+ ↪  DSP
+ * le calcule sur des I/Q a pleine echelle ; l'appli rssi en tire le niveau de la vue synchro (rx_level). */
+uint16_t calypso_bsp_air_dpm(uint16_t val)
+{
+    /* La valeur du DSP ne sert pas : sur des I/Q a pleine echelle il sature (rxlev 63, -47 dBm affiche) et un
+     * simple decalage de 20 log k donnait -73 pour -60. Le modele de a_pm (bsp_apm_de_mav, gain programme de la
+     * fenetre) donne la meme echelle que la vue spectre. */
+    if (!air_niveaux_on() || g_air_dpm < 0 || val == 0) return val;
+    return (uint16_t)g_air_dpm;
+}
+static double air_env_dbm(const char *nom, double defaut);
 /* sonde [apm] (pont.c) : les MAV par fenetre de la trame en cours et de la precedente */
 int calypso_bsp_pm_mav_texte(char *buf, size_t n)
 {
@@ -317,6 +345,15 @@ static uint16_t bsp_apm_de_mav(double mav, int gain_db)
         const char *rr = calypso_getenv("CALYPSO_DECAN_PM_RF_REF");
         if (rr && *rr) {
             rf_ref = atof(rr);
+        } else if (air_niveaux_on()) {
+            /* [2026-10-23] AIR REALISTE : une seule echelle. mav_ref (20929) est le MAV d'un burst module a
+             * 30000, que l'air realiste pose a CALYPSO_AIR_REF_DBM (-40) ; avec -60 ici, la cellule jouee a
+             * -60 dBm (amplitude 3000) ressortait a -83 dBm dans le modele, et l'appli rssi l'affichait
+             * n'importe ou des que le gain de la fenetre manquait (-38 dBm releve le 23/10 18:50). */
+            rf_ref = air_env_dbm("CALYPSO_AIR_REF_DBM", -40.0);
+            /* MAV mesure d'un burst GMSK module a 30000 (moyenne des |I|,|Q| : 1330..1400 a l'amplitude 3000,
+             * trace [apmL] du 23/10) ; 20929 laissait la cellule 4 dB sous sa consigne (-64 pour -60). */
+            if (!(mr && *mr)) mav_ref = 13700.0;
         }
         if (mav_ref < 1.0) {
             mav_ref = 1.0;
@@ -965,12 +1002,18 @@ static int faux_air_gain(uint32_t tick, int limite)
     if (m->magic != CALYPSO_RF_ARFCN_MAGIC) return -999;
     uint32_t w = __atomic_load_n(&m->widx, __ATOMIC_ACQUIRE);
```

```
     uint32_t n = w < CALYPSO_RF_ARFCN_N ? w : CALYPSO_RF_ARFCN_N - 4;
+    int recent = -999;
     for (uint32_t k = 1; k <= n; k++) {
         const CalypsoRfProg *e = &m->e[(w - k) % CALYPSO_RF_ARFCN_N];
+        /* [2026-10-23] air realiste : le dernier reglage de RECEPTION (pas celui d'emission) sert de repli. Hors
+         * pas a pas (banc-rssi, tick 0 ou ~130 trames derriere QEMU) aucun reglage n'est « avant » le tick :
+         * -999 faisait convertir la PM avec le gain de RESET de c54x_exe (calypso_trf6151_apm_for_rf), l'appli
+         * rssi lisait -38 dBm pour une cellule a -60. Meme repli que air_accord. */
+        if (recent == -999 && !((uint16_t)e->reserve & (1u << 14))) recent = e->gain_db;
         int32_t d = (int32_t)(e->fn - tick);
         if (d < 0 || (d == 0 && e->qbit <= limite)) return e->gain_db;
     }
-    return -999;
+    return air_niveaux_on() ? recent : -999;
 }
 static void faux_air_bruit(int16_t *iq, int n_ech)
 {
@@ -1024,22 +1067,31 @@ static int air_accord(uint32_t tick, int limite, int *arfcn, int *ul, int *qbit)
     if (m->magic != CALYPSO_RF_ARFCN_MAGIC) return 0;
     uint32_t w = __atomic_load_n(&m->widx, __ATOMIC_ACQUIRE);
     uint32_t n = w < CALYPSO_RF_ARFCN_N ? w : CALYPSO_RF_ARFCN_N - 4;
+    /* [2026-10-23] REPLI SUR LE REGLAGE LE PLUS RECENT. Hors pas a pas (banc-rssi), le DSP court ~130 trames
+     * derriere QEMU (dsp.log fn=8895, shm fn=9023) et l'anneau ne garde que ~10 trames : aucun reglage n'etait
+     * « avant » le tick, l'accord restait inconnu et tout passait comme avant. Le synthe d'une appli comme rssi
+     * reste longtemps sur la meme frequence : le dernier reglage de reception est le bon. */
+    int r_arfcn = -1, r_ul = 0, r_qbit = -1;
     for (uint32_t k = 1; k <= n; k++) {
         const CalypsoRfProg *e = &m->e[(w - k) % CALYPSO_RF_ARFCN_N];
-        int32_t d = (int32_t)(e->fn - tick);
-        if (!(d < 0 || (d == 0 && e->qbit <= limite))) continue;
         uint16_t f = (uint16_t)e->reserve;
         if (f & (1u << 14)) continue;                           /* reglage d'emission */
-        if ((f & (1u << 15)) && (f & (1u << 13))) { *arfcn = f & 0x7ff; *ul = 1; *qbit = e->qbit; return 1; }
-        if (e->arfcn < 0) continue;
-        *arfcn = e->arfcn; *ul = 0; *qbit = e->qbit; return 1;
+        int a_ul = (f & (1u << 15)) && (f & (1u << 13));
+        int a = a_ul ? (int)(f & 0x7ff) : e->arfcn;
+        if (a < 0) continue;
+        if (r_arfcn < 0) { r_arfcn = a; r_ul = a_ul; r_qbit = e->qbit; }
+        int32_t d = (int32_t)(e->fn - tick);
+        if (!(d < 0 || (d == 0 && e->qbit <= limite))) continue;
+        *arfcn = a; *ul = a_ul; *qbit = e->qbit; return 1;
     }
+    if (r_arfcn >= 0) { *arfcn = r_arfcn; *ul = r_ul; *qbit = r_qbit; return 1; }
     return 0;
 }
 /* le montant copie par le pont, par trame et par TN (BTS) */
 #define AIR_UL_RING (1u << 12)
 static struct { uint32_t fn; uint8_t bits[8][148]; uint8_t presents; } *g_air_ul;
 static unsigned long g_air_ul_recus;
+static struct { uint32_t fn; int valide; } g_air_ul_der[8];   /* dernier fn montant recu par TN */
 static void air_ul_stocker(uint32_t fn, unsigned tn, const uint8_t *bits)
 {
     if (!g_air_ul && !(g_air_ul = calloc(AIR_UL_RING, sizeof *g_air_ul))) return;
@@ -1047,13 +1099,42 @@ static void air_ul_stocker(uint32_t fn, unsigned tn, const uint8_t *bits)
     if (g_air_ul[i].fn != fn) { g_air_ul[i].fn = fn; g_air_ul[i].presents = 0; }
     memcpy(g_air_ul[i].bits[tn & 7], bits, 148);
     g_air_ul[i].presents |= (uint8_t)(1u << (tn & 7));
+    g_air_ul_der[tn & 7].fn = fn; g_air_ul_der[tn & 7].valide = 1;
     g_air_ul_recus++;
 }
+/* [2026-10-23] le DERNIER burst montant de l'intervalle, s'il a moins de deux multitrames (102 trames) d'ecart
 ↪ avec
+ * la trame jouee. Chercher la trame exacte ne trouvait rien : sur le chemin direct la fenetre est servie a
+ * l'arrivee du descendant, que la BTS emet en avance, alors que le montant du meme fn n'existe pas encore
+ * (4050 recus, 0 joues, run du 23/10 14:50). */
```

```
 static const uint8_t *air_ul_bits(uint32_t fn, unsigned tn)
 {
-    if (!g_air_ul) return NULL;
-    unsigned i = fn % AIR_UL_RING;
-    return (g_air_ul[i].fn == fn && (g_air_ul[i].presents & (1u << (tn & 7)))) ? g_air_ul[i].bits[tn & 7] : NULL;
+    if (!g_air_ul || !g_air_ul_der[tn & 7].valide) return NULL;
+    uint32_t f = g_air_ul_der[tn & 7].fn;
+    int32_t d = (int32_t)(fn - f);
+    if (d < -102 || d > 102) return NULL;
+    unsigned i = f % AIR_UL_RING;
+    return (g_air_ul[i].fn == f && (g_air_ul[i].presents & (1u << (tn & 7)))) ? g_air_ul[i].bits[tn & 7] : NULL;
+}
+/* bruit de l'air realiste : plancher CALYPSO_AIR_BRUIT_DBM (-110), sur la meme echelle que les bursts (30000 =
+ * CALYPSO_AIR_REF_DBM). Le bruit du faux air (ecart-type 200, ~-83 dBm sur cette echelle) noyait le montant du
+ * mobile (-75 dBm = 533) : les huit barres de la vue UL restaient au meme niveau (run du 23/10 14:51). */
+static void air_bruit(int16_t *iq, int n_ech)
+{
+    static double sigma = -1; static uint32_t x = 0x9e3779b9u;
+    if (sigma < 0) sigma = 30000.0 * pow(10.0, (air_env_dbm("CALYPSO_AIR_BRUIT_DBM", -110.0)
+                                                - air_env_dbm("CALYPSO_AIR_REF_DBM", -40.0)) / 20.0);
+    for (int k = 0; k < 2 * n_ech; k++) {
+        double u = 0;
+        for (int j = 0; j < 4; j++) { x = x * 1664525u + 1013904223u; u += (double)(x >> 8) / 16777216.0; }
+        iq[k] = (int16_t)lrint((u - 2.0) * 1.732 * sigma);
+    }
+}
+/* ramene des echantillons modules a 30000 (sb_moduler) a l'amplitude choisie par l'air realiste */
+static void air_echelle(int16_t *iq, int n_ech, int amp)
+{
+    if (amp == 30000) return;
+    double k = amp / 30000.0;
+    for (int i = 0; i < 2 * n_ech; i++) iq[i] = (int16_t)lrint(iq[i] * k);
 }
 /* intervalle d'une fenetre de mesure, tire du qbit du reglage du synthe (pose ~200 qbits avant la fenetre) */
 static unsigned air_pos_fenetre(uint32_t tick)
@@ -1075,18 +1156,25 @@ static const uint8_t *air_choisir(uint32_t tick, int limite, uint32_t fn, unsign
     double dbm = 0;
     if (cell < 0 || a == cell) {
         if (!ul) { b = dl; dbm = air_env_dbm("CALYPSO_AIR_DL_DBM", -60.0) - calypso_bsp_get_last_att(); }
-        else     { b = air_ul_bits(pos >= 3 ? fn : fn - 1, (pos + 5) & 7); dbm = air_env_dbm("CALYPSO_AIR_UL_DBM",
 ↪  -75.0); }
+        else     { b = air_ul_bits(fn, (pos + 5) & 7); dbm = air_env_dbm("CALYPSO_AIR_UL_DBM", -75.0); }
     }
+    static int agc = -1;
+    if (agc < 0) { const char *e = calypso_getenv("CALYPSO_AIR_AGC"); agc = !(e && *e == '0'); }
+    typeof(g_air_pm[0][0]) *pm = &g_air_pm[fn % AIR_PM_RING][pos & 7];
+    pm->fn = fn; pm->k = 1.0f;
     if (b) {
         double r = *amp * pow(10.0, (dbm - air_env_dbm("CALYPSO_AIR_REF_DBM", -40.0)) / 20.0);
-        *amp = r > 32000.0 ? 32000 : r < 1.0 ? 1 : (int)r;
+        if (r > 32000.0) r = 32000.0;
+        if (r < 1.0) r = 1.0;
+        if (agc) pm->k = (float)(r / *amp);    /* I/Q a pleine echelle, le niveau va a la mesure */
+        else *amp = (int)r;
         if (ul) n_ul++; else n_dl++;
     } else n_bruit++;
     unsigned long t = n_dl + n_ul + n_bruit;
     if (t <= 30 || t % 5000 == 0)
-        printf("  [air] tick=%u fn=%u pos=%u accord %s %d (cellule %d) -> %s amp=%d [%lu DL, %lu UL, %lu bruit ;
 ↪  %lu UL recus]\n",
+        printf("  [air] tick=%u fn=%u pos=%u accord %s %d (cellule %d) -> %s amp=%d k_pm=%.3f [%lu DL, %lu UL,
 ↪  %lu bruit ; %lu UL recus]\n",
                tick, fn, pos, ul ? "UL" : "DL", a, cell, b ? (ul ? "burst montant" : "burst descendant") : "bruit",
-               b ? *amp : 0, n_dl, n_ul, n_bruit, g_air_ul_recus);
+               b ? *amp : 0, (double)pm->k, n_dl, n_ul, n_bruit, g_air_ul_recus);
     return b;
```

```
 }
 /* [2026-10-21 19:20] BRUIT DE FOND (CALYPSO_BSP_BRUIT_FOND=1, Magnetite). Les intervalles sans burst etaient
@@ -2360,8 +2448,11 @@ static void bsp_ts0_livrer(uint32_t tick_fn, unsigned i)
         if (one_shot && nwin > 0 && nwin < 150) pm_servir(nwin);
         devie = true;
     }
-    if (hors_freq) faux_air_bruit(iq + 2 * marge, 148);
-    else if (sb) sb_moduler(bits, iq + 2 * marge, g_ts0[i].fn);   /* [2026-09-30] voir sb_moduler() */
+    if (hors_freq) { if (air_niveaux_on()) air_bruit(iq + 2 * marge, 148); else faux_air_bruit(iq + 2 * marge,
 ↪  148); }
+    else if (sb) {
+        sb_moduler(bits, iq + 2 * marge, g_ts0[i].fn);   /* [2026-09-30] voir sb_moduler() */
+        if (air_niveaux_on()) air_echelle(iq + 2 * marge, 148, amp_ts0);   /* [2026-10-23] meme niveau que la
 ↪  cellule */
+    }
     else    gmsk_moduler(bits_mod, 148, amp_ts0, 0.0, 0.5, iq + 2 * marge);
     if (!sb && !fcch && !hors_freq) {
         static double sym = -2;
@@ -2446,7 +2537,7 @@ static void bsp_ts0_livrer(uint32_t tick_fn, unsigned i)
                 b = air_choisir(tick_fn, 5000, g_ts0[i].fn, (unsigned)tn, b, &amp_autre);
                 if (!b) {
                     static int16_t bruit_autre[2 * 148];
-                    faux_air_bruit(bruit_autre, 148);
+                    air_bruit(bruit_autre, 148);
                     calypso_bsp_rx_burst((uint8_t)tn, g_ts0[i].fn, bruit_autre, 2 * 148);
                     continue;
                 }
@@ -3275,9 +3366,22 @@ static void bsp_trxd_readable(void *opaque)
             est_sb = 1;
             for (int k = 0; k < 64 && est_sb; k++) if ((bits[42 + k] & 1) != train_sb[k]) est_sb = 0;
         }
-        if (est_sb) sb_moduler(bits, iq + iq_count, fn);
+        /* [2026-10-23] air realiste aussi sur ce chemin direct (sans CALYPSO_BSP_STREAM : banc-rssi) */
+        int amp_dir = 30000, bruit_dir = 0;
+        if (air_niveaux_on() && nbits == 148) {
+            const uint8_t *b2 = air_choisir(g_tick_courant, 5000, fn, tn, bits, &amp_dir);
+            if (!b2) bruit_dir = 1;
+            else if (b2 != bits) { bits = b2; est_sb = 0; }
+        }
+        if (bruit_dir) air_bruit(iq + iq_count, nbits);
+        else if (est_sb) {
+           /* [2026-10-23] le SCH au meme niveau que le reste de la cellule : a 30000 au milieu de bursts a 3000,
+             * il saturait apres le reglage de gain sur le FB (FB1 -71 dBm puis « Sync error ») */
+            sb_moduler(bits, iq + iq_count, fn);
+            air_echelle(iq + iq_count, nbits, amp_dir);
+        }
         else {
-            gmsk_moduler(bits, nbits, 30000, 0.0, 0.5, iq + iq_count);
+            gmsk_moduler(bits, nbits, amp_dir, 0.0, 0.5, iq + iq_count);
             if (nbits == 148) {
                 static double sym = -2;
                if (sym == -2) { const char *e = calypso_getenv("CALYPSO_BSP_NB_SYM"); sym = (e && *e) ? atof(e) :
 ↪  0.3; }
@@ -4023,13 +4127,19 @@ void calypso_bsp_rx_burst(uint8_t tn, uint32_t fn,
         return;
     }
     if (n_int16 <= 0 || iq == NULL) return;
+    g_air_k_der = air_niveaux_on() ? air_pm_k(fn, tn) : 1.0;
+    if (air_niveaux_on()) {
+        uint64_t acc = 0;
+        for (int i = 0; i < n_int16; i++) { int v = iq[i]; acc += (v < 0) ? (uint32_t)(-v) : (uint32_t)v; }
+        g_air_dpm = bsp_apm_de_mav((double)acc / n_int16 * g_air_k_der, faux_air_gain(g_tick_courant, 5000));
+    }
     if (calypso_bsp_dma_trace_on())
         printf("  [dma-dep] tick=%u insn=%u fn=%u tn=%u n=%d armee=%d one_shot=%d fenetre=%d\n", g_tick_courant,
                bsp.dsp->insn_count, (unsigned)fn, (unsigned)tn, n_int16 / 2, (int)calypso_rhea_dma_rx_armed(),
```

```
               (int)calypso_rhea_dma_one_shot(), calypso_rhea_dma_get_len_words() / 2);
     if (calypso_rhea_dma_one_shot() && calypso_rhea_dma_get_len_words() / 2 > 0 &&
         calypso_rhea_dma_get_len_words() / 2 < 150)
-        bsp_pm_mav_noter(iq, n_int16);    /* fenetre de mesure : voir calypso_bsp_rssi_apm_k() */
+        bsp_pm_mav_noter(iq, n_int16, g_air_k_der);   /* fenetre de mesure : voir calypso_bsp_rssi_apm_k() */

     /* [2026-09-20] FULL TIMESLOT. A bare 148-symbol burst (296 words) is padded
      * with the 8.25 guard symbols of its timeslot (156 symbols, 157 on
diff --git a/hw/arm/calypso/l1-dsp/calypso_bsp.h b/hw/arm/calypso/l1-dsp/calypso_bsp.h
index 2c0d5db2..dee8dac8 100644
--- a/hw/arm/calypso/l1-dsp/calypso_bsp.h
+++ b/hw/arm/calypso/l1-dsp/calypso_bsp.h
@@ -128,6 +128,8 @@ uint16_t calypso_bsp_rssi_apm_k(int k);
 int calypso_bsp_pm_mav_texte(char *buf, size_t n);
 int calypso_bsp_pm_mav_fort(void);
 uint16_t calypso_bsp_rssi_apm_suivante(int *depile_mav);
+/* [2026-10-23] air realiste + AGC : le D_PM ecrit par le DSP (a_serv_demod[1]) ramene au niveau du dernier burst
 ↪  */
+uint16_t calypso_bsp_air_dpm(uint16_t val);
 int calypso_apm_trace_on(void);

 int calypso_bsp_fenetre_tn(void);
```

3.300 /opt/GSM/osmo-operator/patches/qosmo-calypso-bsp-magnetite-sb-fin-trame.patch

30508 octets, 499 lignes → 499 lignes

```
diff --git a/hw/arm/calypso/l1-dsp/calypso_bsp.c b/hw/arm/calypso/l1-dsp/calypso_bsp.c
index 881cd317..6d72ad53 100644
--- a/hw/arm/calypso/l1-dsp/calypso_bsp.c
+++ b/hw/arm/calypso/l1-dsp/calypso_bsp.c
@@ -603,6 +603,101 @@ uint8_t  calypso_bsp_get_last_att(void)   { return bsp.last_att; }
 static struct { uint32_t fn; uint8_t bits[148]; uint8_t valid, joue; } *g_ts0;
 static uint32_t g_ts0_next; static int g_ts0_any;
 static int64_t g_ts0_offset = INT64_MIN;
+/* [2026-10-21] firmware TI en mode dedie (taches ALLC vues par montant.c) : voir CALYPSO_BSP_DEDIE_DECAL */
+static int g_dedie_ti;
+extern unsigned long calypso_bsp_routeur_bilan(char *buf, size_t n);
+void calypso_bsp_set_dedie_ti(int actif)
+{
+    char b[160] = "";
+    if (actif != g_dedie_ti) {
+        if (!actif) calypso_bsp_routeur_bilan(b, sizeof b);
+        printf("  [ts0] firmware TI %s le mode dedie%s\n", actif ? "entre dans" : "quitte", b);
+    }
+    g_dedie_ti = actif;
+}
+/* [2026-10-21] FENETRE SB EN FIN DE TRAME (CALYPSO_BSP_SB_FIN_TRAME=1, Magnetite / L1 TI TCS211).
+ * La L1 TI ouvre sa fenetre SB SB_MARGIN (92 qbits) avant le debut du burst : quand le SCH est cale
+ * en debut de trame, elle s'ouvre A LA FIN DE LA TRAME PRECEDENTE (SB_I time_alignmt=4928), et la L1
+ * en deduit « trame BTS N = trame locale k+1 » (l1a_add_time_for_nb : 4928+80 -> 8, fn_offset-1).
+ * Le banc, lui, livrait au tick k le burst du tick k : toutes les fenetres NB etaient armees un
+ * tick APRES leur burst (BCCH : p51=2 dans aucune fenetre, p51=3..5 dans celles de 2..4), CRC KO
+ * sur chaque bloc. Dans ce mode l'offset est cale sur fn - tick - 1 au SCH livre en fenetre SB,
+ * puis VERIFIE sur la lecture BCCH elle-meme (bsp_bcch_calage) : la L1 arme ses fenetres NB sur les
+ * trames ou elle croit le bloc, ce qui dit directement de combien le flux est decale. Une lecture
+ * en avance du SCH (premier essai) aggravait les choses : a la resynchro la L1 arme DEUX fenetres SB
+ * de suite et cale son horloge sur la seconde. */
+static int sb_fin_trame(void)
+{
+    static int on = -1;
+    if (on < 0) { const char *e = calypso_getenv("CALYPSO_BSP_SB_FIN_TRAME"); on = (e && *e == '1'); }
+    return on;
```

```
+}
+/* [2026-10-21] FAUX AIR (CALYPSO_FAUX_AIR=<arfcn de la cellule>, Magnetite). Le pont livrait les bursts
+ * du BTS quelle que soit la frequence du telephone : la L1 TI voyait la cellule (BSIC 7) sur 65, 81,
+ * 265... et RR s'y perdait. QEMU publie chaque programmation du synthetiseur (calypso_rf_arfcn.h) ;
+ * ici, une fenetre accordee ailleurs que sur la cellule recoit du bruit (CALYPSO_FAUX_AIR_BRUIT,
+ * ecart-type, 200 par defaut) au lieu du burst. ARFCN inconnu (pas de shm, pas encore programme) :
+ * comportement d'avant. */
+#include "hw/arm/calypso/calypso_rf_arfcn.h"
+#include <sys/mman.h>
+#include <fcntl.h>
+static int faux_air_cellule(void)
+{
+    static int a = -2;
+    if (a == -2) { const char *e = calypso_getenv("CALYPSO_FAUX_AIR"); a = (e && *e) ? atoi(e) : -1;
+                   if (a >= 0) printf("  [faux-air] cellule sur l'ARFCN %d : bruit sur les autres frequences\n",
 ↪  a); }
+    return a;
+}
+/* ARFCN de la fenetre du tick : derniere programmation faite avant `limite` qbits dans la trame tick. */
+static int faux_air_arfcn(uint32_t tick, int limite)
+{
+    static const CalypsoRfArfcnShm *m;
+    static uint32_t essai;
+    if (!m) {
+        if (essai++ % 216) return -1;              /* QEMU cree le fichier a sa premiere ecriture */
+        int fd = open(CALYPSO_RF_ARFCN_SHM, O_RDONLY);
+        if (fd < 0) return -1;
+        void *p = mmap(NULL, sizeof *m, PROT_READ, MAP_SHARED, fd, 0);
+        close(fd);
+        if (p == MAP_FAILED) return -1;
+        m = p;
+    }
+    if (m->magic != CALYPSO_RF_ARFCN_MAGIC) return -1;
+    uint32_t w = __atomic_load_n(&m->widx, __ATOMIC_ACQUIRE);
+    uint32_t n = w < CALYPSO_RF_ARFCN_N ? w : CALYPSO_RF_ARFCN_N - 4;   /* les plus vieilles peuvent etre
 ↪  reecrites */
+    for (uint32_t k = 1; k <= n; k++) {
+        const CalypsoRfProg *e = &m->e[(w - k) % CALYPSO_RF_ARFCN_N];
+        int32_t d = (int32_t)(e->fn - tick);
+        if (d < 0 || (d == 0 && e->qbit <= limite)) return e->arfcn;
+    }
+    return -1;
+}
+static void faux_air_bruit(int16_t *iq, int n_ech)
+{
+    static double sigma = -1; static uint32_t x = 0x12345678u;
+    if (sigma < 0) { const char *e = calypso_getenv("CALYPSO_FAUX_AIR_BRUIT"); sigma = (e && *e) ? atof(e) :
 ↪  200.0; }
+    for (int k = 0; k < 2 * n_ech; k++) {
+        double u = 0;
+        for (int j = 0; j < 4; j++) { x = x * 1664525u + 1013904223u; u += (double)(x >> 8) / 16777216.0; }
+        iq[k] = (int16_t)lrint((u - 2.0) * 1.732 * sigma);   /* somme de 4 uniformes ~ gaussienne */
+    }
+}
+/* [2026-10-21 19:20] BRUIT DE FOND (CALYPSO_BSP_BRUIT_FOND=1, Magnetite). Les intervalles sans burst etaient
+ * bourres de ZEROS : la L1 TI y prend ses mesures de puissance, le DSP rendait PM = 0, et au bout de 100
+ * (l1_check_pm_error) la L1 se croyait plantee : l1_initialize_for_recovery, DSP relance toutes les ~173
+ * trames (« > RECOVERY »), retards et campement casse. Un vrai recepteur voit toujours du bruit. */
+extern uint16_t g_remplissage_ts0[];
+static const int16_t *bsp_remplissage(void)
+{
+    static int on = -1;
+    static int16_t bruit[2 * 148];
+    if (on < 0) { const char *e = calypso_getenv("CALYPSO_BSP_BRUIT_FOND"); on = (e && *e == '1'); }
+    if (!on) return (const int16_t *)g_remplissage_ts0;
+    faux_air_bruit(bruit, 148);
```

```
+    return bruit;
+}
+static int g_bcch_verrou;        /* offset confirme par la lecture BCCH : les SCH ne le recalent plus */
+static uint32_t g_bcch_vu_tick;  /* dernier tick ou une fenetre NB BCCH a ete vue armee */
 /* [2026-10-20] MODE TRX (CALYPSO_BSP_TRX=1, c54x_exe/run.sh OSMO_TRX=1) : cette instance est la radio de la
  * BTS (firmware trx), elle ne recoit jamais de SB ; ses bursts montants arrivent de l'air (pont/air.py)
  * tagues du fn GSM du mobile. L'offset fn <-> tick est alors POSE par pont.c, qui decode le sch du SB
@@ -1272,15 +1367,116 @@ int calypso_bsp_sb_retenter(void)
     g_dernier_sch.livre_sb = 1;
     /* Le SCH est livre a ce tick-ci : c'est lui qui fixe l'offset ARM-tick,
      * exactement comme la livraison native en fenetre SB (bsp_ts0_livrer). */
-    int64_t off = (int64_t)g_dernier_sch.fn - (int64_t)g_tick_courant;
+    int64_t off = (int64_t)g_dernier_sch.fn - (int64_t)g_tick_courant - sb_fin_trame();
     int recale = (off != g_ts0_offset);
-    if (!g_trx_cale) g_ts0_offset = off;
+    if (!g_trx_cale && !g_bcch_verrou) g_ts0_offset = off;
     if (g_sb_relivres <= 100 || g_sb_relivres % 100 == 0)
         printf("  [sbwin] SB fn=%u RELIVRE au tick %u (depose au tick %u, fenetre SB armee apres) : page=%d
 ↪  marge=%d%s [%lu relivres, %lu expires]\n",
                g_dernier_sch.fn, g_tick_courant, g_dernier_sch.tick, total, marge,
                recale ? ", offset ARM-tick recale" : "", g_sb_relivres, g_sb_expires);
     return 1;
 }
+/* [2026-10-21 22:45] ROUTAGE DES FENETRES EN DEDIE (firmware TI). Le banc depose UN burst (148 echantillons,
+ * TS0) par tick ; en dedie la ROM arme DEUX fenetres one-shot par trame : la mesure de puissance (64) puis la
+ * reception NB (151), 215 echantillons (CALYPSO_BSP_TS0_DEBUG, run de 22:19 : SDCCH p51 22..25 deposes en
+ * « fenetre=64 », les 151 armees 3 a 5 ticks plus tard et de plus en plus tard : +67 echantillons par trame).
+ * Sur silicium la mesure tombe sur un autre intervalle et consomme du vide. Ici : fenetre de mesure (< 150)
+ * <- remplissage (bsp_remplissage), fenetre NB (150..189) <- le vrai burst, garde dans un anneau si la NB
+ * n'est pas armee au depot ; pont.c fait tourner le DSP entre deux livraisons dans le meme tick (boucle
+ * pump). Seulement en dedie TI (calypso_bsp_set_dedie_ti) ; CALYPSO_BSP_NB_RETENTER=0 pour couper. */
+#define NB_GARDE_N 4
+static struct { int valide; uint32_t fn, tick; uint8_t bits[148]; } g_nb_garde[NB_GARDE_N];
+static unsigned g_nb_garde_w;
+static unsigned long g_nb_relivres, g_nb_expires, g_pm_servies;
+static int nb_retenter_on(void)
+{
+    static int on = -1;
+    if (on < 0) { const char *e = calypso_getenv("CALYPSO_BSP_NB_RETENTER"); on = !(e && *e == '0'); }
+    return on && g_dedie_ti;
+}
+static void nb_garde_poser(uint32_t fn, uint32_t tick, const uint8_t *bits)
+{
+    unsigned k = g_nb_garde_w++ % NB_GARDE_N;
+    if (g_nb_garde[k].valide) g_nb_expires++;
+    g_nb_garde[k].fn = fn; g_nb_garde[k].tick = tick; g_nb_garde[k].valide = 1;
+    memcpy(g_nb_garde[k].bits, bits, 148);
+}
+static void nb_moduler(const uint8_t *bits, int16_t *iq, int marge)
+{
+    static double sym = -2;
+    if (sym == -2) { const char *e = calypso_getenv("CALYPSO_BSP_NB_SYM"); sym = (e && *e) ? atof(e) : 0.3; }
+    gmsk_moduler(bits, 148, 30000, 0.0, 0.5, iq + 2 * marge);
+    gmsk_elargir(iq + 2 * marge, 148, sym);
+}
+static int nb_marge_ti(void)
+{
+    static int nb_marge = -1;
+    if (nb_marge < 0) { const char *e = calypso_getenv("CALYPSO_BSP_NB_MARGE"); nb_marge = (e && *e) ? atoi(e) :
 ↪  3;
+                        if (nb_marge < 0 || nb_marge > 3) nb_marge = 3; }
+    return nb_marge;
+}
+/* la fenetre de mesure armee recoit du remplissage : elle se termine, la ROM arme la suivante */
+static int pm_servir(int nwin)
+{
```

```
+    calypso_rif_flush();
+    calypso_bsp_rx_burst(0, g_depot_fn, bsp_remplissage(), 2 * nwin);
+    g_pm_servies++;
+    return 1;
+}
+/* [2026-10-21 22:55] SONDE DMA (CALYPSO_BSP_DMA_TRACE=1) : chaque changement d'etat du canal RX (armee,
+ * one-shot, longueur) avec le tick, pendant le dedie TI. Le modele rhea-dma trace ses registres sur stderr,
+ * mais c54x_exe (verbosite.c) les avale. Sert a voir ce que la ROM arme sur le TCH (TS1), ou aucune fenetre
+ * n'etait armee au depot (appel du 21/10 22:42 : a_dd BFI, « >N200 of SABM » sur la FACCH). 3000 lignes. */
+static void dma_trace(const char *ou)
+{
+    static int on = -1; static unsigned n; static int armee_p = -1, os_p = -1, len_p = -1;
+    if (on < 0) { const char *e = calypso_getenv("CALYPSO_BSP_DMA_TRACE"); on = (e && *e == '1'); }
+    if (!on || !g_dedie_ti || n >= 3000) return;
+    int armee = calypso_rhea_dma_rx_armed(), os = calypso_rhea_dma_one_shot(), len =
 ↪  calypso_rhea_dma_get_len_words() / 2;
+    if (armee == armee_p && os == os_p && len == len_p) return;
+    armee_p = armee; os_p = os; len_p = len; n++;
+    printf("  [dma] tick=%u %s : armee=%d one_shot=%d fenetre=%d rif=%d\n", g_tick_courant, ou, armee, os, len,
 ↪  calypso_rif_level());
+}
+void calypso_bsp_dma_trace(const char *ou) { dma_trace(ou); }
+unsigned long calypso_bsp_routeur_bilan(char *buf, size_t n)
+{
+    snprintf(buf, n, " (routeur : %lu NB relivres, %lu expires, %lu mesures servies)", g_nb_relivres,
 ↪  g_nb_expires, g_pm_servies);
+    return g_nb_relivres;
+}
+int calypso_bsp_nb_retenter(void)
+{
+    dma_trace("pump");
+    if (!nb_retenter_on()) return 0;
+    if (!calypso_rhea_dma_rx_armed() || !calypso_rhea_dma_one_shot()) return 0;
+    int nwin = calypso_rhea_dma_get_len_words() / 2;
+    if (nwin <= 0) return 0;
+    if (nwin < 150) return pm_servir(nwin);
+    if (nwin >= 190) return 0;                                  /* fenetre SB : calypso_bsp_sb_retenter */
+    /* la NB est armee : le plus ancien burst garde encore frais (2 ticks) */
+    int k_min = -1;
+    for (int k = 0; k < NB_GARDE_N; k++) {
+        if (!g_nb_garde[k].valide) continue;
+        if ((int32_t)(g_tick_courant - g_nb_garde[k].tick) > 2) { g_nb_garde[k].valide = 0; g_nb_expires++;
 ↪  continue; }
+        if (k_min < 0 || (int32_t)(g_nb_garde[k].fn - g_nb_garde[k_min].fn) < 0) k_min = k;
+    }
+    if (k_min < 0) return 0;
+    int marge = nb_marge_ti();
+    static int16_t iq[2 * 256];
+    int total = nwin > marge + 148 ? nwin : marge + 148;
+    if (total > 256) total = 256;
+    memset(iq, 0, sizeof iq);
+    nb_moduler(g_nb_garde[k_min].bits, iq, marge);
+    calypso_rif_flush();
+    calypso_bsp_rx_burst(0, g_nb_garde[k_min].fn, iq, 2 * total);
+    bsp.bursts_written++;
+    g_nb_garde[k_min].valide = 0;
+    g_nb_relivres++;
+    if (g_nb_relivres <= 40 || g_nb_relivres % 500 == 0)
+        printf("  [nbwin] NB fn=%u p51=%u RELIVRE au tick %u (depose au tick %u) : fenetre=%d [%lu relivres, %lu
 ↪  expires, %lu mesures servies]\n",
+               g_nb_garde[k_min].fn, g_nb_garde[k_min].fn % 51u, g_tick_courant, g_nb_garde[k_min].tick, nwin,
+               g_nb_relivres, g_nb_expires, g_pm_servies);
+    return 1;
+}
 /* pont.c : tracer CHAQUE SCH pendant `ticks` ticks (apres une detection FB, ou
  * quand l'ARM poste une tache SB), pour voir le depot exact autour des tentatives. */
```

```
 void calypso_bsp_sb_trace(unsigned ticks)
@@ -1288,6 +1484,44 @@ void calypso_bsp_sb_trace(unsigned ticks)
     uint32_t j = g_tick_courant + ticks;
     if ((int32_t)(j - g_sb_trace_jusqua) > 0) g_sb_trace_jusqua = j;
 }
+/* [2026-10-21] SB_FIN_TRAME : la lecture BCCH (p51 2..5) dit ou la L1 croit le bloc. Fenetres NB
+ * armees sur 3..6 = flux en avance d'une trame (offset - 1) ; sur 1..4 = en retard (offset + 1).
+ * Deux blocs de suite avant de corriger ; un bloc bien cale verrouille l'offset (les SCH relus ne le
+ * recalent plus), deverrouille quand plus aucune fenetre BCCH n'est vue (la L1 a perdu la cellule). */
+static void bsp_bcch_calage(uint32_t tick, unsigned p, bool nb)
+{
+    static uint8_t arme[7];
+    static int tard, tot;
+    if (p >= 1 && p <= 6) {
+        arme[p] = nb;
+        if (nb) g_bcch_vu_tick = tick;
+    }
+    if (p == 6) {
+        bool bon = arme[2] && arme[3] && arme[4] && arme[5] && !arme[6];
+        bool retard = !arme[2] && arme[3] && arme[4] && arme[5] && arme[6];
+        bool avance = arme[1] && arme[2] && arme[3] && arme[4] && !arme[5];
+        if (retard) { tard++; tot = 0; }
+        else if (avance) { tot++; tard = 0; }
+        else if (bon) {
+            tard = tot = 0;
+            if (!g_bcch_verrou) printf("  [bcch] tick=%u bloc BCCH dans ses fenetres (2..5) : offset %lld
 ↪  verrouille\n",
+                                       tick, (long long)g_ts0_offset);
+            g_bcch_verrou = 1;
+        }
+        if (tard >= 2 || tot >= 2) {
+            int d = tard >= 2 ? -1 : +1;
+            g_ts0_offset = (g_ts0_offset + d + BSP_FN_MAX) % BSP_FN_MAX;
+            printf("  [bcch] tick=%u fenetres NB sur p51 %s : offset %+d -> %lld\n",
+                   tick, d < 0 ? "3..6 (flux en avance)" : "1..4 (flux en retard)", d, (long long)g_ts0_offset);
+            tard = tot = 0;
+        }
+        memset(arme, 0, sizeof arme);
+    }
+    if (g_bcch_verrou && (uint32_t)(tick - g_bcch_vu_tick) > 160u) {
+        g_bcch_verrou = 0;
+        printf("  [bcch] tick=%u plus de lecture BCCH depuis %u trames : offset deverrouille\n", tick, tick -
 ↪  g_bcch_vu_tick);
+    }
+}
 static void bsp_ts0_livrer(uint32_t tick_fn, unsigned i)
 {
     static int16_t iq[2 * 512];
@@ -1434,7 +1668,26 @@ static void bsp_ts0_livrer(uint32_t tick_fn, unsigned i)
     const int page_prog = calypso_rhea_dma_get_len_words() / 2;   /* ALGTH/2 mots = echantillons par page,
 ↪  ONE_SHOT ou pas */
     const int rif_avant = calypso_rif_level();
     int nwin = one_shot ? page_prog : 0;
-    int marge = nwin >= 190 ? 21 : nwin >= 150 ? 3 : 0;
+    /* [2026-10-21] CALYPSO_BSP_NB_MARGE : la L1 TI vise un TOA NB de 3 (l1ctl_toa_update : ISH =
+     * TOA_estime - 16 qbits) ; a la marge 3 le DSP mesure 5, d'ou +8 qbits de tpu_offset a chaque
+     * periode sans fin (le banc ne deplace pas le burst). Magnetite : 1. */
+    static int nb_marge = -1;
+    if (nb_marge < 0) { const char *e = calypso_getenv("CALYPSO_BSP_NB_MARGE"); nb_marge = (e && *e) ? atoi(e) :
 ↪  3;
+                        if (nb_marge < 0 || nb_marge > 3) nb_marge = 3; }
+    int marge = nwin >= 190 ? 21 : nwin >= 150 ? nb_marge : 0;
+    /* [2026-10-21] faux air : fenetre NB (debut de trame) -> synthetiseur programme avant 1250 qbits ;
+     * fenetre SB (fin de trame), PM ou continue -> derniere programmation de la trame. */
+    bool hors_freq = false;
+    if (faux_air_cellule() >= 0) {
```

```
+        int a = faux_air_arfcn(tick_fn, (nwin >= 150 && nwin < 190) ? 1250 : 5000);
+        hors_freq = (a >= 0 && a != faux_air_cellule());
+        static unsigned n_hors, n_dans;
+        if (hors_freq) n_hors++; else if (a >= 0) n_dans++;
+        if (hors_freq && (n_hors <= 20 || n_hors % 5000 == 0))
+            printf("  [faux-air] tick=%u fn=%u fenetre=%d accorde sur %d : bruit [%u hors, %u sur la cellule]\n",
+                   tick_fn, g_ts0[i].fn, nwin, a, n_hors, n_dans);
+        if (hors_freq) sb = fcch = 0;
+    }
     /* [2026-09-29] Un SCH est memorise ici (g_dernier_sch) ; si la fenetre SB
      * ne s'arme qu'apres ce depot, calypso_bsp_sb_retenter() l'y livrera. */
     const char *sb_action = "cadree comme un burst normal";
@@ -1452,9 +1705,19 @@ static void bsp_ts0_livrer(uint32_t tick_fn, unsigned i)
     if (g_trx_cale && !sb && !fcch && !bsp_ts0_est_rach(bits)) {
        g_trx_ul.fn = g_ts0[i].fn; g_trx_ul.tick = tick_fn; memcpy(g_trx_ul.bits, bits, 148); g_trx_ul.valide = 1;
     }
-    if (sb) sb_moduler(bits, iq + 2 * marge, g_ts0[i].fn);   /* [2026-09-30] voir sb_moduler() */
+    /* [2026-10-21 22:45] dedie TI (routage, voir calypso_bsp_nb_retenter) : la NB n'est pas armee au depot ->
+     * le burst va dans l'anneau de garde, et une fenetre de mesure armee recoit du remplissage a sa place */
+    dma_trace("depot");
+    bool devie = false;
+    if (!sb && !fcch && !hors_freq && nb_retenter_on() && !(one_shot && nwin >= 150 && nwin < 190)) {
+        nb_garde_poser(g_ts0[i].fn, tick_fn, bits);
+        if (one_shot && nwin > 0 && nwin < 150) pm_servir(nwin);
+        devie = true;
+    }
+    if (hors_freq) faux_air_bruit(iq + 2 * marge, 148);
+    else if (sb) sb_moduler(bits, iq + 2 * marge, g_ts0[i].fn);   /* [2026-09-30] voir sb_moduler() */
     else    gmsk_moduler(bits, 148, 30000, 0.0, 0.5, iq + 2 * marge);
-    if (!sb && !fcch) {
+    if (!sb && !fcch && !hors_freq) {
         static double sym = -2;
         if (sym == -2) { const char *e = calypso_getenv("CALYPSO_BSP_NB_SYM"); sym = (e && *e) ? atof(e) : 0.3; }
         gmsk_elargir(iq + 2 * marge, 148, sym);
@@ -1481,10 +1744,10 @@ static void bsp_ts0_livrer(uint32_t tick_fn, unsigned i)
         }
     }
     if (sb && one_shot && nwin >= 190) {
-        int64_t off = (int64_t)g_ts0[i].fn - (int64_t)tick_fn;
+        int64_t off = (int64_t)g_ts0[i].fn - (int64_t)tick_fn - sb_fin_trame();
         if (off != g_ts0_offset) printf("  [ts0] SB fn=%u livree au tick %u dans une fenetre SB : offset ARM-tick
 ↪  = %lld%s\n",
                                         g_ts0[i].fn, tick_fn, (long long)off, g_ts0_offset == INT64_MIN ? "
↪  (premier calage)" : " (recalage)");
-        if (!g_trx_cale) g_ts0_offset = off;
+        if (!g_trx_cale && !g_bcch_verrou) g_ts0_offset = off;
     }
     /* [2026-09-21] Une ligne par trame livree : c'etait la sonde de calage des
      * fenetres (nwin > 0 est vrai pour TOUT burst normal depuis la DMA one-shot,
@@ -1502,11 +1765,20 @@ static void bsp_ts0_livrer(uint32_t tick_fn, unsigned i)
                           : (nl < 20 || nl % 5000 == 0);
       nl++;
       if (trame_dediee && nd < 300) { trace = true; nd++; }
+      /* [2026-10-21] Magnetite : SB decodee (BSIC 7) mais chaque bloc BCCH en CRC KO (BS_I
+       * error_flag=1) ; les 400 premieres trames BCCH normal (p51 2..5) disent quelle fenetre
+       * DMA la ROM avait armee quand le burst est arrive. */
+      if (sb_fin_trame() && g_dedie_tn <= 0)
+          bsp_bcch_calage(tick_fn, g_ts0[i].fn % 51u, nwin >= 150 && nwin < 190);
+      { static unsigned nbcch;
+        const unsigned p = g_ts0[i].fn % 51u;
+        if (p >= 2 && p <= 5 && nbcch < 400) { trace = true; nbcch++; } }
       if (trace) printf("  [ts0] tick=%u fn=%u p51=%u %s fenetre=%d marge=%d rif_avant=%d%s\n",
                         tick_fn, g_ts0[i].fn, g_ts0[i].fn % 51u,
                         sb ? "SB" : fcch ? "FCCH" : "NB", nwin, marge, calypso_rif_level(),
                         trame_dediee ? (burst_dedie ? "  <- canal dedie" : "  <- canal dedie, BURST MANQUANT") :
↪  ""); }
```

```
     g_ts0[i].joue = 1;
+    if (devie) return;                      /* garde / remplissage faits ci-dessus */
     calypso_bsp_rx_burst(0, g_ts0[i].fn, iq, 2 * total);
     bsp.bursts_written++;
     /* Continuous DMA (FB search): the ROM counts 1250 samples per frame and
@@ -1521,8 +1793,7 @@ static void bsp_ts0_livrer(uint32_t tick_fn, unsigned i)
         for (int tn = 1; tn < 8; tn++) {
             const uint8_t *b = bsp_autres_bits(g_ts0[i].fn, (unsigned)tn);
             if (!b) {
-                calypso_bsp_rx_burst((uint8_t)tn, g_ts0[i].fn,
-                                     (const int16_t *)g_remplissage_ts0, 2 * 148);
+                calypso_bsp_rx_burst((uint8_t)tn, g_ts0[i].fn, bsp_remplissage(), 2 * 148);
                 continue;
             }
             memset(iq_autre, 0, sizeof iq_autre);
@@ -1568,6 +1839,18 @@ static void bsp_ts0_livrer(uint32_t tick_fn, unsigned i)
  * (0 avant le premier calage sur SB : l'horloge reste libre), trame BTS
  * reclamee, tick de l'ARM. CALYPSO_BSP_HORLOGE=0 coupe la publication. */
 #define SHM_HORLOGE "/dev/shm/calypso_horloge"
+/* [2026-10-21 18:45] trame BTS d'un tick, d'apres le calage du flux TS0 (SB, puis verrou BCCH) : la
+ * trame que la L1 du telephone croit vivre a ce tick. pont.c (tx_tsp_fn) en date les bursts montants
+ * quand le firmware ne pose pas a_a5fn (L1 TI : le RACH de Magnetite partait « TICK NU », l'air ne le
+ * placait pas, B->A 0). 0 si pas encore cale. */
+int calypso_bsp_fn_bts_du_tick(uint32_t tick, uint32_t *fn_bts)
+{
+    if (g_ts0_offset == INT64_MIN) return 0;
+    int64_t w = ((int64_t)tick + g_ts0_offset) % (int64_t)BSP_FN_MAX;
+    if (w < 0) w += BSP_FN_MAX;
+    *fn_bts = (uint32_t)w;
+    return 1;
+}
 static void bsp_horloge_publier(uint32_t fn_bts, uint32_t tick_fn, int cale)
 {
     static int actif = -1, fd = -1;
@@ -1665,9 +1948,9 @@ static void bsp_ts0_service(uint32_t tick_fn)
     if (g_sc.actif) {
         if (tick_fn == g_sc.tick + 1) {
             if (sc_rom_a_decode()) {
-                int64_t off = (int64_t)g_sc.fn_sch - (int64_t)g_sc.tick;
+                int64_t off = (int64_t)g_sc.fn_sch - (int64_t)g_sc.tick - sb_fin_trame();
                 if (off < 0) off += BSP_FN_MAX;
-                g_sc.ok++; if (!g_trx_cale) g_ts0_offset = off;
+                g_sc.ok++; if (!g_trx_cale && !g_bcch_verrou) g_ts0_offset = off;
                 if (g_sc.ok <= 300 || g_sc.ok % 50 == 0)
                    printf("  [sbwin] SECONDE CHANCE : SB fn=%u DECODE a la tentative 1 (tick %u), offset ARM-tick
 ↪  recale [%u/%u]\n",
                           g_sc.fn_sch, g_sc.tick, g_sc.ok, g_sc.n);
@@ -1687,8 +1970,63 @@ static void bsp_ts0_service(uint32_t tick_fn)
     static unsigned manques, manques_log;
     if (g_ts0_offset != INT64_MIN) {
         int64_t w = ((int64_t)tick_fn + g_ts0_offset) % (int64_t)BSP_FN_MAX; if (w < 0) w += BSP_FN_MAX;
+        /* [2026-10-21 19:10] RATTRAPAGE DU DIRECT (CALYPSO_BSP_RATTRAPE=1, Magnetite). Au boot l'instance
+         * prend du retard sur le BTS (relances DSP du firmware, verrou ARM qui attend jusqu'a 5 s par trame)
+         * et ne le rattrape jamais : mesure 18:55, le TRX attendait le montant fn 36209 et recevait 33528,
+         * 2700 trames (12 s) de retard, les RACH datees du passe n'etaient jamais jouees. Au-dela de 200
+         * trames de retard sur la derniere trame recue, l'offset saute au direct (marge 2, voir plus bas) et le
 ↪  verrou
+         * BCCH tombe : la L1 voit une discontinuite et se resynchronise sur le flux en direct. */
+        {
+            /* [2026-10-21 20:10] marge 2 (au lieu de 20) et seuil 30 (au lieu de 200) : le montant n'a que ~2
+             * trames de jeu chez le TRX (BTS_RX_LAG) ; avec 20 a 200 trames de retard les RACH arrivaient
+             * apres leur trame (mesure : fn 14556 recu quand le TRX etait a 14583), jamais joues. */
+            static int rattrape = -1, marge_r = 2, seuil_r = 30;
+            if (rattrape < 0) { const char *e = calypso_getenv("CALYPSO_BSP_RATTRAPE"); rattrape = (e && *e ==
 ↪  '1');
+                                e = calypso_getenv("CALYPSO_BSP_RATTRAPE_MARGE"); if (e && *e) marge_r = atoi(e);
```

```
+                               e = calypso_getenv("CALYPSO_BSP_RATTRAPE_SEUIL"); if (e && *e) seuil_r = atoi(e);
+                                if (seuil_r <= marge_r) seuil_r = marge_r + 1; }
+           int32_t retard = (int32_t)(g_ts0_fn_max - (uint32_t)w);
+           if (rattrape && g_ts0_any && !g_trx_cale && retard > seuil_r) {
+               int64_t saut = retard - marge_r;
+               g_ts0_offset = (g_ts0_offset + saut) % (int64_t)BSP_FN_MAX;
+               g_bcch_verrou = 0;
+               w = ((int64_t)tick_fn + g_ts0_offset) % (int64_t)BSP_FN_MAX; if (w < 0) w += BSP_FN_MAX;
+               printf("  [ts0] RATTRAPAGE : %d trames de retard sur le BTS (derniere recue fn=%u) -> offset
 ↪ %+lld, livre fn=%lld\n",
+                      retard, g_ts0_fn_max, (long long)saut, (long long)w);
+           }
+       }
+       /* [2026-10-21 22:05] DECALAGE EN DEDIE (CALYPSO_BSP_DEDIE_DECAL=<n>, Magnetite). L'offset est verrouille
+        * sur la lecture BCCH (491 au run de 21:59 alors que les SB donnaient 489-490) ; en dedie la L1 TI ne
+        * decode ni SACCH (SA_I en erreur) ni SDCCH (l'UA jamais lu, « >N200 of SABM »). Applique seulement
+        * pendant les taches dediees signalees par montant.c (calypso_bsp_set_dedie_ti). */
+       {
+           static int decal = -999;
+           if (decal == -999) { const char *e = calypso_getenv("CALYPSO_BSP_DEDIE_DECAL"); decal = (e && *e) ?
 ↪ atoi(e) : 0;
+                                if (decal) printf("  [ts0] decalage en dedie TI : %+d trame(s)\n", decal); }
+           if (decal && g_dedie_ti) { w = (w + decal) % (int64_t)BSP_FN_MAX; if (w < 0) w += BSP_FN_MAX; }
+       }
        unsigned i = (unsigned)w % BSP_TS0_RING;
        bsp_horloge_publier((uint32_t)w, tick_fn, 1);
+       /* [2026-10-21 18:55] SB_FIN_TRAME, offset VERROUILLE par la lecture BCCH : la L1 TI ouvre sa
+        * fenetre SB a la fin de la trame qui precede le SB ; flux cale sur les NB, le SCH arrivait au tick
+        * suivant, fenetre fermee (relectures SB du campement toutes ratees : SB_I flag 0, NET_I 0 ->
+        * la cellule etait abandonnee). Une fenetre SB armee pendant la trame w recoit donc le SCH w+1,
+        * sans recalage (verrou). Hors verrou (resynchro), la lecture en avance faussait le calage. */
+       if (sb_fin_trame() && g_bcch_verrou && g_dedie_tn <= 0 && calypso_rhea_dma_rx_armed() &&
+           calypso_rhea_dma_one_shot() && calypso_rhea_dma_get_len_words() / 2 >= 190) {
+           uint32_t fn1 = (uint32_t)((w + 1) % BSP_FN_MAX);
+           unsigned j = fn1 % BSP_TS0_RING;
+           if (g_ts0[j].valid && g_ts0[j].fn == fn1 && bsp_ts0_est_sb(g_ts0[j].bits)) {
+               static unsigned n_avance;
+               if (++n_avance <= 20 || n_avance % 500 == 0)
+                 printf("  [sbwin] SB_FIN_TRAME (verrou) : fenetre SB au tick %u (trame %u) -> SCH fn=%u [%u]\n",
+                        tick_fn, (uint32_t)w, fn1, n_avance);
+               bsp_ts0_livrer(tick_fn, j);
+               g_dernier_sch.valide = 0;
+               g_ts0[i].joue = 1;
+               return;
+           }
+       }
        if (g_ts0[i].valid && g_ts0[i].fn == (uint32_t)w && sb_double_on() && g_dedie_tn <= 0 &&
            bsp_ts0_est_fcch(g_ts0[i].bits) && calypso_rhea_dma_rx_armed() && calypso_rhea_dma_one_shot() &&
            calypso_rhea_dma_get_len_words() / 2 >= 190) {
@@ -1855,16 +2193,16 @@ static void bsp_ts0_service(uint32_t tick_fn)
            int total = marge > 0 ? (nwin > marge + 148 ? nwin : marge + 148) : 148;
            if (total > 256) total = 256;
            memset(vide, 0, (size_t)(2 * total) * sizeof *vide);
+           if (bsp_remplissage() != (const int16_t *)g_remplissage_ts0)
+               faux_air_bruit(vide, total);              /* bruit de fond, pas des zeros (PM = 0) */
            calypso_bsp_rx_burst(0, (uint32_t)w, vide, 2 * total);
            bsp.bursts_written++;
            /* En mode continu la trame doit faire 1250 echantillons, sinon
             * le compteur de symboles de la ROM derive : memes remplissages
             * que bsp_ts0_livrer(). */
            if (!one_shot) {
-               extern uint16_t g_remplissage_ts0[];
                for (int tn = 1; tn < 8; tn++) {
-                   calypso_bsp_rx_burst((uint8_t)tn, (uint32_t)w,
-                                        (const int16_t *)g_remplissage_ts0, 2 * 148);
+                   calypso_bsp_rx_burst((uint8_t)tn, (uint32_t)w, bsp_remplissage(), 2 * 148);
```

```
                    }
                }
                g_ts0_silences++;
diff --git a/hw/arm/calypso/l1-dsp/calypso_bsp.h b/hw/arm/calypso/l1-dsp/calypso_bsp.h
index b4e684a3..faad7b02 100644
--- a/hw/arm/calypso/l1-dsp/calypso_bsp.h
+++ b/hw/arm/calypso/l1-dsp/calypso_bsp.h
@@ -41,6 +41,7 @@ int calypso_bsp_trx_calage(uint32_t fn_sb, uint32_t tick);
 int calypso_bsp_trx_ul_prothese(uint16_t *dspdata, uint32_t tick, int w_page);   /* [2026-10-20] mode TRX : bits
 ↪  exacts du burst montant dans la page de resultat (pont.c, phase A) */   /* [2026-10-20] mode TRX : offset
 ↪  fn/tick depuis le SB emis (pont.c) */
 int  calypso_bsp_service(uint32_t current_fn);
 int  calypso_bsp_sb_retenter(void);
+int  calypso_bsp_nb_retenter(void);   /* [2026-10-21] NB garde en dedie TI */
 void calypso_bsp_sb_trace(unsigned ticks);   /* [2026-09-29] tracer chaque SCH pendant n ticks ([sbwin]) */
 ↪  /* [2026-09-29] relivre un SCH garde faute de DMA armee ; 1 si livre (pont.c, boucle de pompe) */  /* drains
 ↪  UDP 6702 and delivers the current frame (standalone host) */

 /*
@@ -80,6 +81,7 @@ bool calypso_bsp_send_rach_ra(uint8_t ra, uint8_t bsic, uint32_t fn, uint8_t tn)

 uint16_t calypso_bsp_get_daram_addr(void);
 uint32_t calypso_bsp_get_last_fn(void);
+int calypso_bsp_fn_bts_du_tick(uint32_t tick, uint32_t *fn_bts);
 uint16_t calypso_bsp_get_daram_len(void);

 /* Reference probe: compares DARAM against the last bursts the BSP was handed;
```

3.301 /opt/GSM/osmo-operator/patches/qosmo-calypso-bsp-pas-de-routeur-nb-sur-tch.patch

9215 octets, 159 lignes → 159 lignes

```
diff --git a/hw/arm/calypso/l1-dsp/calypso_bsp.c b/hw/arm/calypso/l1-dsp/calypso_bsp.c
index 881cd317..a4b4646c 100644
--- a/hw/arm/calypso/l1-dsp/calypso_bsp.c
+++ b/hw/arm/calypso/l1-dsp/calypso_bsp.c
@@ -1272,15 +1381,152 @@ int calypso_bsp_sb_retenter(void)
     g_dernier_sch.livre_sb = 1;
     /* Le SCH est livre a ce tick-ci : c'est lui qui fixe l'offset ARM-tick,
      * exactement comme la livraison native en fenetre SB (bsp_ts0_livrer). */
-    int64_t off = (int64_t)g_dernier_sch.fn - (int64_t)g_tick_courant;
+    int64_t off = (int64_t)g_dernier_sch.fn - (int64_t)g_tick_courant - sb_fin_trame();
+    g_off_mobile = (int64_t)g_dernier_sch.fn - (int64_t)g_tick_courant;   /* ce que le mobile vient d'apprendre */
     int recale = (off != g_ts0_offset);
-    if (!g_trx_cale) g_ts0_offset = off;
+    if (!g_trx_cale && !g_bcch_verrou) g_ts0_offset = off;
     if (g_sb_relivres <= 100 || g_sb_relivres % 100 == 0)
         printf("  [sbwin] SB fn=%u RELIVRE au tick %u (depose au tick %u, fenetre SB armee apres) : page=%d
 ↪  marge=%d%s [%lu relivres, %lu expires]\n",
                g_dernier_sch.fn, g_tick_courant, g_dernier_sch.tick, total, marge,
                recale ? ", offset ARM-tick recale" : "", g_sb_relivres, g_sb_expires);
     return 1;
 }
+/* [2026-10-21 22:45] ROUTAGE DES FENETRES EN DEDIE (firmware TI). Le banc depose UN burst (148 echantillons,
+ * TS0) par tick ; en dedie la ROM arme DEUX fenetres one-shot par trame : la mesure de puissance (64) puis la
+ * reception NB (151), 215 echantillons (CALYPSO_BSP_TS0_DEBUG, run de 22:19 : SDCCH p51 22..25 deposes en
+ * « fenetre=64 », les 151 armees 3 a 5 ticks plus tard et de plus en plus tard : +67 echantillons par trame).
+ * Sur silicium la mesure tombe sur un autre intervalle et consomme du vide. Ici : fenetre de mesure (< 150)
+ * <- remplissage (bsp_remplissage), fenetre NB (150..189) <- le vrai burst, garde dans un anneau si la NB
+ * n'est pas armee au depot ; pont.c fait tourner le DSP entre deux livraisons dans le meme tick (boucle
+ * pump). Seulement en dedie TI (calypso_bsp_set_dedie_ti) ; CALYPSO_BSP_NB_RETENTER=0 pour couper. */
+#define NB_GARDE_N 4
+static struct { int valide; uint32_t fn, tick; uint8_t bits[148]; } g_nb_garde[NB_GARDE_N];
```

```
+static unsigned g_nb_garde_w;
+static unsigned long g_nb_relivres, g_nb_expires, g_pm_servies;
+static int nb_retenter_on(void)
+{
+    /* [2026-10-21 23:00] EN VEILLE AUSSI (plus seulement en dedie TI) : la L1 TI mesure des voisins sur les
+     * trames de paging et de BCCH ; archive 22:40 (TS0_DEBUG) : au depot, fenetre de mesure (64) sur 661 des
+     * 1972 trames de paging (p51 16..19) et 516 des 2364 BCCH -> bloc de paging en erreur (NP_I error_flag=1,
+     * IL 59), compteur DSC epuise, reselection de cellule toutes les ~8 s (BS_S, BAND_R, RXL_R, CEL_R : les
+     * barres d'antenne apparaissent 5 s puis disparaissent), et aucun paging recu (pas de SMS/appel entrant).
+     * CALYPSO_BSP_NB_RETENTER=0 pour couper, =2 : dedie seul. */
+    static int on = -1;
+    if (on < 0) { const char *e = calypso_getenv("CALYPSO_BSP_NB_RETENTER"); on = (e && *e) ? atoi(e) : 1; }
+    return on == 1 || (on == 2 && g_dedie_ti);
+}
+static int pm_sur_cellule(void);   /* la mesure vise-t-elle la frequence de la cellule ? (faux air) */
+static void nb_garde_poser(uint32_t fn, uint32_t tick, const uint8_t *bits)
+{
+    unsigned k = g_nb_garde_w++ % NB_GARDE_N;
+    if (g_nb_garde[k].valide) g_nb_expires++;
+    g_nb_garde[k].fn = fn; g_nb_garde[k].tick = tick; g_nb_garde[k].valide = 1;
+    memcpy(g_nb_garde[k].bits, bits, 148);
+}
+static void nb_moduler(const uint8_t *bits, int16_t *iq, int marge)
+{
+    static double sym = -2;
+    if (sym == -2) { const char *e = calypso_getenv("CALYPSO_BSP_NB_SYM"); sym = (e && *e) ? atof(e) : 0.3; }
+    gmsk_moduler(bits, 148, 30000, 0.0, 0.5, iq + 2 * marge);
+    gmsk_elargir(iq + 2 * marge, 148, sym);
+}
+static int nb_marge_ti(void)
+{
+    static int nb_marge = -1;
+    if (nb_marge < 0) { const char *e = calypso_getenv("CALYPSO_BSP_NB_MARGE"); nb_marge = (e && *e) ? atoi(e) :
 ↪  3;
+                        if (nb_marge < 0 || nb_marge > 3) nb_marge = 3; }
+    return nb_marge;
+}
+/* la fenetre de mesure armee recoit de quoi se terminer (la ROM arme alors la suivante) : la tete du dernier
+ * burst garde si la mesure vise la cellule (sa puissance est la bonne), du remplissage sinon */
+static int pm_servir(int nwin)
+{
+    static int16_t iq[2 * 256];
+    const int16_t *src = bsp_remplissage();
+    int k_rec = -1;
+    for (int k = 0; k < NB_GARDE_N; k++)
+        if (g_nb_garde[k].valide && (k_rec < 0 || (int32_t)(g_nb_garde[k].fn - g_nb_garde[k_rec].fn) > 0)) k_rec
 ↪  = k;
+    if (k_rec >= 0 && pm_sur_cellule()) {
+        memset(iq, 0, sizeof iq);
+        nb_moduler(g_nb_garde[k_rec].bits, iq, 0);
+        src = iq;
+    }
+    calypso_rif_flush();
+    calypso_bsp_rx_burst(0, g_depot_fn, src, 2 * (nwin > 148 ? 148 : nwin));
+    g_pm_servies++;
+    return 1;
+}
+/* [2026-10-21 22:55] SONDE DMA (CALYPSO_BSP_DMA_TRACE=1) : chaque changement d'etat du canal RX (armee,
+ * one-shot, longueur) avec le tick, pendant le dedie TI. Le modele rhea-dma trace ses registres sur stderr,
+ * mais c54x_exe (verbosite.c) les avale. Sert a voir ce que la ROM arme sur le TCH (TS1), ou aucune fenetre
+ * n'etait armee au depot (appel du 21/10 22:42 : a_dd BFI, « >N200 of SABM » sur la FACCH). 3000 lignes. */
+static int pm_sur_cellule(void)
+{
+    int cell = faux_air_cellule();
+    if (cell < 0) return 1;
+    int a = faux_air_arfcn(g_tick_courant, 5000);
```

```
+    return a < 0 || a == cell;
+}
+static void dma_trace(const char *ou)
+{
+    static int on = -1; static unsigned n; static int armee_p = -1, os_p = -1, len_p = -1;
+    if (on < 0) { const char *e = calypso_getenv("CALYPSO_BSP_DMA_TRACE"); on = (e && *e == '1'); }
+    if (!on || !g_dedie_ti || n >= 3000) return;
+    int armee = calypso_rhea_dma_rx_armed(), os = calypso_rhea_dma_one_shot(), len =
 ↪  calypso_rhea_dma_get_len_words() / 2;
+    if (armee == armee_p && os == os_p && len == len_p) return;
+    armee_p = armee; os_p = os; len_p = len; n++;
+    printf("  [dma] tick=%u %s : armee=%d one_shot=%d fenetre=%d rif=%d\n", g_tick_courant, ou, armee, os, len,
 ↪  calypso_rif_level());
+}
+void calypso_bsp_dma_trace(const char *ou) { dma_trace(ou); }
+unsigned long calypso_bsp_routeur_bilan(char *buf, size_t n)
+{
+    snprintf(buf, n, " (routeur : %lu NB relivres, %lu expires, %lu mesures servies)", g_nb_relivres,
 ↪  g_nb_expires, g_pm_servies);
+    return g_nb_relivres;
+}
+int calypso_bsp_nb_retenter(void)
+{
+    dma_trace("pump");
+    if (!nb_retenter_on()) return 0;
+    /* [2026-10-21 23:40] PAS SUR LE TCH. pont.c y depose deja apres l'IDLE, fenetre armee (PONT_TCH_DEPOT_IDLE).
+     * Appel Magnetite de 23:34 : deux NB gardes relivres dans le meme tick 11084, IT DMA en attente (IFR=4000)
+     * avec le bit 14 retire de l'IMR (3000), puis la ROM part a pc=0 (SP-CORRUPT) ; la L1 TI reinitialise
+     * le DSP ~1400 trames plus tard et le mobile perd la cellule a chaque appel.
+     * CALYPSO_BSP_NB_RETENTER_TCH=1 : ancien comportement. */
+    {
+        static int sur_tch = -1;
+        if (sur_tch < 0) { const char *e = calypso_getenv("CALYPSO_BSP_NB_RETENTER_TCH"); sur_tch = (e && *e ==
 ↪  '1'); }
+        if (!sur_tch && g_dedie_tn > 0 && g_dedie_genre == BSP_DEDIE_TCH) return 0;
+    }
+    if (!calypso_rhea_dma_rx_armed() || !calypso_rhea_dma_one_shot()) return 0;
+    int nwin = calypso_rhea_dma_get_len_words() / 2;
+    if (nwin <= 0) return 0;
+    if (nwin < 150) return pm_servir(nwin);
+    if (nwin >= 190) return 0;                                  /* fenetre SB : calypso_bsp_sb_retenter */
+    /* la NB est armee : le plus ancien burst garde encore frais (2 ticks) */
+    int k_min = -1;
+    for (int k = 0; k < NB_GARDE_N; k++) {
+        if (!g_nb_garde[k].valide) continue;
+        if ((int32_t)(g_tick_courant - g_nb_garde[k].tick) > 2) { g_nb_garde[k].valide = 0; g_nb_expires++;
 ↪  continue; }
+        if (k_min < 0 || (int32_t)(g_nb_garde[k].fn - g_nb_garde[k_min].fn) < 0) k_min = k;
+    }
+    if (k_min < 0) return 0;
+    int marge = nb_marge_ti();
+    static int16_t iq[2 * 256];
+    int total = nwin > marge + 148 ? nwin : marge + 148;
+    if (total > 256) total = 256;
+    memset(iq, 0, sizeof iq);
+    nb_moduler(g_nb_garde[k_min].bits, iq, marge);
+    calypso_rif_flush();
+    calypso_bsp_rx_burst(0, g_nb_garde[k_min].fn, iq, 2 * total);
+    bsp.bursts_written++;
+    g_nb_garde[k_min].valide = 0;
+    g_nb_relivres++;
+    if (g_nb_relivres <= 40 || g_nb_relivres % 500 == 0)
+        printf("  [nbwin] NB fn=%u p51=%u RELIVRE au tick %u (depose au tick %u) : fenetre=%d [%lu relivres, %lu
 ↪  expires, %lu mesures servies]\n",
+               g_nb_garde[k_min].fn, g_nb_garde[k_min].fn % 51u, g_tick_courant, g_nb_garde[k_min].tick, nwin,
+               g_nb_relivres, g_nb_expires, g_pm_servies);
```

```
+    return 1;
+}
 /* pont.c : tracer CHAQUE SCH pendant `ticks` ticks (apres une detection FB, ou
  * quand l'ARM poste une tache SB), pour voir le depot exact autour des tentatives. */
 void calypso_bsp_sb_trace(unsigned ticks)
```

3.302 /opt/GSM/osmo-operator/patches/qosmo-calypso-bsp-pm-par-fenetre.patch

13734 octets, 245 lignes → 245 lignes

```
diff --git a/hw/arm/calypso/l1-dsp/calypso_bsp.c b/hw/arm/calypso/l1-dsp/calypso_bsp.c
index 881cd317..7c5b08ee 100644
--- a/hw/arm/calypso/l1-dsp/calypso_bsp.c
+++ b/hw/arm/calypso/l1-dsp/calypso_bsp.c
@@ -171,7 +171,45 @@ bool calypso_bsp_fb_stream_next(uint16_t *outI, uint16_t *outQ)
     return true;
 }

+/* [2026-10-21 23:50] UNE MESURE PAR FENETRE. En veille la L1 TI fait plusieurs mesures dans la meme trame
+ * (d_task_md = 0x200 + n : la cellule ARFCN 1 et le voisin 277 = DCS 614), et apm_jamais_nul (pont.c) les
+ * remplissait toutes avec le MAV du DERNIER paquet livre : la fenetre du voisin (bruit du faux air) passait
+ * en dernier et la cellule tombait au plancher avec lui -- IBA_I (1 54) -> (1 8), barres perdues des
+ * l'entree en veille paging (run 23:43). On garde donc le MAV de chaque fenetre de mesure (one-shot,
+ * < 150 echantillons) dans l'ordre de la trame ; la k-ieme PM prend le k-ieme. */
+#define BSP_PM_MAV_N 8
+static uint32_t g_tick_courant;        /* (definition anticipee, voir bsp_ts0_service) */
+static uint16_t g_pm_mav[BSP_PM_MAV_N], g_pm_mav_der[BSP_PM_MAV_N];
+static int g_pm_mav_n, g_pm_mav_der_n;
+static uint32_t g_pm_mav_tick;
+static void bsp_pm_mav_noter(const int16_t *iq, int n)
+{
+    if (g_pm_mav_tick != g_tick_courant) {
+        if (g_pm_mav_n > 0) { memcpy(g_pm_mav_der, g_pm_mav, sizeof g_pm_mav); g_pm_mav_der_n = g_pm_mav_n; }
+        g_pm_mav_tick = g_tick_courant; g_pm_mav_n = 0;
+    }
+    if (g_pm_mav_n >= BSP_PM_MAV_N || n <= 0) return;
+    uint64_t acc = 0;
+    for (int i = 0; i < n; i++) { int v = iq[i]; acc += (v < 0) ? (uint32_t)(-v) : (uint32_t)v; }
+    uint32_t mav = (uint32_t)(acc / (uint32_t)n);
+    g_pm_mav[g_pm_mav_n++] = mav > 0xffff ? 0xffff : (uint16_t)mav;
+}
+static uint16_t bsp_apm_de_mav(double mav);
+uint16_t calypso_bsp_rssi_apm_k(int k)
+{
+    if (g_pm_mav_tick == g_tick_courant && k < g_pm_mav_n) return bsp_apm_de_mav(g_pm_mav[k]);
+    if (g_pm_mav_tick != g_tick_courant || g_pm_mav_n == 0) {
+        if (k < g_pm_mav_der_n) return bsp_apm_de_mav(g_pm_mav_der[k]);
+    }
+    return calypso_bsp_rssi_apm();
+}
+
 uint16_t calypso_bsp_rssi_apm(void)
+{
+    return bsp_apm_de_mav((double)bsp_last_mav);
+}
+
+static uint16_t bsp_apm_de_mav(double mav)
 {
     static double mav_ref = 20929.0, rf_ref = -60.0;
     static int init;
@@ -1272,15 +1418,169 @@ int calypso_bsp_sb_retenter(void)
     g_dernier_sch.livre_sb = 1;
     /* Le SCH est livre a ce tick-ci : c'est lui qui fixe l'offset ARM-tick,
      * exactement comme la livraison native en fenetre SB (bsp_ts0_livrer). */
```

```
-    int64_t off = (int64_t)g_dernier_sch.fn - (int64_t)g_tick_courant;
+    int64_t off = (int64_t)g_dernier_sch.fn - (int64_t)g_tick_courant - sb_fin_trame();
+    g_off_mobile = (int64_t)g_dernier_sch.fn - (int64_t)g_tick_courant;   /* ce que le mobile vient d'apprendre */
     int recale = (off != g_ts0_offset);
-    if (!g_trx_cale) g_ts0_offset = off;
+    if (!g_trx_cale && !g_bcch_verrou) g_ts0_offset = off;
     if (g_sb_relivres <= 100 || g_sb_relivres % 100 == 0)
         printf("  [sbwin] SB fn=%u RELIVRE au tick %u (depose au tick %u, fenetre SB armee apres) : page=%d
 ↪  marge=%d%s [%lu relivres, %lu expires]\n",
               g_dernier_sch.fn, g_tick_courant, g_dernier_sch.tick, total, marge,
               recale ? ", offset ARM-tick recale" : "", g_sb_relivres, g_sb_expires);
     return 1;
 }
+/* [2026-10-21 22:45] ROUTAGE DES FENETRES EN DEDIE (firmware TI). Le banc depose UN burst (148 echantillons,
+ * TS0) par tick ; en dedie la ROM arme DEUX fenetres one-shot par trame : la mesure de puissance (64) puis la
+ * reception NB (151), 215 echantillons (CALYPSO_BSP_TS0_DEBUG, run de 22:19 : SDCCH p51 22..25 deposes en
+ * « fenetre=64 », les 151 armees 3 a 5 ticks plus tard et de plus en plus tard : +67 echantillons par trame).
+ * Sur silicium la mesure tombe sur un autre intervalle et consomme du vide. Ici : fenetre de mesure (< 150)
+ * <- remplissage (bsp_remplissage), fenetre NB (150..189) <- le vrai burst, garde dans un anneau si la NB
+ * n'est pas armee au depot ; pont.c fait tourner le DSP entre deux livraisons dans le meme tick (boucle
+ * pump). Seulement en dedie TI (calypso_bsp_set_dedie_ti) ; CALYPSO_BSP_NB_RETENTER=0 pour couper. */
+#define NB_GARDE_N 4
+static struct { int valide; uint32_t fn, tick; uint8_t bits[148]; } g_nb_garde[NB_GARDE_N];
+static unsigned g_nb_garde_w;
+static unsigned long g_nb_relivres, g_nb_expires, g_pm_servies;
+static int nb_retenter_on(void)
+{
+    /* [2026-10-21 23:00] EN VEILLE AUSSI (plus seulement en dedie TI) : la L1 TI mesure des voisins sur les
+     * trames de paging et de BCCH ; archive 22:40 (TS0_DEBUG) : au depot, fenetre de mesure (64) sur 661 des
+     * 1972 trames de paging (p51 16..19) et 516 des 2364 BCCH -> bloc de paging en erreur (NP_I error_flag=1,
+     * IL 59), compteur DSC epuise, reselection de cellule toutes les ~8 s (BS_S, BAND_R, RXL_R, CEL_R : les
+     * barres d'antenne apparaissent 5 s puis disparaissent), et aucun paging recu (pas de SMS/appel entrant).
+     * CALYPSO_BSP_NB_RETENTER=0 pour couper, =2 : dedie seul. */
+    static int on = -1;
+    if (on < 0) { const char *e = calypso_getenv("CALYPSO_BSP_NB_RETENTER"); on = (e && *e) ? atoi(e) : 1; }
+    return on == 1 || (on == 2 && g_dedie_ti);
+}
+static int pm_sur_cellule(void);   /* la mesure vise-t-elle la frequence de la cellule ? (faux air) */
+static void nb_garde_poser(uint32_t fn, uint32_t tick, const uint8_t *bits)
+{
+    unsigned k = g_nb_garde_w++ % NB_GARDE_N;
+    if (g_nb_garde[k].valide) g_nb_expires++;
+    g_nb_garde[k].fn = fn; g_nb_garde[k].tick = tick; g_nb_garde[k].valide = 1;
+    memcpy(g_nb_garde[k].bits, bits, 148);
+}
+static void nb_moduler(const uint8_t *bits, int16_t *iq, int marge)
+{
+    static double sym = -2;
+    if (sym == -2) { const char *e = calypso_getenv("CALYPSO_BSP_NB_SYM"); sym = (e && *e) ? atof(e) : 0.3; }
+    gmsk_moduler(bits, 148, 30000, 0.0, 0.5, iq + 2 * marge);
+    gmsk_elargir(iq + 2 * marge, 148, sym);
+}
+static int nb_marge_ti(void)
+{
+    static int nb_marge = -1;
+    if (nb_marge < 0) { const char *e = calypso_getenv("CALYPSO_BSP_NB_MARGE"); nb_marge = (e && *e) ? atoi(e) :
 ↪  3;
+                        if (nb_marge < 0 || nb_marge > 3) nb_marge = 3; }
+    return nb_marge;
+}
+/* la fenetre de mesure armee recoit de quoi se terminer (la ROM arme alors la suivante) : la tete du dernier
+ * burst garde si la mesure vise la cellule (sa puissance est la bonne), du remplissage sinon */
+static int pm_servir(int nwin)
+{
+    static int16_t iq[2 * 256];
+    const int16_t *src = bsp_remplissage();
```

```
+    int k_rec = -1;
+    for (int k = 0; k < NB_GARDE_N; k++)
+        if (g_nb_garde[k].valide && (k_rec < 0 || (int32_t)(g_nb_garde[k].fn - g_nb_garde[k_rec].fn) > 0)) k_rec
 ↪  = k;
+    int sur_cell = pm_sur_cellule();
+    if (k_rec >= 0 && sur_cell) {
+        memset(iq, 0, sizeof iq);
+        nb_moduler(g_nb_garde[k_rec].bits, iq, 0);
+        src = iq;
+    }
+    /* [2026-10-21 23:45] SONDE : en veille paging (Magnetite, IBA_I), la mesure de la cellule (ARFCN 1) tombe
+     * au plancher de bruit (54 -> 8, comme le voisin vide) et les barres disparaissent : la fenetre de
+     * mesure recoit-elle du remplissage faute de burst garde (k_rec<0) ou parce que le faux air la voit
+     * accordee ailleurs ? 40 premieres puis 1 sur 500. */
+    {
+        static unsigned long n, n_sans_garde, n_hors;
+        n++; if (k_rec < 0) n_sans_garde++; if (!sur_cell) n_hors++;
+        if (n <= 40 || n % 500 == 0)
+            printf("  [pm] tick=%u nwin=%d garde=%s (fn %u, %d ticks) faux-air=%d cellule=%d -> %s "
+                   "[%lu servies, %lu sans garde, %lu hors cellule]\n",
+                   g_tick_courant, nwin, k_rec >= 0 ? "oui" : "non",
+                   k_rec >= 0 ? g_nb_garde[k_rec].fn : 0u,
+                   k_rec >= 0 ? (int)(g_tick_courant - g_nb_garde[k_rec].tick) : -1,
+                   faux_air_arfcn(g_tick_courant, 5000), faux_air_cellule(),
+                   src == iq ? "burst" : "remplissage", n, n_sans_garde, n_hors);
+    }
+    calypso_rif_flush();
+    calypso_bsp_rx_burst(0, g_depot_fn, src, 2 * (nwin > 148 ? 148 : nwin));
+    g_pm_servies++;
+    return 1;
+}
+/* [2026-10-21 22:55] SONDE DMA (CALYPSO_BSP_DMA_TRACE=1) : chaque changement d'etat du canal RX (armee,
+ * one-shot, longueur) avec le tick, pendant le dedie TI. Le modele rhea-dma trace ses registres sur stderr,
+ * mais c54x_exe (verbosite.c) les avale. Sert a voir ce que la ROM arme sur le TCH (TS1), ou aucune fenetre
+ * n'etait armee au depot (appel du 21/10 22:42 : a_dd BFI, « >N200 of SABM » sur la FACCH). 3000 lignes. */
+static int pm_sur_cellule(void)
+{
+    int cell = faux_air_cellule();
+    if (cell < 0) return 1;
+    int a = faux_air_arfcn(g_tick_courant, 5000);
+    return a < 0 || a == cell;
+}
+static void dma_trace(const char *ou)
+{
+    static int on = -1; static unsigned n; static int armee_p = -1, os_p = -1, len_p = -1;
+    if (on < 0) { const char *e = calypso_getenv("CALYPSO_BSP_DMA_TRACE"); on = (e && *e == '1'); }
+    if (!on || !g_dedie_ti || n >= 3000) return;
+    int armee = calypso_rhea_dma_rx_armed(), os = calypso_rhea_dma_one_shot(), len =
 ↪  calypso_rhea_dma_get_len_words() / 2;
+    if (armee == armee_p && os == os_p && len == len_p) return;
+    armee_p = armee; os_p = os; len_p = len; n++;
+    printf("  [dma] tick=%u %s : armee=%d one_shot=%d fenetre=%d rif=%d\n", g_tick_courant, ou, armee, os, len,
 ↪  calypso_rif_level());
+}
+void calypso_bsp_dma_trace(const char *ou) { dma_trace(ou); }
+unsigned long calypso_bsp_routeur_bilan(char *buf, size_t n)
+{
+    snprintf(buf, n, " (routeur : %lu NB relivres, %lu expires, %lu mesures servies)", g_nb_relivres,
 ↪  g_nb_expires, g_pm_servies);
+    return g_nb_relivres;
+}
+int calypso_bsp_nb_retenter(void)
+{
+    dma_trace("pump");
+    if (!nb_retenter_on()) return 0;
+    /* [2026-10-21 23:40] PAS SUR LE TCH. pont.c y depose deja apres l'IDLE, fenetre armee (PONT_TCH_DEPOT_IDLE).
```

```
+     * Appel Magnetite de 23:34 : deux NB gardes relivres dans le meme tick 11084, IT DMA en attente (IFR=4000)
+     * avec le bit 14 retire de l'IMR (3000), puis la ROM part a pc=0 (SP-CORRUPT) ; la L1 TI reinitialise
+     * le DSP ~1400 trames plus tard et le mobile perd la cellule a chaque appel.
+     * CALYPSO_BSP_NB_RETENTER_TCH=1 : ancien comportement. */
+    {
+        static int sur_tch = -1;
+        if (sur_tch < 0) { const char *e = calypso_getenv("CALYPSO_BSP_NB_RETENTER_TCH"); sur_tch = (e && *e ==
 ↪  '1'); }
+        if (!sur_tch && g_dedie_tn > 0 && g_dedie_genre == BSP_DEDIE_TCH) return 0;
+    }
+    if (!calypso_rhea_dma_rx_armed() || !calypso_rhea_dma_one_shot()) return 0;
+    int nwin = calypso_rhea_dma_get_len_words() / 2;
+    if (nwin <= 0) return 0;
+    if (nwin < 150) return pm_servir(nwin);
+    if (nwin >= 190) return 0;                                  /* fenetre SB : calypso_bsp_sb_retenter */
+    /* la NB est armee : le plus ancien burst garde encore frais (2 ticks) */
+    int k_min = -1;
+    for (int k = 0; k < NB_GARDE_N; k++) {
+        if (!g_nb_garde[k].valide) continue;
+        if ((int32_t)(g_tick_courant - g_nb_garde[k].tick) > 2) { g_nb_garde[k].valide = 0; g_nb_expires++;
 ↪  continue; }
+        if (k_min < 0 || (int32_t)(g_nb_garde[k].fn - g_nb_garde[k_min].fn) < 0) k_min = k;
+    }
+    if (k_min < 0) return 0;
+    int marge = nb_marge_ti();
+    static int16_t iq[2 * 256];
+    int total = nwin > marge + 148 ? nwin : marge + 148;
+    if (total > 256) total = 256;
+    memset(iq, 0, sizeof iq);
+    nb_moduler(g_nb_garde[k_min].bits, iq, marge);
+    calypso_rif_flush();
+    calypso_bsp_rx_burst(0, g_nb_garde[k_min].fn, iq, 2 * total);
+    bsp.bursts_written++;
+    g_nb_garde[k_min].valide = 0;
+    g_nb_relivres++;
+    if (g_nb_relivres <= 40 || g_nb_relivres % 500 == 0)
+        printf("  [nbwin] NB fn=%u p51=%u RELIVRE au tick %u (depose au tick %u) : fenetre=%d [%lu relivres, %lu
 ↪  expires, %lu mesures servies]\n",
+               g_nb_garde[k_min].fn, g_nb_garde[k_min].fn % 51u, g_tick_courant, g_nb_garde[k_min].tick, nwin,
+               g_nb_relivres, g_nb_expires, g_pm_servies);
+    return 1;
+}
 /* pont.c : tracer CHAQUE SCH pendant `ticks` ticks (apres une detection FB, ou
  * quand l'ARM poste une tache SB), pour voir le depot exact autour des tentatives. */
 void calypso_bsp_sb_trace(unsigned ticks)
@@ -2978,6 +3421,9 @@ void calypso_bsp_rx_burst(uint8_t tn, uint32_t fn,
         return;
     }
     if (n_int16 <= 0 || iq == NULL) return;
+    if (calypso_rhea_dma_one_shot() && calypso_rhea_dma_get_len_words() / 2 > 0 &&
+        calypso_rhea_dma_get_len_words() / 2 < 150)
+        bsp_pm_mav_noter(iq, n_int16);    /* fenetre de mesure : voir calypso_bsp_rssi_apm_k() */

     /* [2026-09-20] FULL TIMESLOT. A bare 148-symbol burst (296 words) is padded
      * with the 8.25 guard symbols of its timeslot (156 symbols, 157 on
diff --git a/hw/arm/calypso/l1-dsp/calypso_bsp.h b/hw/arm/calypso/l1-dsp/calypso_bsp.h
index b4e684a3..628e69a0 100644
--- a/hw/arm/calypso/l1-dsp/calypso_bsp.h
+++ b/hw/arm/calypso/l1-dsp/calypso_bsp.h
@@ -111,6 +114,8 @@ bool calypso_bsp_fb_stream_next(uint16_t *outI, uint16_t *outQ);
  * a_pm = calib_RF(20*log10(MAV/MAV_REF) + RF_REF). It stands in for the ABB
  * power register the DSP reads, which the emulated ADC does not provide. */
 uint16_t calypso_bsp_rssi_apm(void);
+/* [2026-10-21] la k-ieme mesure de la trame (fenetres de mesure dans l'ordre), voir calypso_bsp.c */
+uint16_t calypso_bsp_rssi_apm_k(int k);
```

```
 int calypso_bsp_fenetre_tn(void);
 #endif /* HW_ARM_CALYPSO_BSP_H */
```

3.303 /opt/GSM/osmo-operator/patches/qosmo-calypso-faux-air-arfcn.patch

5813 octets, 148 lignes → 148 lignes

```
diff --git a/hw/arm/calypso/calypso_tpu.c b/hw/arm/calypso/calypso_tpu.c
index c779de4f..3bcd5e67 100644
--- a/hw/arm/calypso/calypso_tpu.c
+++ b/hw/arm/calypso/calypso_tpu.c
@@ -81,6 +81,12 @@ static void seq_run(uint32_t fn)
     seq.active = false;
 }

+/* [2026-10-21] position courante du sequenceur dans la trame (calypso_tsp.c : faux air) */
+int calypso_tpu_qbit(void)
+{
+    return seq.qbit;
+}
+
 void calypso_tpu_run_scenario(uint16_t *tpu_ram, uint32_t fn, uint16_t *tpu_regs)
 {
     memcpy(seq.insns, tpu_ram, sizeof(seq.insns));
diff --git a/hw/arm/calypso/calypso_trf6151.c b/hw/arm/calypso/calypso_trf6151.c
index dad03b3b..53638ded 100644
--- a/hw/arm/calypso/calypso_trf6151.c
+++ b/hw/arm/calypso/calypso_trf6151.c
@@ -43,7 +43,10 @@ static int trf6151_gain_from_reg(uint16_t reg_rx)

 void calypso_trf6151_tsp_write(uint8_t dev_idx, uint32_t word)
 {
-    if (dev_idx != TRF6151_TSP_DEV) {
+    /* [2026-10-21 18:50] TSPEN1 (osmocom, Pirelli) ou TSPEN2 : le firmware TI des Motorola C1xx
+     * adresse le TRF6151 sur le peripherique TSP 2 (tpudrv12.h, TC1_DEVICE_RF = TC1_DEVICE2) ; ignore,
+     * l'ARFCN restait inconnu et le faux air ne filtrait rien. */
+    if (dev_idx != TRF6151_TSP_DEV && dev_idx != 2) {
         return;
     }
     switch (word & TRF6151_REG_ADDR_MASK) {
@@ -101,6 +104,13 @@ uint16_t calypso_trf6151_apm_for_rf(int target_rf_dbm)
         bb_dbm = 0;
     }
     apm = bb_dbm * 64;
+    /* [2026-10-21 19:25] jamais 0 : un vrai DSP mesure toujours du bruit. a_pm = 0 compte comme une
+     * panne pour la L1 TI (l1_check_pm_error : 100 PM nuls -> l1_initialize_for_recovery, DSP relance
+     * toutes les ~173 trames en balayant des frequences vides). La L1 lit pm = a_pm >> 5
+     * (l1ddsp_meas_read) : plancher 64 (1 dB au-dessus de 0), pm = 2. */
+    if (apm < 64) {
+        apm = 64;
+    }
     if (apm > 0xFFFF) {
         apm = 0xFFFF;
     }
diff --git a/hw/arm/calypso/calypso_tsp.c b/hw/arm/calypso/calypso_tsp.c
index 79c596c9..94cc9006 100644
--- a/hw/arm/calypso/calypso_tsp.c
+++ b/hw/arm/calypso/calypso_tsp.c
@@ -2,6 +2,37 @@
 #include "qemu/osdep.h"
 #include "hw/arm/calypso/calypso_trf6151.h"
 #include "hw/arm/calypso/calypso_tsp.h"
+#include "hw/arm/calypso/calypso_rf_arfcn.h"
+#include <sys/mman.h>
```

```
+#include <fcntl.h>
+
+
+/* [2026-10-21] FAUX AIR : chaque programmation du synthetiseur part vers c54x_exe
+ * (calypso_rf_arfcn.h). Ouvert a la premiere ecriture ; sans /dev/shm, rien. */
+static void rf_arfcn_publier(uint32_t fn)
+{
+    static CalypsoRfArfcnShm *m;
+    static int essaye;
+    if (!m && !essaye) {
+        essaye = 1;
+        int fd = open(CALYPSO_RF_ARFCN_SHM, O_RDWR | O_CREAT, 0644);
+        if (fd >= 0) {
+            if (ftruncate(fd, sizeof *m) == 0) {
+                void *p = mmap(NULL, sizeof *m, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
+                if (p != MAP_FAILED) { m = p; m->magic = CALYPSO_RF_ARFCN_MAGIC; m->widx = 0; }
+            }
+            close(fd);
+        }
+    }
+    if (!m) {
+        return;
+    }
+    CalypsoRfProg *e = &m->e[m->widx % CALYPSO_RF_ARFCN_N];
+    e->fn = fn;
+    e->qbit = (int16_t)calypso_tpu_qbit();
+    e->arfcn = (int16_t)calypso_trf6151_arfcn();
+    __atomic_store_n(&m->widx, m->widx + 1, __ATOMIC_RELEASE);
+}

 static struct {
     uint8_t tx[4];
@@ -47,6 +78,9 @@ void calypso_tsp_move(uint8_t addr, uint8_t data, uint32_t fn)
                         ((uint32_t)tsp.tx[2] << 8) | tsp.tx[3];
             }
             calypso_trf6151_tsp_write(dev_idx, dout);
+            if ((dev_idx == 1 || dev_idx == 2) && ((dout & 7) == 1 || (dout & 7) == 2)) {   /* TRF6151 PLL ou PWR
 ↪  (bande), TSPEN1 ou 2 */
+                rf_arfcn_publier(fn);
+            }
         }
         break;
     default:
diff --git a/include/hw/arm/calypso/calypso_rf_arfcn.h b/include/hw/arm/calypso/calypso_rf_arfcn.h
new file mode 100644
index 00000000..af142baa
--- /dev/null
+++ b/include/hw/arm/calypso/calypso_rf_arfcn.h
@@ -0,0 +1,31 @@
+/* SPDX-License-Identifier: GPL-2.0-or-later */
+/* [2026-10-21] FAUX AIR : l'ARFCN sur lequel le telephone est accorde, de QEMU vers c54x_exe.
+ *
+ * Le synthetiseur (TRF6151, registres PLL/PWR) est programme par le TPU, donc dans QEMU
+ * (calypso_tsp.c), alors que les bursts entrent dans le DSP dans c54x_exe (calypso_bsp.c).
+ * QEMU ecrit chaque programmation (trame, position dans la trame en qbits, ARFCN) dans un
+ * anneau en memoire partagee ; c54x_exe y retrouve l'ARFCN de la fenetre qu'il remplit et
+ * livre du bruit hors de la frequence de la cellule (CALYPSO_FAUX_AIR=<arfcn>).
+ * Un fichier par instance : chaque instance a son propre /dev/shm. */
+#ifndef CALYPSO_RF_ARFCN_H
+#define CALYPSO_RF_ARFCN_H
+
+#include <stdint.h>
+
+#define CALYPSO_RF_ARFCN_SHM    "/dev/shm/calypso_rf_arfcn"
+#define CALYPSO_RF_ARFCN_MAGIC  0x52464131u   /* "RFA1" */
+#define CALYPSO_RF_ARFCN_N      64u
```

```
+
+typedef struct {
+    uint32_t fn;        /* trame QEMU (celle du PONT_TICK) */
+    int16_t  qbit;      /* position du MOVE dans la trame (sequenceur TPU) */
+    int16_t  arfcn;     /* calypso_trf6151_arfcn(), -1 si inconnu */
+} CalypsoRfProg;
+
+typedef struct {
+    uint32_t magic;
+    volatile uint32_t widx;                   /* nombre total d'entrees ecrites */
+    CalypsoRfProg e[CALYPSO_RF_ARFCN_N];
+} CalypsoRfArfcnShm;
+
+#endif
diff --git a/include/hw/arm/calypso/calypso_tsp.h b/include/hw/arm/calypso/calypso_tsp.h
index 68b3cab2..6ac3484e 100644
--- a/include/hw/arm/calypso/calypso_tsp.h
+++ b/include/hw/arm/calypso/calypso_tsp.h
@@ -23,4 +23,7 @@ bool calypso_tsp_owns_addr(uint8_t addr);

 void calypso_tsp_move(uint8_t addr, uint8_t data, uint32_t fn);

+/* position du sequenceur TPU dans la trame, en qbits (calypso_tpu.c) */
+int calypso_tpu_qbit(void);
+
 #endif
```

3.304 /opt/GSM/osmo-operator/patches/qosmo-calypso-sim-ef-table.patch

22369 octets, 486 lignes → 486 lignes

```
diff --git a/hw/arm/calypso/calypso_comp128.c b/hw/arm/calypso/calypso_comp128.c
new file mode 100644
index 00000000..fcb03ed2
--- /dev/null
+++ b/hw/arm/calypso/calypso_comp128.c
@@ -0,0 +1,252 @@
+/* [2026-10-21] calypso_comp128.c — COMP128v1 pour la carte SIM emulee (calypso_sim.c, RUN GSM ALGORITHM).
+ * Copie de libosmocore src/gsm/comp128.c (QEMU ne lie pas libosmogsm). Avant : SRES = RAND ^ 0xAA, Kc = 0,
+ * la Ki chargee de mobile.cfg n etait jamais utilisee -> LOCATION UPDATING REJECT du MSC (COMP128v1). */
+/*! \file comp128.c
+ *  COMP128 v1; common/old GSM Authentication Algorithm (A3/A8).
+ *
+ * This code is inspired by original code from :
+ *  Marc Briceno <marc@scard.org>, Ian Goldberg <iang@cs.berkeley.edu>,
+ *  and David Wagner <daw@cs.berkeley.edu>
+ *
+ * But it has been fully rewritten from various PDFs found online describing
+ * the algorithm because the licence of the code referenced above was unclear.
+ * A comment snippet from the original code is included below, it describes
+ * where the doc came from and how the algorithm was reverse engineered.
+ *
+ * This code derived from a leaked document from the GSM standards.
+ * Some missing pieces were filled in by reverse-engineering a working SIM.
+ * We have verified that this is the correct COMP128 algorithm.
+ *
+ * The first page of the document identifies it as
+ *
+ *     _Technical Information: GSM System Security Study_.
+ *     10-1617-01, 10th June 1988.
+ *
+ * The bottom of the title page is marked
+ *
+ *     Racal Research Ltd.
```

```
+ *     Worton Drive, Worton Grange Industrial Estate,
+ *     Reading, Berks. RG2 0SB, England.
+ *     Telephone: Reading (0734) 868601   Telex: 847152
+ *
+ * The relevant bits are in Part I, Section 20 (pages 66--67).  Enjoy!
+ *
+ * Note: There are three typos in the spec (discovered by
+ * reverse-engineering).
+ * - First, "z = (2 * x[n] + x[n]) mod 2^(9-j)" should clearly read
+ *   "z = (2 * x[m] + x[n]) mod 2^(9-j)".
+ * - Second, the "k" loop in the "Form bits from bytes" section is severely
+ *   botched: the k index should run only from 0 to 3, and clearly the range
+ *   on "the (8-k)th bit of byte j" is also off (should be 0..7, not 1..8,
+ *   to be consistent with the subsequent section).
+ * - Third, SRES is taken from the first 8 nibbles of x[], not the last 8 as
+ *   claimed in the document.  (And the document doesn't specify how Kc is
+ *   derived, but that was also easily discovered with reverse engineering.)
+ * All of these typos have been corrected in the following code.
+ */
+/*
+ * (C) 2009 by Sylvain Munaut <tnt@246tNt.com>
+ *
+ * All Rights Reserved
+ *
+ * SPDX-License-Identifier: GPL-2.0+
+ *
+ * This program is free software; you can redistribute it and/or modify
+ * it under the terms of the GNU General Public License as published by
+ * the Free Software Foundation; either version 2 of the License, or
+ * (at your option) any later version.
+ *
+ * This program is distributed in the hope that it will be useful,
+ * but WITHOUT ANY WARRANTY; without even the implied warranty of
+ * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
+ * GNU General Public License for more details.
+ *
+ */
+
+#include <string.h>
+#include <stdint.h>
+void comp128v1(const uint8_t *ki, const uint8_t *rand, uint8_t *sres, uint8_t *kc);
+void comp128(const uint8_t *ki, const uint8_t *rand, uint8_t *sres, uint8_t *kc);
+
+/*! \addtogroup auth
+ *  @{
+ * \file comp128.c */
+
+/* The compression tables (just copied ...) */
+static const uint8_t table_0[512] = {
+ 102, 177, 186, 162,   2, 156, 112,  75,  55,  25,   8,  12, 251, 193, 246, 188,
+ 109, 213, 151,  53,  42,  79, 191, 115, 233, 242, 164, 223, 209, 148, 108, 161,
+ 252,  37, 244,  47,  64, 211,   6, 237, 185, 160, 139, 113,  76, 138,  59,  70,
+  67,  26,  13, 157,  63, 179, 221,  30, 214,  36, 166,  69, 152, 124, 207, 116,
+ 247, 194,  41,  84,  71,   1,  49,  14,  95,  35, 169,  21,  96,  78, 215, 225,
+ 182, 243,  28,  92, 201, 118,   4,  74, 248, 128,  17,  11, 146, 132, 245,  48,
+ 149,  90, 120,  39,  87, 230, 106, 232, 175,  19, 126, 190, 202, 141, 137, 176,
+ 250,  27, 101,  40, 219, 227,  58,  20,  51, 178,  98, 216, 140,  22,  32, 121,
+  61, 103, 203,  72,  29, 110,  85, 212, 180, 204, 150, 183,  15,  66, 172, 196,
+  56, 197, 158,   0, 100,  45, 153,   7, 144, 222, 163, 167,  60, 135, 210, 231,
+ 174, 165,  38, 249, 224,  34, 220, 229, 217, 208, 241,  68, 206, 189, 125, 255,
+ 239,  54, 168,  89, 123, 122,  73, 145, 117, 234, 143,  99, 129, 200, 192,  82,
+ 104, 170, 136, 235,  93,  81, 205, 173, 236,  94, 105,  52,  46, 228, 198,   5,
+  57, 254,  97, 155, 142, 133, 199, 171, 187,  50,  65, 181, 127, 107, 147, 226,
+ 184, 218, 131,  33,  77,  86,  31,  44,  88,  62, 238,  18,  24,  43, 154,  23,
+  80, 159, 134, 111,   9, 114,   3,  91,  16, 130,  83,  10, 195, 240, 253, 119,
+ 177, 102, 162, 186, 156,   2,  75, 112,  25,  55,  12,   8, 193, 251, 188, 246,
```

```
+ 213, 109,  53, 151,  79,  42, 115, 191, 242, 233, 223, 164, 148, 209, 161, 108,
+  37, 252,  47, 244, 211,  64, 237,   6, 160, 185, 113, 139, 138,  76,  70,  59,
+  26,  67, 157,  13, 179,  63,  30, 221,  36, 214,  69, 166, 124, 152, 116, 207,
+ 194, 247,  84,  41,   1,  71,  14,  49,  35,  95,  21, 169,  78,  96, 225, 215,
+ 243, 182,  92,  28, 118, 201,  74,   4, 128, 248,  11,  17, 132, 146,  48, 245,
+  90, 149,  39, 120, 230,  87, 232, 106,  19, 175, 190, 126, 141, 202, 176, 137,
+  27, 250,  40, 101, 227, 219,  20,  58, 178,  51, 216,  98,  22, 140, 121,  32,
+ 103,  61,  72, 203, 110,  29, 212,  85, 204, 180, 183, 150,  66,  15, 196, 172,
+ 197,  56,   0, 158,  45, 100,   7, 153, 222, 144, 167, 163, 135,  60, 231, 210,
+ 165, 174, 249,  38,  34, 224, 229, 220, 208, 217,  68, 241, 189, 206, 255, 125,
+  54, 239,  89, 168, 122, 123, 145,  73, 234, 117,  99, 143, 200, 129,  82, 192,
+ 170, 104, 235, 136,  81,  93, 173, 205,  94, 236,  52, 105, 228,  46,   5, 198,
+ 254,  57, 155,  97, 133, 142, 171, 199,  50, 187, 181,  65, 107, 127, 226, 147,
+ 218, 184,  33, 131,  86,  77,  44,  31,  62,  88,  18, 238,  43,  24,  23, 154,
+ 159,  80, 111, 134, 114,   9,  91,   3, 130,  16,  10,  83, 240, 195, 119, 253,
+}, table_1[256] = {
+  19,  11,  80, 114,  43,   1,  69,  94,  39,  18, 127, 117,  97,   3,  85,  43,
+  27, 124,  70,  83,  47,  71,  63,  10,  47,  89,  79,   4,  14,  59,  11,   5,
+  35, 107, 103,  68,  21,  86,  36,  91,  85, 126,  32,  50, 109,  94, 120,   6,
+  53,  79,  28,  45,  99,  95,  41,  34,  88,  68,  93,  55, 110, 125, 105,  20,
+  90,  80,  76,  96,  23,  60,  89,  64, 121,  56,  14,  74, 101,   8,  19,  78,
+  76,  66, 104,  46, 111,  50,  32,   3,  39,   0,  58,  25,  92,  22,  18,  51,
+  57,  65, 119, 116,  22, 109,   7,  86,  59,  93,  62, 110,  78,  99,  77,  67,
+  12, 113,  87,  98, 102,   5,  88,  33,  38,  56,  23,   8,  75,  45,  13,  75,
+  95,  63,  28,  49, 123, 120,  20, 112,  44,  30,  15,  98, 106,   2, 103,  29,
+  82, 107,  42, 124,  24,  30,  41,  16, 108, 100, 117,  40,  73,  40,   7, 114,
+  82, 115,  36, 112,  12, 102, 100,  84,  92,  48,  72,  97,   9,  54,  55,  74,
+ 113, 123,  17,  26,  53,  58,   4,   9,  69, 122,  21, 118,  42,  60,  27,  73,
+ 118, 125,  34,  15,  65, 115,  84,  64,  62,  81,  70,   1,  24, 111, 121,  83,
+ 104,  81,  49, 127,  48, 105,  31,  10,   6,  91,  87,  37,  16,  54, 116, 126,
+  31,  38,  13,   0,  72, 106,  77,  61,  26,  67,  46,  29,  96,  37,  61,  52,
+ 101,  17,  44, 108,  71,  52,  66,  57,  33,  51,  25,  90,   2, 119, 122,  35,
+}, table_2[128] = {
+ 52,  50,  44,   6,  21,  49,  41,  59,  39,  51,  25,  32,  51,  47,  52,  43,
+ 37,   4,  40,  34,  61,  12,  28,   4,  58,  23,   8,  15,  12,  22,   9,  18,
+ 55,  10,  33,  35,  50,   1,  43,   3,  57,  13,  62,  14,   7,  42,  44,  59,
+ 62,  57,  27,   6,   8,  31,  26,  54,  41,  22,  45,  20,  39,   3,  16,  56,
+ 48,   2,  21,  28,  36,  42,  60,  33,  34,  18,   0,  11,  24,  10,  17,  61,
+ 29,  14,  45,  26,  55,  46,  11,  17,  54,  46,   9,  24,  30,  60,  32,   0,
+ 20,  38,   2,  30,  58,  35,   1,  16,  56,  40,  23,  48,  13,  19,  19,  27,
+ 31,  53,  47,  38,  63,  15,  49,   5,  37,  53,  25,  36,  63,  29,   5,   7,
+}, table_3[64] = {
+  1,   5,  29,   6,  25,   1,  18,  23,  17,  19,   0,   9,  24,  25,   6,  31,
+ 28,  20,  24,  30,   4,  27,   3,  13,  15,  16,  14,  18,   4,   3,   8,   9,
+ 20,   0,  12,  26,  21,   8,  28,   2,  29,   2,  15,   7,  11,  22,  14,  10,
+ 17,  21,  12,  30,  26,  27,  16,  31,  11,   7,  13,  23,  10,   5,  22,  19,
+}, table_4[32] = {
+ 15,  12,  10,   4,   1,  14,  11,   7,   5,   0,  14,   7,   1,   2,  13,   8,
+ 10,   3,   4,   9,   6,   0,   3,   2,   5,   6,   8,   9,  11,  13,  15,  12,
+};
+
+static const uint8_t *_comp128_table[5] = { table_0, table_1, table_2, table_3, table_4 };
+
+
+static inline void
+_comp128_compression_round(uint8_t *x, int n, const uint8_t *tbl)
+{
+   int i, j, m, a, b, y, z;
+   m = 4 - n;
+   for (i=0; i<(1<<n); i++)
+       for (j=0; j<(1<<m); j++) {
+           a = j + i * (2<<m);
+           b = a + (1<<m);
+           y = (x[a] + (x[b]<<1)) & ((32<<m)-1);
+           z = ((x[a]<<1) + x[b]) & ((32<<m)-1);
+           x[a] = tbl[y];
```

```
+           x[b] = tbl[z];
+       }
+}
+
+static inline void
+_comp128_compression(uint8_t *x)
+{
+   int n;
+   for (n=0; n<5; n++)
+       _comp128_compression_round(x, n, _comp128_table[n]);
+}
+
+static inline void
+_comp128_bitsfrombytes(uint8_t *x, uint8_t *bits)
+{
+   int i;
+   memset(bits, 0x00, 128);
+   for (i=0; i<128; i++)
+       if (x[i>>2] & (1<<(3-(i&3))))
+           bits[i] = 1;
+}
+
+static inline void
+_comp128_permutation(uint8_t *x, uint8_t *bits)
+{
+   int i;
+   memset(&x[16], 0x00, 16);
+   for (i=0; i<128; i++)
+       x[(i>>3)+16] |= bits[(i*17) & 127] << (7-(i&7));
+}
+
+/*! Perform COMP128v1 algorithm
+ *  \param[in] ki Secret Key K(i) of subscriber
+ *  \param[in] rand Random Challenge
+ *  \param[out] sres user-supplied buffer for storing computed SRES value
+ *  \param[out] kc user-supplied buffer for storing computed Kc value */
+void
+comp128v1(const uint8_t *ki, const uint8_t *rand, uint8_t *sres, uint8_t *kc)
+{
+   int i;
+   uint8_t x[32], bits[128];
+
+   /* x[16-31] = RAND */
+   memcpy(&x[16], rand, 16);
+
+   /* Round 1-7 */
+   for (i=0; i<7; i++) {
+       /* x[0-15] = Ki */
+       memcpy(x, ki, 16);
+
+       /* Compression */
+       _comp128_compression(x);
+
+       /* FormBitFromBytes */
+       _comp128_bitsfrombytes(x, bits);
+
+       /* Permutation */
+       _comp128_permutation(x, bits);
+   }
+
+   /* Round 8 (final) */
+       /* x[0-15] = Ki */
+   memcpy(x, ki, 16);
+
+       /* Compression */
+   _comp128_compression(x);
```

```
+
+   /* Output stage */
+   for (i=0; i<8; i+=2)
+       sres[i>>1] = x[i]<<4 | x[i+1];
+
+   for (i=0; i<12; i+=2)
+       kc[i>>1] = (x[i + 18] << 6) |
+                  (x[i + 19] << 2) |
+                  (x[i + 20] >> 2);
+
+   kc[6] = (x[30]<<6) | (x[31]<<2);
+   kc[7] = 0;
+}
+
+
+/*! Perform COMP128v1 algorithm
+ *  \param[in] ki Secret Key K(i) of subscriber
+ *  \param[in] rand Random Challenge
+ *  \param[out] sres user-supplied buffer for storing computed SRES value
+ *  \param[out] kc user-supplied buffer for storing computed Kc value */
+void
+comp128(const uint8_t *ki, const uint8_t *rand, uint8_t *sres, uint8_t *kc)
+{
+    comp128v1(ki, rand, sres, kc);
+}
+
+/*! @} */
diff --git a/hw/arm/calypso/calypso_sim.c b/hw/arm/calypso/calypso_sim.c
index add33055..416b6f5b 100644
--- a/hw/arm/calypso/calypso_sim.c
+++ b/hw/arm/calypso/calypso_sim.c
@@ -5,9 +5,19 @@
 #include "qemu/main-loop.h"
 #include "exec/cpu-common.h"
 #include "hw/core/cpu.h"
+#include "target/arm/cpu.h"
 #include "hw/arm/calypso/calypso_sim.h"

 #define SIM_LOG(...) do { fprintf(stderr, "[sim] " __VA_ARGS__); fputc('\n', stderr); } while(0)
+static uint32_t sim_guest_pc(void)
+{
+    return (first_cpu && first_cpu->cc && first_cpu->cc->get_pc) ? (uint32_t)first_cpu->cc->get_pc(first_cpu) :
 ↪  0;
+}
+static uint32_t sim_guest_lr(void)
+{
+    ARMCPU *cpu = ARM_CPU(first_cpu);
+    return (uint32_t)cpu->env.regs[14];
+}

 #define APDU_MAX_LEN     261
 #define RX_FIFO_SIZE     512
@@ -56,11 +66,47 @@ static SimFile sim_files[] = {
         0xFF, 0xFF, 0xFF,
         0x00, 0x00,
         0xFF, 0x00 } },
+    /* [2026-10-21] EF_BCCH (liste BA) : seul l'ARFCN 1 (bit 6 de l'octet 0, MSB d'abord).
+     * A 0xFF la liste couvrait presque tous les ARFCN : la recherche balayait 1..409 au hasard. */
     { 0x6F74, 0x7F20, EF_TRANSPARENT, 16, 0,
-      { 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
-        0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF } },
+      { 0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
+        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 } },
     { 0x6F46, 0x7F20, EF_TRANSPARENT, 17, 0,
       { 0x01, 'Q','E','M','U','-','S','I','M', ' ',' ',' ',' ',' ',' ',' ',' ' } },
+    /* [2026-10-21] EF absents -> « SIM Invalid » sur Magnetite (la pile TI lit Kc et
```

```
+     * PHASE a l'activation ; « file not found » les fait echouer). GSM 11.11 :
+     * Kc = 8 octets + CKSN (0x07 = pas de cle) ; PHASE 0x03 = phase 2+ ;
+     * HPLMN 0xFF = recherche desactivee ; FPLMN vide (0xFF). */
+    { 0x6F20, 0x7F20, EF_TRANSPARENT, 9, 0,
+      { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07 } },
+    /* [2026-10-21] ECC (112) et LP (anglais) : « file not found » juste apres l'activation,
+     * le MMI affiche « SIM Invalid » (activation SIM_NO_ERROR, cause 0x80FF mesuree). */
+    { 0x6FB7, 0x7F20, EF_TRANSPARENT, 3, 0,
+      { 0x11, 0xF2, 0xFF } },
+    { 0x6F05, 0x7F20, EF_TRANSPARENT, 1, 0,
+      { 0x01 } },
+    /* [2026-10-21] EF_ADN (repertoire) : la SST annonce ADN (service 1). Sans ce fichier le driver
+     * SIM met crdFun = SIM_NO_OPERATION -> SS_INV -> CME_ERR_SimWrong -> « SIM Invalid ». */
+    { 0x6FAE, 0x7F20, EF_TRANSPARENT, 1, 0,
+      { 0x03 } },
+    { 0x6F31, 0x7F20, EF_TRANSPARENT, 1, 0,
+      { 0xFF } },
+    { 0x6F7B, 0x7F20, EF_TRANSPARENT, 12, 0,
+      { 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF } },
+    /* [2026-10-21] EF_SMS : capacite de stockage SMS (memory.total = nb d'enregistrements).
+     * Sans ce fichier g_max_messages = 0 -> ALLOC_MEMORY(0) dans smsidle_sms_cb -> « mfwAlloc invalid »
+     * et « Messages: full ». 4 enregistrements libres de 176 octets. */
+    { 0x6F3C, 0x7F20, EF_LINEAR_FIXED, 704, 176,
+      { 0 } },
+    /* [2026-10-21] SMS init : EF SMSP (1 enregistrement libre), SMSS et CBMI vides */
+    { 0x6F42, 0x7F20, EF_LINEAR_FIXED, 28, 28,
+      { 0xFF } },
+    { 0x6F43, 0x7F20, EF_TRANSPARENT, 2, 0,
+      { 0xFF, 0xFF } },
+    { 0x6F45, 0x7F20, EF_TRANSPARENT, 2, 0,
+      { 0xFF, 0xFF } },
+    { 0x6F48, 0x7F20, EF_TRANSPARENT, 2, 0,
+      { 0xFF, 0xFF } },
 };
 #define SIM_FILE_COUNT (sizeof(sim_files) / sizeof(sim_files[0]))

@@ -297,9 +343,14 @@ static int build_select_response(CalypsoSim *s, SimFile *f, uint8_t *out)
         out[p++] = 0x00;
         out[p++] = 0x00; out[p++] = 0x00;
         out[p++] = 0x00;
-        out[p++] = 0xAA;
-        out[p++] = 0x00; out[p++] = 0x00;
-        out[p++] = 0x00;
+        /* [2026-10-21] file status (octet 11) : b1 = 1 -> fichier non invalide. 0xAA (b1 = 0)
+         * faisait croire a app_check_imsi_loci_validation() que IMSI/LOCI sont invalides ->
+         * rehabilitation -> SIM_NO_OPERATION (« SIM Invalid ») */
+        out[p++] = 0xAB;
+        /* [2026-10-21] octets 13-15 (GSM 11.11 9.2.1) : longueur des donnees suivantes (2), structure,
+         * longueur d'enregistrement. Trois zeros de trop decalaient tout : structure lue 00
+         * (transparent) et longueur d'enregistrement 0 -> l'entite SIM ne lisait que l'enregistrement 1
+         * de EF SMS, capacite SMS 0 -> mfwAlloc(0) en boucle et « Messages: full » dans l'IHM. */
         out[p++] = 0x02;
         out[p++] = (f->structure == EF_TRANSPARENT) ? 0x00
                 : (f->structure == EF_LINEAR_FIXED) ? 0x01 : 0x03;
@@ -353,7 +404,7 @@ static void cmd_select(CalypsoSim *s, uint8_t p1, uint8_t p2,

     uint8_t resp[32];
     int n = build_select_response(s, f, resp);
-    SIM_LOG("SELECT 0x%04x (%s) → %d bytes pending",
+    SIM_LOG("pc=0x%08x SELECT 0x%04x (%s) → %d bytes pending", sim_guest_pc(),
             fid,
             f->structure == EF_MF ? "MF" :
             f->structure == EF_DF ? "DF" : "EF",
@@ -373,6 +424,35 @@ static void cmd_get_response(CalypsoSim *s, uint8_t le)
     s->resp_len = 0;
 }
```

```
+static void cmd_read_record(CalypsoSim *s, uint8_t rec, uint8_t mode, uint8_t le)
+{
+    SimFile *f = find_file(s->selected_ef, 0);
+    if (!f || f->structure != EF_LINEAR_FIXED) {
+        respond_sw(s, 0x94, 0x00);
+        return;
+    }
+    int nrec = f->size / f->rec_len;
+    /* [2026-10-21] modes 02 (suivant) / 03 (precedent) en plus de 04 (absolu), GSM 11.11 9.2.5 */
+    static uint16_t rec_ef; static int rec_cur;
+    if (rec_ef != s->selected_ef) { rec_ef = s->selected_ef; rec_cur = 0; }
+    if (mode == 0x02) rec = (uint8_t)(rec_cur + 1);
+    else if (mode == 0x03) rec = (uint8_t)(rec_cur > 0 ? rec_cur - 1 : nrec);
+    else if (mode != 0x04) { respond_sw(s, 0x6B, 0x00); return; }
+    if (rec < 1 || rec > nrec) {
+        respond_sw(s, 0x94, 0x02);
+        return;
+    }
+    rec_cur = rec;
+    /* enregistrements libres (0xFF) : pas de contenu stocke pour ce fichier */
+    uint8_t buf[256];
+    int n = (le == 0) ? f->rec_len : le;
+    if (n > f->rec_len) n = f->rec_len;
+    memset(buf, 0xFF, n);
+    rx_push_n(s, buf, n);
+    SIM_LOG("READ_RECORD EF=0x%04x rec=%d len=%d", s->selected_ef, rec, n);
+    respond_sw(s, 0x90, 0x00);
+}
+
 static void cmd_read_binary(CalypsoSim *s, uint8_t p1, uint8_t p2, uint8_t le)
 {
     SimFile *f = find_file(s->selected_ef, 0);
@@ -391,6 +471,29 @@ static void cmd_read_binary(CalypsoSim *s, uint8_t p1, uint8_t p2, uint8_t le)
     respond_sw(s, 0x90, 0x00);
 }

+/* [2026-10-21] UPDATE BINARY / UPDATE RECORD : refuses (6D00) jusqu'ici, le telephone ne pouvait
+ * rien ecrire (LOCI, Kc, FPLMN, SMS...). Le transparent est garde dans data[] tant qu'il y tient ;
+ * les enregistrements sont acceptes sans etre gardes (les lectures rendent des enregistrements libres). */
+static void cmd_update_binary(CalypsoSim *s, uint8_t p1, uint8_t p2, uint8_t lc, const uint8_t *data)
+{
+    SimFile *f = find_file(s->selected_ef, 0);
+    if (!f || f->structure != EF_TRANSPARENT) { respond_sw(s, 0x94, 0x08); return; }
+    int offset = (p1 << 8) | p2;
+    if (offset + lc > f->size) { respond_sw(s, 0x67, 0x00); return; }
+    if (offset + lc <= (int)sizeof f->data) memcpy(f->data + offset, data, lc);
+    SIM_LOG("UPDATE_BINARY EF=0x%04x off=%d len=%d", s->selected_ef, offset, lc);
+    respond_sw(s, 0x90, 0x00);
+}
+
+static void cmd_update_record(CalypsoSim *s, uint8_t rec, uint8_t mode, uint8_t lc)
+{
+    SimFile *f = find_file(s->selected_ef, 0);
+    if (!f || f->structure != EF_LINEAR_FIXED) { respond_sw(s, 0x94, 0x08); return; }
+    if (lc != f->rec_len) { respond_sw(s, 0x67, 0x00); return; }
+    SIM_LOG("UPDATE_RECORD EF=0x%04x rec=%d mode=%d len=%d (non garde)", s->selected_ef, rec, mode, lc);
+    respond_sw(s, 0x90, 0x00);
+}
+
 static void cmd_status(CalypsoSim *s, uint8_t le)
 {
     SimFile *df = find_file(s->selected_df, 0);
@@ -411,16 +514,25 @@ static void cmd_verify_chv(CalypsoSim *s, uint8_t p2, uint8_t lc,
     respond_sw(s, 0x90, 0x00);
 }
```

```
+void comp128v1(const uint8_t *ki, const uint8_t *rand, uint8_t *sres, uint8_t *kc);   /* calypso_comp128.c */
 static void cmd_run_gsm_algo(CalypsoSim *s, uint8_t lc, const uint8_t *data)
 {

     if (lc != 16) { respond_sw(s, 0x67, 0x00); return; }

     uint8_t resp[12];
-    for (int i = 0; i < 4; i++) resp[i]    = data[i] ^ 0xAA;
-    for (int i = 0; i < 8; i++) resp[4+i] = 0x00;
-    SIM_LOG("RUN_GSM_ALGORITHM (RAND[0]=0x%02x) → SRES[0]=0x%02x",
-            data[0], resp[0]);
+    /* [2026-10-21] le VRAI calcul : COMP128v1 avec la Ki de mobile.cfg (calypso_comp128.c). Le faux SRES
+     * (RAND ^ 0xAA, Kc 0) ne genait pas le mobile osmocom (sim test : comp128 fait dans layer23), mais le
+     * firmware TI (Magnetite) demande le calcul a la carte -> LOCATION UPDATING REJECT du MSC. */
+    if (s->ki_valid) {
+        comp128v1(s->ki, data, resp, resp + 4);
+        SIM_LOG("RUN_GSM_ALGORITHM COMP128v1 (RAND[0]=0x%02x) → SRES=%02x%02x%02x%02x Kc[0]=0x%02x",
+                data[0], resp[0], resp[1], resp[2], resp[3], resp[4]);
+    } else {
+        for (int i = 0; i < 4; i++) resp[i]    = data[i] ^ 0xAA;
+        for (int i = 0; i < 8; i++) resp[4+i] = 0x00;
+        SIM_LOG("RUN_GSM_ALGORITHM SANS Ki (CALYPSO_SIM_CFG ?) → SRES bidon RAND^0xAA");
+    }
     respond_with_data_pending(s, resp, 12);
 }

@@ -451,6 +563,9 @@ static void dispatch_apdu(CalypsoSim *s)
     switch (ins) {
     case 0xA4: cmd_select(s, p1, p2, p3, data ? data : (const uint8_t *)""); break;
     case 0xB0: cmd_read_binary(s, p1, p2, p3); break;
+    case 0xB2: cmd_read_record(s, p1, p2, p3); break;
+    case 0xD6: cmd_update_binary(s, p1, p2, p3, data ? data : (const uint8_t *)""); break;
+    case 0xDC: cmd_update_record(s, p1, p2, p3); break;
     case 0xC0: cmd_get_response(s, p3); break;
     case 0xF2: cmd_status(s, p3); break;
     case 0x20: cmd_verify_chv(s, p2, p3, data ? data : (const uint8_t *)""); break;
@@ -582,7 +697,7 @@ void calypso_sim_reg_write(CalypsoSim *s, hwaddr off, uint16_t val)
              * driver TI : la carte repondait avant que SIM_ManualStart ait mis
              * VCC, SIO et l'horloge, et la sequence ne finissait jamais. L'ATR
              * du mode manuel part sur CONF1 (SCLKEN / SRSTLEV, ci-dessous). */
-            SIM_LOG("RESET → ATR delivered (synchronous)");
+            SIM_LOG("RESET → ATR delivered (synchronous) pc=0x%08x lr=0x%08x", sim_guest_pc(), sim_guest_lr());
             s->apdu_pos = 0;
             s->apdu_expected = 0;
             s->resp_len = 0;
diff --git a/hw/arm/calypso/meson.build b/hw/arm/calypso/meson.build
index 97e185f8..ca293409 100644
--- a/hw/arm/calypso/meson.build
+++ b/hw/arm/calypso/meson.build
@@ -12,6 +12,7 @@ arm_ss.add(when: 'CONFIG_CALYPSO', if_true: files(
   'calypso_tsp.c',
   'calypso_trf6151.c',
   'calypso_sim.c',
+  'calypso_comp128.c',   # [2026-10-21] COMP128v1 de la SIM emulee (copie de libosmocore)
   'calypso_l1_dispatch.c',
   'calypso_dcch_tap.c',
   # [2026-10-19] la carte vue de l'utilisateur : clavier (ARMIO), I2C reel et
```

3.305 /opt/GSM/osmo-operator/patches/qosmo-calypso-trx-tdma-dsp-ack.patch

12761 octets, 280 lignes → 280 lignes

```
diff --git a/hw/arm/calypso/calypso_armio.c b/hw/arm/calypso/calypso_armio.c
index 3bcb17e4..c68139fd 100644
--- a/hw/arm/calypso/calypso_armio.c
+++ b/hw/arm/calypso/calypso_armio.c
@@ -105,9 +105,15 @@ static const char *armio_key_name(int bit)
     return "?";
 }

+/* [2026-10-21] Le firmware Magnetite (driver kpd, BOARD 41) masque la source par AI_MaskIT (bit
+ * ARMIO_MASKIT_KBD = 1 dans ARMIO_KBD_GPIO_MASKIT, 0x18) puis active un HISR qui lit la matrice.
+ * Sans ce masque, la ligne restait levee : le CPU reprenait l'IRQ en boucle (622 000 fois) et le
+ * HISR ne tournait jamais. La source est aussi reflétee dans ARMIO_GPIO_INT (0x16). */
 static void armio_update_irq(CalypsoArmioState *s)
 {
-    int level = (s->keys != 0) && ((s->kbc & 0x1f) == 0);
+    int active = (s->keys != 0) && ((s->kbc & 0x1f) == 0);
+    int level = active && !(s->gpio_maskit & 0x1);
+    s->gpio_int = active ? 0x1 : 0;
     qemu_set_irq(s->irq, level);
 }

@@ -284,6 +290,11 @@ static const QemuInputHandler armio_input_handler = {

 /* ---- registres ---------------------------------------------------------- */

+static uint32_t armio_guest_pc(void)
+{
+    return (first_cpu && first_cpu->cc && first_cpu->cc->get_pc) ? (uint32_t)first_cpu->cc->get_pc(first_cpu) :
 ↪  0;
+}
+
 static uint64_t armio_read(void *opaque, hwaddr offset, unsigned size)
 {
     CalypsoArmioState *s = opaque;
@@ -303,6 +314,13 @@ static uint64_t armio_read(void *opaque, hwaddr offset, unsigned size)
                 rows |= (s->keys >> (col * 5)) & 0x1f;
             }
         }
+        if (s->debug) {
+            static int nlec;
+            if (nlec++ < 40 || nlec % 2000 == 0) {
+                fprintf(stderr, "[armio] lecture KBR=0x%02x keys=0x%07x kbc=0x%02x (x%d) pc=0x%08x\n",
+                        (~rows) & 0x1f, s->keys, s->kbc, nlec, armio_guest_pc());
+            }
+        }
         return (~rows) & 0x1f;
     case ARMIO_KBC:
         return s->kbc;
@@ -328,6 +346,12 @@ static void armio_write(void *opaque, hwaddr offset, uint64_t value, unsigned si
         break;
     case ARMIO_KBC:
         s->kbc = value & 0xff;
+        if (s->debug) {
+            static int necr;
+            if (necr++ < 40 || necr % 2000 == 0) {
+                fprintf(stderr, "[armio] ecriture KBC=0x%02x (x%d) pc=0x%08x\n", s->kbc, necr, armio_guest_pc());
+            }
+        }
         armio_update_irq(s);
         break;
     case ARMIO_GPIO_INT:
@@ -335,6 +359,7 @@ static void armio_write(void *opaque, hwaddr offset, uint64_t value, unsigned si
         break;
     case ARMIO_GPIO_MASKIT:
         s->gpio_maskit = value;
+        armio_update_irq(s);
```

```
         break;
     default:
         break;
diff --git a/hw/arm/calypso/calypso_soc.c b/hw/arm/calypso/calypso_soc.c
index b998fbb8..92da4d42 100644
--- a/hw/arm/calypso/calypso_soc.c
+++ b/hw/arm/calypso/calypso_soc.c
@@ -10,6 +10,8 @@
 #include "chardev/char.h"
 #include "exec/address-spaces.h"
 #include "sysemu/sysemu.h"
+#include "sysemu/runstate.h"
+#include "hw/core/cpu.h"
 #include "hw/arm/calypso/calypso_soc.h"
 #include "hw/arm/calypso/calypso_trx.h"
 #include "hw/arm/calypso/calypso_uart.h"
@@ -111,6 +113,29 @@ static const MemoryRegionOps cntl_ops = {
     .impl = { .min_access_size = 2, .max_access_size = 2 },
 };

+/* [2026-10-21] Chien de garde : dar_reset() (seul appelant de WATCHDOG_TIM_MODE = 0xFFFFF804)
+ * ecrit 0x80F5 puis attend le reset materiel de l'ARM. Sans ce reset, le firmware boucle
+ * dans dar_reset (« gel »). On emule le reset : redemarrage de la machine (flash conservee). */
+static uint64_t wdt_read(void *opaque, hwaddr off, unsigned size)
+{
+    return 0;
+}
+
+static void wdt_write(void *opaque, hwaddr off, uint64_t val, unsigned size)
+{
+    if (off == 0x04 && (val & 0xffff) == 0x80F5) {
+        fprintf(stderr, "[wdt] chien de garde : 0x80F5 -> reset de l'ARM\n");
+        cpu_reset(first_cpu);   /* reset du CPU ARM : retour au vecteur de reset, boot repart */
+    }
+}
+
+static const MemoryRegionOps wdt_ops = {
+    .read = wdt_read,
+    .write = wdt_write,
+    .endianness = DEVICE_NATIVE_ENDIAN,
+    .valid = { .min_access_size = 1, .max_access_size = 4 },
+};
+
 static void add_stub(MemoryRegion *sys, const char *name, hwaddr base,
                      const MemoryRegionOps *ops, uint64_t size)
 {
@@ -242,6 +267,7 @@ static void calypso_soc_realize(DeviceState *dev, Error **errp)
     add_stub(sysmem, "calypso.conf", 0xFFFEF000, &stub16_ops, 0x100);
     add_stub(sysmem, "calypso.dpll", 0xFFFF9800, &stub16_ops, 0x100);
     add_stub(sysmem, "calypso.mmio_f0xx", 0xFFFFF000, &stub16_ops, 0x100);
+    add_stub(sysmem, "calypso.wdt", 0xFFFFF800, &wdt_ops, 0x100);
     add_stub(sysmem, "calypso.rhea", 0xFFFFF900, &stub16_ops, 0x100);
     add_stub(sysmem, "calypso.clkm", 0xFFFFFB00, &stub16_ops, 0x100);
     add_stub(sysmem, "calypso.rhea_dma", CALYPSO_RHEA_DMA_BASE, &stub16_ops, 0x100);
diff --git a/hw/arm/calypso/calypso_trx.c b/hw/arm/calypso/calypso_trx.c
index d43de482..9ad3c591 100644
--- a/hw/arm/calypso/calypso_trx.c
+++ b/hw/arm/calypso/calypso_trx.c
@@ -134,6 +134,7 @@ typedef struct CalypsoTRX {
     uint8_t dsp_page;
     bool bl_booted;
     unsigned bl_polls;
+    uint32_t bl_lu_fn;          /* trame de la derniere lecture de DOWNLOAD_STATUS par l'ARM (verrou ARM) */
     MemoryRegion tpu_iomem;
     MemoryRegion tpu_ram_iomem;
     uint16_t tpu_regs[CALYPSO_TPU_SIZE / 2];
```

```
@@ -490,6 +491,20 @@ static bool pont_reqref_corrigee(CalypsoTRX *s, hwaddr woff, uint16_t *out)
     return true;
 }

+static void pont_echange(CalypsoTRX *s);
+static void pont_boot_pas(CalypsoTRX *s);
+
+/* CALYPSO_PONT_BOOT_POLL (defaut 1) : voir api_read, attente de LEAD_READY */
+static bool pont_boot_poll(void)
+{
+    static int v = -1;
+    if (v < 0) {
+        const char *e = getenv("CALYPSO_PONT_BOOT_POLL");
+        v = (e && *e == '0') ? 0 : 1;
+    }
+    return v;
+}
+
 static uint64_t api_read(void *opaque, hwaddr off, unsigned size)
 {
     CalypsoTRX *s = opaque;
@@ -536,6 +551,23 @@ static uint64_t api_read(void *opaque, hwaddr off, unsigned size)
          * BL_CMD_STATUS commands itself. The emulation below, written for the
          * DSP-less shunt, would force IDLE after three reads, BEFORE the DSP
          * has seen the command, making it miss the startup COPY_BLOCK. */
+        /* [2026-10-21] TELECHARGEMENT DSP A TEMPS COMPTE EN TOURS DE BOUCLE. Le firmware TI
+         * (leadapi.c, LA_LoadPage) attend LEAD_READY dans DOWNLOAD_STATUS (= 0x0FFE) au plus
+         * LA_TIMEOUT = 10000 lectures : quelques ms sur le vrai Calypso, quelques us ici, alors
+         * que le DSP n'avance que d'une trame par echange du pont. Selon la course, le
+         * telechargement passait (DSP:3606h) ou partait en LA_ERR_TIMEOUT : DSP laisse dans son
+         * chargeur (« occupe boot »), version lue 4e2a, ni L1 ni audio (keybeep jamais fini).
+         * Avant le TDMA, une lecture de 0x0FFE sans LEAD_READY fait donc avancer le DSP d'une
+         * trame toutes les 32 lectures. CALYPSO_PONT_BOOT_POLL=0 coupe. */
+        if (off == API_BL_STATUS) {
+            s->bl_lu_fn = s->fn + 1;    /* +1 : 0 = jamais lu */
+        }
+        if (off == API_BL_STATUS && val != 1 && !s->tdma_running && pont_boot_poll()) {
+            if ((++s->bl_polls & 31) == 0) {
+                pont_boot_pas(s);
+                val = s->api_ram[API_BL_STATUS / 2];
+            }
+        }
         return val;
     }
     uint16_t rv;
@@ -659,6 +691,14 @@ void calypso_trx_dsp_reset_line(bool active)
          * parks at 0xb41c, ready for the ARM COPY_BLOCKs. This is the ONLY
          * point where the C54x is reset - never on a bootloader command, which
          * would destroy the pending command. */
+        /* [2026-10-21 18:20] Le reset part en asynchrone vers c54x_exe : le LEAD_READY (1) laisse
+         * dans 0x0FFE par le chargeur d'AVANT le reset etait lu tout de suite par l'ARM (Magnetite,
+         * l1_func.c dsp_power_on), qui y ecrivait son demarrage (2, adresse 0x7000) ; le chargeur
+         * relance ensuite par le reset reecrivait 1 (ROM b419) et le demarrage etait perdu : DSP
+         * pour toujours dans sa boucle 0xb41c. Sur le vrai Calypso le chargeur redemarre en
+         * quelques us, avant la lecture de l'ARM. On efface donc le mot : l'ARM attend le
+         * LEAD_READY du NOUVEAU chargeur (api_read fait avancer le DSP pendant cette attente). */
+        s->api_ram[API_BL_STATUS / 2] = 0;
         pont_send(s, PONT_RESET, 0, 0, 0);
         fprintf(stderr, "[trx] pont DSP : RESET_DSP relache par le firmware -> "
                 "PONT_RESET (fn=%u)\n", s->fn);
@@ -922,6 +962,17 @@ static void *wall_clock_loop(void *arg)
     return NULL;
 }

+/* [2026-10-21] CALYPSO_PONT_ARM_LOCK=1 : voir tdma_tick (attente de l'ARM sans GO force). */
+static bool pont_arm_lock(void)
```

```
+{
+    static int v = -1;
+    if (v < 0) {
+        const char *e = getenv("CALYPSO_PONT_ARM_LOCK");
+        v = (e && *e == '1') ? 1 : 0;
+    }
+    return v == 1;
+}
+
 /* TICK.b: d_dsp_page plus the one-shot DSP frame interrupt bit, consumed here. */
 static uint32_t pont_tick_b(CalypsoTRX *s)
 {
@@ -987,6 +1038,30 @@ static void pont_echange(CalypsoTRX *s)
     }
 }

+/* [2026-10-21] une trame DSP pendant que l'ARM attend LEAD_READY (api_read) : comme
+ * pont_echange, mais en ATTENDANT le DONE (20 ms au plus), sinon la boucle de l'ARM
+ * repasse avant que le DSP ait fini sa trame et rien n'avance. */
+static void pont_boot_pas(CalypsoTRX *s)
+{
+    if (s->pont_fd < 0) {
+        return;
+    }
+    if (!s->pont_pending) {
+        pont_echange(s);                /* envoie le TICK */
+    }
+    if (s->pont_pending) {
+        CalypsoPontMsg m;
+        if (pont_recv(s, &m, 20) && m.type == PONT_DONE) {
+            s->pont_pending = false;
+            s->pont_frames++;
+            if (m.a & PONT_DONE_API_IRQ) {
+                s->pont_api_irqs++;
+                qemu_irq_raise(s->irqs[CALYPSO_IRQ_API]);
+            }
+        }
+    }
+}
+
 static bool arm_first(void)
 {
     static int v = -1;
@@ -1104,6 +1179,25 @@ static void tdma_tick(void *opaque)
             timer_mod_ns(s->tdma_timer, qemu_clock_get_ns(QEMU_CLOCK_REALTIME) + PONT_RETRY_NS);
             return;
         }
+        /* [2026-10-21] CALYPSO_PONT_ARM_LOCK=1 (Magnetite) : ne pas forcer le GO tant que l'ARM n'a pas
+         * fini l1_sync. Forcer fait arriver les trames plus vite que la L1 ne les traite : la file L1
+         * se remplit (9/9), « IRQ write attempt to L1 queue failed » -> os_SystemError (gel). */
+        /* [2026-10-21 18:10] borne : au demarrage (fn=3) l'ARM prend l'IT trame sans que l1_sync
+         * tourne encore (EOI 2/3) ; sans borne le pont attendait l'ARM et l'ARM le DSP, pour toujours.
+         * Au-dela de 64 x PONT_ATTENTES_MAX (~5 s) le GO part quand meme. */
+        /* [2026-10-21 19:15] pas d'attente non plus quand l'ARM sonde DOWNLOAD_STATUS (relance du DSP par
+         * le firmware, dsp_power_on) : l1_sync ne tourne pas, attendre ne faisait que figer l'instance
+         * (retards de 240 puis 586 trames sur le BTS, mesures). */
+        bool arm_relance_dsp = s->bl_lu_fn && (uint32_t)(s->fn + 1 - s->bl_lu_fn) <= 4;
+        if (!fini && pont_arm_lock() && !arm_relance_dsp && s->eoi_attentes < (unsigned)PONT_ATTENTES_MAX * 64) {
+            if (++s->eoi_attentes == (unsigned)PONT_ATTENTES_MAX * 64) {
+                fprintf(stderr, "[trx] pont DSP : CALYPSO_PONT_ARM_LOCK, l1_sync toujours pas finie "
+                        "(fn=%u, EOI %" PRIu64 "/%" PRIu64 ") : GO force\n",
+                        s->fn, calypso_inth_frame_eoi(), s->eoi_cible);
+            }
+            timer_mod_ns(s->tdma_timer, qemu_clock_get_ns(QEMU_CLOCK_REALTIME) + PONT_RETRY_NS);
+            return;
+        }
```

```
         if (!fini) {
             if (!s->phase_a_recue) {
                 /* no ISR answer: give the ARM its frame anyway, do not stall */
@@ -1245,6 +1339,13 @@ static void tdma_tick(void *opaque)
         timer_mod_ns(s->tdma_timer, qemu_clock_get_ns(QEMU_CLOCK_REALTIME) + PONT_RETRY_NS);
         return;
     }
+    /* [2026-10-21] Sans DSP externe (Magnetite : pas de pont), personne ne remet
+     * TPU_CTRL_DSP_EN a zero : le HISR de l1_sync voit « IT DSP en attente » a
+     * chaque trame et ecrit « MCU CPU overload » en boucle. Ici le DSP est
+     * considere fini des la trame : on acquitte le bit avant la trame suivante. */
+    if (s->pont_fd < 0) {
+        s->tpu_regs[TPU_CTRL / 2] &= (uint16_t)~TPU_CTRL_DSP_EN;
+    }
     calypso_timer_lost_frame_tick(s->fn);
     qemu_irq_raise(s->irqs[CALYPSO_IRQ_TPU_FRAME]);
     timer_mod_ns(s->frame_irq_timer,
```

3.306 /opt/GSM/osmo-operator/patches/qosmo-gardien-bts0.patch

2697 octets, 53 lignes → 53 lignes

```
diff --git a/run_modules/09-teardown.sh b/run_modules/09-teardown.sh
index b4f62a65..acb224e9 100644
--- a/run_modules/09-teardown.sh
+++ b/run_modules/09-teardown.sh
@@ -6,6 +6,7 @@ MOD_TIMEOUT[teardown]=25
 _td_patterns() {
     cat <<'EOF'
 f:qemu-system-arm
+f:gardien-bts
 x:osmo-bts-trx
 x:osmocon
 x:mobile
diff --git a/run_modules/60-bts.sh b/run_modules/60-bts.sh b/run_modules/60-bts.sh
index 4eccac07..9f0371eb 100644
--- a/run_modules/60-bts.sh
+++ b/run_modules/60-bts.sh
@@ -48,7 +48,25 @@ mod_bts_status() {
 mod_bts_start() {
     mkdir -p "${RUN_DIR:-/tmp/calypso}" "$(dirname "$BTS_LOG")" 2>/dev/null || true
     : > "$BTS_LOG" 2>/dev/null || true
-    setsid stdbuf -oL -eL $(rt_prefixe) "$OSMO_BTS_TRX" -c "$BTS_CFG" >>"$BTS_LOG" 2>&1 </dev/null &
+    # [2026-10-23] GARDIEN (BTS_GARDIEN=1 par defaut) : osmo-bts-trx SORT (exit 1) sur « PC clock skew too high »
+    # quand l'horloge du pont, cadencee par le DSP, cale plus de 50 trames (run du 23/10 17:37, Magnetite : apres
+    # une liberation SDCCH, BTS0 partie, plus aucune voie descendante, reseau perdu jusqu'a la relance du banc).
+    # La boucle la relance apres BTS_GARDIEN_PAUSE s ; elle attend si une autre BTS de la meme config tourne
+    # deja (c54x_exe/run.sh la relance aussi en --bts). Le pid du fichier est celui de la boucle : mod_bts_stop
+    # tue la boucle d'abord, puis la BTS.
+    if [ "${BTS_GARDIEN:-1}" = 1 ]; then
+        setsid bash -c '
+            bin="$1" cfg="$2" log="$3" pause="$4"; shift 4
+            while :; do
+                if pgrep -f -- "osmo-bts-trx -c $cfg\$" >/dev/null 2>&1; then sleep 2; continue; fi
+                stdbuf -oL -eL "$@" "$bin" -c "$cfg" </dev/null; rc=$?
+                printf "\n[gardien bts] %s : osmo-bts-trx sorti (code %s), relance dans %s s\n" "$(date +%T)"
 ↪  "$rc" "$pause"
+                sleep "$pause"
+            done' gardien-bts "$OSMO_BTS_TRX" "$BTS_CFG" "$BTS_LOG" "${BTS_GARDIEN_PAUSE:-3}" $(rt_prefixe) \
+            >>"$BTS_LOG" 2>&1 </dev/null &
+    else
+        setsid stdbuf -oL -eL $(rt_prefixe) "$OSMO_BTS_TRX" -c "$BTS_CFG" >>"$BTS_LOG" 2>&1 </dev/null &
+    fi
     printf '%s\n' "$!" > "${RUN_DIR:-/tmp/calypso}/bts.pid"
```

```
     mod_ok
 }
@@ -85,6 +103,9 @@ mod_bts_stop() {
     local pidf="${RUN_DIR:-/tmp/calypso}/bts.pid" pid _p
     pid="$(cat "$pidf" 2>/dev/null || echo 0)"
     [ "$pid" != 0 ] && kill "$pid" 2>/dev/null
+    for _p in $(pgrep -f '^bash -c .*gardien-bts' 2>/dev/null) $(pgrep -f "gardien-bts .*$BTS_CFG" 2>/dev/null);
 ↪  do
+        _bts_est_le_notre "$_p" && kill "$_p" 2>/dev/null
+    done
     for _p in $(pgrep -x osmo-bts-trx 2>/dev/null); do
         _bts_est_le_notre "$_p" && kill "$_p" 2>/dev/null
     done
```

3.307 /opt/GSM/osmo-operator/pont/init.py

2663 octets, 76 lignes → 76 lignes

```python
import gc
import logging
import signal
import sys
import threading
import time

from . import config as _config
from .cipher import Cipher
from .downlink import Downlink, Feeder
from .gsm import load_timeslots, plan_for
from .record import Recorder
from .state import Dedicated, Tch
from .stats import Reporter, Stats
from .trx import Clock, Transmitter, Trx
from .uplink import TchScheduler, Uplink

log = logging.getLogger("pont")


def _guarded(name, fn):
    def run():
        while True:
            try:
                fn()
                log.warning("thread %s termine", name)
                return
            except Exception:
                log.exception("thread %s : exception, relance dans 1 s", name)
                time.sleep(1)
    return threading.Thread(target=run, name=name, daemon=True)


def main(argv=None):
    cfg = _config.parse(argv)
    logging.basicConfig(stream=sys.stdout, level=logging.INFO,
                        format="%(asctime)s [pont] %(message)s", datefmt="%H:%M:%S")
    gc.disable()
    gc.freeze()

    stats = Stats()
    clock = Clock()
    cipher = Cipher(cfg.kc_retention, stats)
    dedicated = Dedicated()
    tch = Tch(stats)
    record = Recorder(cfg, clock) if cfg.record else None
```

```
    trx = Trx(cfg, clock, stats, cipher, record, dedicated, tch)
    transmitter = Transmitter(cfg, clock, trx, stats)
    feeder = Feeder(cfg, stats)
    timeslots = load_timeslots(cfg.bsc_cfg)
    downlink = Downlink(cfg, stats, cipher, dedicated, tch, feeder, record, timeslots)
    uplink = Uplink(cfg, clock, stats, dedicated, tch, transmitter, feeder, cipher)
    scheduler = TchScheduler(cfg, clock, stats, tch, uplink, transmitter)

    log.info("pont TRX : ports %d/%d/%d, ARFCN %d, BSIC %d, avance UL %d trames",
             cfg.trx_base, cfg.trx_base + 1, cfg.trx_base + 2, cfg.arfcn, cfg.bsic, cfg.ul_fn_advance)
    for tn in sorted(timeslots):
        plan = plan_for(timeslots[tn])
        log.info("TS%d %-12s %s", tn, timeslots[tn], ("plan %s, %d sous-voies" % (plan.name, plan.n_sub)) if plan
↪   else "")
    log.info("QEMU : GSMTAP udp/%d, SCH udp/%d ; tap GSMTAP %s", cfg.gsmtap_port, cfg.sch_port,
             ("udp/%d" % cfg.tap_port) if cfg.tap else "coupe")

    threads = [
        _guarded("ctrl", trx.run_ctrl),
        _guarded("clock", trx.run_clock),
        _guarded("data", lambda: trx.run_data(downlink.dispatch)),
        transmitter, uplink, scheduler,
        Reporter(stats, clock, dedicated, record),
    ]
    if record:
        signal.signal(signal.SIGUSR1, record.split)
        threads.append(record)
    for t in threads:
        t.start()
    while True:
        time.sleep(3600)
```

3.308 /opt/GSM/osmo-operator/pont/air.py

19501 octets, 384 lignes → 384 lignes

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""air.py - l'air entre deux Calypsos emules : les bursts que l'un emet, l'autre les recoit.

[2026-10-19] Montage a deux instances (c54x_exe/run.sh INSTANCE=2) : l'instance 1 est le
TRX de la BTS#0 (firmware trx, start-native.sh --osmo-trx), l'instance 2 est le mobile.
Chaque c54x_exe publie les bursts finaux que sa ROM emet dans un anneau
/dev/shm/calypso_tx_rom (pont.c, tx_tsp_publier) : en-tete w(4) slots(4), puis des cases de
160 octets : seq(4) fn(4) type(1) tsc(1) offset(1) src(1) bits(148). Chaque BSP (calypso_bsp.c)
attend les bursts qu'il doit recevoir en TRXDv0 sur UDP (6702, CALYPSO_BSP_PORT) :
tn(1) fn(4, gros-boutiste) rssi(1) toa(2) bits(148, un octet 0/1 par bit).

Ce relais lit l'anneau de A et l'envoie au BSP de B, et l'anneau de B au BSP de A : le
descendant (FB, SB, dummy, NB du TRX) va au mobile, le montant (RACH, NB du mobile) va au TRX.
Il part de la fin des anneaux (pas de rejeu), et compte ce qui passe toutes les 5 s.

  python3 air.py --a-ring /dev/shm/calypso_tx_rom --a-port 6702 \\
                 --b-ring /dev/shm/calypso2/calypso_tx_rom --b-port 6712
Depuis l'instance 2 (run.sh, espace de montage prive) l'anneau de l'instance 1 se lit en
/run/calypso-shm-hote/calypso_tx_rom.

[2026-10-20] Troisieme Calypso, le telephone 2 du TRX (TS1, instance 3 : --c-ring / --c-port) : ses bursts
vont au mobile avec tn=1 ; les bursts du mobile vont au TRX du TS0 (A) tant qu'il est sur le SDCCH, au TRX
du TS1 (C) une fois l'ASSIGNMENT COMMAND passe. L'air lit cet ASSIGNMENT COMMAND dans les blocs SDCCH/4 du
TS0 (pont.gsm, libosmocoding) et l'annonce au c54x_exe du mobile par --tch-cfg (format de pont/state.py :
seq, tn, tsc, arfcn), comme le faisait le pont ; un RACH du mobile rend le SDCCH (tn=0).

[2026-10-21] Le telephone 2 a son PROPRE compteur de trames (il part de 0 a son boot, comme le 1) : ses bursts
```

```
sortent de son anneau avec son fn local, et son BSP attend les bursts montants dans sa numerotation. Le
transceiver hote mesure D = fn1 - fn2 sur les DATA_IND des deux telephones et l'ecrit dans --fn-delta
(/tmp/osmocom_fn_delta_2) ; l'air tague C->B avec fn + D et envoie B->C avec fn - D, en tn 0 (le BSP du
telephone 2 range un burst tn=1 dans les « autres » intervalles, rx[2], que son firmware ne lit pas :
pour lui le TS1 est SON intervalle). Tant que D est inconnu, rien ne passe par C.
"""
import argparse
import os
import socket
import struct
import sys
import time

# [2026-10-20] le decodeur de blocs du pont (libosmocoding par ctypes) pour lire l'ASSIGNMENT COMMAND
sys.path.insert(0, os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
try:
    from pont import gsm
except Exception as _e:   # pragma: no cover - sans libosmocoding l'air relaie quand meme, sans TCH
    gsm = None
    print("air : pont.gsm indisponible (%s) : pas de detection de l'ASSIGNMENT COMMAND" % _e, flush=True)

HDR = 8
SLOT = 160
TYPES = {0: "?", 1: "RACH", 2: "NB", 3: "SB", 4: "DUMMY", 5: "FB", 101: "TS1", 102: "TS2", 103: "TS3", 104: "TS4",
 ↪  105: "TS5", 106: "TS6", 107: "TS7"}


class Anneau:
    """Un anneau calypso_tx_rom ; on ne lit que ce qui est ecrit apres notre arrivee."""

    def __init__(self, path, nom):
        self.path, self.nom = path, nom
        self.fd = -1
        self.seq = None          # derniere seq lue
        self.slots = 256
        self.rate = 0
        # [2026-10-20] CONTINUITE DU fn. En mode TRX le fn publie est le tick du c54x_exe emetteur
        # (le firmware trx ne pose pas a_a5fn) : il repart de 0 quand on relance banc-trx, et le BSP
        # du mobile (STREAM : un burst par trame, dans l'ordre des fn) attendait alors un fn de
        # 357073 en voyant arriver 15000 (« le DSP court devant le BTS », 61200 manques). On relaie
        # donc fn + decalage, le decalage etant recale a chaque recul du fn pour que la numerotation
        # vue du recepteur reste monotone et continue (+1 par rapport au dernier fn relaye).
        self.fn_dec = 0
        self.fn_dernier = None
        self.recalages = 0

    def fn_continu(self, fn):
        out = (fn + self.fn_dec) % 2715648
        if self.fn_dernier is not None:
            d = (out - self.fn_dernier) % 2715648
            if d > 100000:                # recul (ou saut absurde) : on recale juste apres le dernier ; d == 0 =
             ↪  un autre slot de la meme trame
                self.fn_dec = (self.fn_dernier + 1 - fn) % 2715648
                out = (fn + self.fn_dec) % 2715648
                self.recalages += 1
                print("air : anneau %s : fn recule (%d apres %d), decalage recale a +%d (%d recalages)"
                      % (self.nom, fn, self.fn_dernier, self.fn_dec, self.recalages), flush=True)
        self.fn_dernier = out
        return out

    def _ouvrir(self):
        if self.fd >= 0:
            return True
        try:
            self.fd = os.open(self.path, os.O_RDONLY)
        except OSError:
```

```
            return False
        hdr = os.pread(self.fd, HDR, 0)
        if len(hdr) < HDR:
            os.close(self.fd); self.fd = -1
            return False
        w, slots = struct.unpack("<II", hdr)
        self.slots = slots or 256
        self.seq = w                        # on part de la fin
        print("air : anneau %s ouvert (%s), %d cases, seq=%d" % (self.nom, self.path, self.slots, w), flush=True)
        return True

    def nouveaux(self):
        """Rend les cases (seq, fn, type, tsc, offset, src, bits) ecrites depuis le dernier appel."""
        if not self._ouvrir():
            return []
        # [2026-10-20 22:25] l'autre instance relancee (./start-native.sh --bts pendant que le mobile tourne) :
        # son c54x_exe RECREE l'anneau (nouvel inode) et l'ancien descripteur lit un fichier supprime
        # pour toujours (A->B fige, le mobile n'a plus de descendant). On compare l'inode a chaque tour.
        self.tours = getattr(self, "tours", 0) + 1
        if self.tours % 200 == 0:
            try:
                if os.stat(self.path).st_ino != os.fstat(self.fd).st_ino:
                    print("air : anneau %s recree (%s) : reouverture" % (self.nom, self.path), flush=True)
                    os.close(self.fd); self.fd = -1
                    if not self._ouvrir():
                        return []
            except OSError:
                pass
        try:
            hdr = os.pread(self.fd, HDR, 0)
        except OSError:
            os.close(self.fd); self.fd = -1
            return []
        if len(hdr) < HDR:
            return []
        w = struct.unpack("<I", hdr[:4])[0]
        if w == self.seq:
            return []
        if w < self.seq:                    # anneau recree (nouveau run)
            print("air : anneau %s remis a zero (seq %d -> %d)" % (self.nom, self.seq, w), flush=True)
            self.seq = 0
        debut = self.seq + 1
        if w - debut >= self.slots:         # trop de retard : on saute au plus recent
            self.rate += (w - debut) - (self.slots - 1)
            debut = w - (self.slots - 1)
        out = []
        for s in range(debut, w + 1):
            off = HDR + ((s - 1) % self.slots) * SLOT
            try:
                case = os.pread(self.fd, SLOT, off)
            except OSError:
                break
            if len(case) < SLOT:
                break
            seq, fn = struct.unpack("<II", case[:8])
            typ, tsc, offset, src = case[8], case[9], case[10], case[11]
            if seq != s:                    # case pas encore ecrite / ecrasee
                continue
            out.append((seq, fn, typ, tsc, offset, src, case[12:160]))
        self.seq = w
        return out


class Cible:
    """Le BSP de l'autre instance, en TRXDv0 sur UDP."""
```

```
    def __init__(self, host, port, nom, tn):
        self.addr, self.nom, self.tn = (host, port), nom, tn
        self.sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        self.envoyes = 0
        self.par_type = {}

    def envoyer(self, fn, typ, bits, tn=None):
        """tn : l'intervalle annonce dans l'en-tete TRXD ; par defaut celui de la cible. [2026-10-20] les
        anneaux portent maintenant le TS du burst (bits 4-6 de src) : on le relaie tel quel."""
        if tn is None:
            tn = self.tn
        pkt = bytes([tn & 7]) + struct.pack(">I", fn) + b"\x00\x00\x00" + bytes(b & 1 for b in bits)
        try:
            self.sock.sendto(pkt, self.addr)
        except OSError as e:
            print("air : envoi vers %s %s:%d : %s" % (self.nom, self.addr[0], self.addr[1], e), flush=True)
            return
        self.envoyes += 1
        self.par_type[typ] = self.par_type.get(typ, 0) + 1
        if tn:
            self.par_type[100 + tn] = self.par_type.get(100 + tn, 0) + 1   # compte par intervalle (TS1..7)


class Decalage:
    """D = fn1 - fn2, publie par le transceiver (--fn-delta-file) ; relu quand le fichier change."""

    def __init__(self, chemin):
        self.chemin, self.d, self.mtime, self.prochain = chemin, None, None, 0.0
        self.changements = 0

    def lire(self):
        now = time.monotonic()
        if now < self.prochain or not self.chemin:
            return self.d
        self.prochain = now + 0.5
        try:
            st = os.stat(self.chemin)
        except OSError:
            return self.d
        if st.st_mtime == self.mtime and self.d is not None:
            return self.d
        try:
            d = int(open(self.chemin).read().strip())
        except (OSError, ValueError):
            return self.d
        self.mtime = st.st_mtime
        if d != self.d:
            self.changements += 1
            print("air : decalage de trames du telephone 2 : fn1 - fn2 = %d (%s, %d changements)" % (d,
             ↪  self.chemin, self.changements), flush=True)
            self.d = d
        return self.d


SDCCH4_DL = {22: 0, 23: 0, 24: 0, 25: 0, 26: 1, 27: 1, 28: 1, 29: 1, 32: 2, 33: 2, 34: 2, 35: 2, 36: 3, 37: 3, 38:
 ↪  3, 39: 3}


class Dedie:
    """L'ASSIGNMENT COMMAND lu sur le SDCCH/4 du TS0, et l'annonce du TCH au mobile (calypso_tch_cfg)."""

    def __init__(self, chemin):
        self.chemin = chemin
        self.blocs = {}          # sous-canal -> [(fn, bits)]
        self.tn = None
        self.tsc = 7
```

```
        self.seq = 0
        self.decodes = 0
        self.rates = 0

    def _ecrire(self, tn, tsc, arfcn):
        if not self.chemin:
            return
        self.seq += 1
        buf = bytearray(16)
        buf[0:4] = int(self.seq).to_bytes(4, "little")
        buf[4] = tn & 0xff
        buf[5] = tsc & 0xff
        buf[6:8] = int(arfcn).to_bytes(2, "little")
        fd = os.open(self.chemin, os.O_CREAT | os.O_WRONLY, 0o644)
        try:
            os.pwrite(fd, bytes(buf), 0)
        finally:
            os.close(fd)

    def descendant(self, fn, typ, bits):
        """Un burst TS0 du TRX : accumule les blocs SDCCH/4 et cherche l'ASSIGNMENT COMMAND."""
        if gsm is None or typ != 2:
            return
        ss = SDCCH4_DL.get(fn % 51)
        if ss is None:
            return
        acc = self.blocs.setdefault(ss, [])
        if acc and (fn - acc[-1][0]) % 2715648 != 1:
            acc.clear()
        acc.append((fn, bytes(b & 1 for b in bits)))
        if len(acc) < 4:
            return
        quatre = acc[:]; acc.clear()
        try:
            l2 = gsm.xcch_decode([gsm.coded_from_burst(b) for _, b in quatre])
        except Exception as e:
            print("air : decodage SDCCH : %s" % e, flush=True); return
        if l2 is None:
            self.rates += 1; return
        self.decodes += 1
        off, mt = gsm.rr_message_type(l2)
        if mt == gsm.RR_ASSIGNMENT_COMMAND and len(l2) > off + 4:
            b0, b1, b2 = l2[off + 2], l2[off + 3], l2[off + 4]
            if (b1 >> 4) & 1:
                print("air : ASSIGNMENT COMMAND avec saut de frequence (fn=%u) : ignore" % fn, flush=True); return
            tn, tsc, arfcn = b0 & 0x07, (b1 >> 5) & 0x07, ((b1 & 0x03) << 8) | b2
            self.tn, self.tsc = tn, tsc
            self._ecrire(tn, tsc, arfcn)
            print("air : ASSIGNMENT COMMAND (fn=%u, SDCCH/4 ss%d) : TCH TS%d TSC=%d ARFCN=%d -> annonce %s (seq
             ↪  %d), montant du mobile vers le TRX du TS%d"
                  % (fn, ss, tn, tsc, arfcn, self.chemin, self.seq, tn), flush=True)
        elif mt == gsm.RR_CHANNEL_RELEASE and self.tn is not None:
            self.liberer("CHANNEL RELEASE sur le SDCCH")

    def liberer(self, raison):
        if self.tn is None:
            return
        tn = self.tn; self.tn = None
        self._ecrire(0, 0, 0)
        print("air : TCH TS%d rendu (%s) : montant du mobile vers le TRX du TS0 (seq %d)" % (tn, raison,
         ↪  self.seq), flush=True)


def main():
    ap = argparse.ArgumentParser(description=__doc__, formatter_class=argparse.RawDescriptionHelpFormatter)
    ap.add_argument("--a-ring", required=True, help="anneau TX de A, le TRX du TS0 (ce que A emet)")
```

```
    ap.add_argument("--a-port", type=int, required=True, help="port UDP du BSP de A (ce que A recoit)")
    ap.add_argument("--b-ring", required=True, help="anneau TX de B, le mobile")
    ap.add_argument("--b-port", type=int, required=True, help="port UDP du BSP de B")
    ap.add_argument("--c-ring", default=None, help="anneau TX de C, le telephone 2 du TRX (TS1), optionnel")
    ap.add_argument("--c-port", type=int, default=None, help="port UDP du BSP de C")
    ap.add_argument("--tch-cfg", default=None, help="annonce du TCH au c54x_exe du mobile (calypso_tch_cfg de son
↪   /dev/shm)")
    ap.add_argument("--fn-delta", default="/tmp/osmocom_fn_delta_2", help="fichier ou le transceiver publie fn1 -
↪   fn2 du telephone 2 (avec --c-ring)")
    ap.add_argument("--c-toujours", action="store_true", help="relayer C->B meme hors TCH (defaut : seulement
↪   pendant un TCH assigne, les bursts TS1 dans la trame des SB cassaient la synchro du mobile le 21/10)")
    ap.add_argument("--c-tn", type=int, default=0, help="tn de l'en-tete TRXD des bursts du mobile envoyes au
↪   telephone 2 (0 : son intervalle principal)")
    ap.add_argument("--host", default="127.0.0.1")
    ap.add_argument("--tn", type=int, default=0, help="timeslot annonce dans l'en-tete TRXD des bursts de A
↪   (defaut 0)")
    ap.add_argument("--b-fn-delta", type=int, default=int(os.environ.get("AIR_B_FN_DELTA", "0") or 0),
                    help="[21/10] trames ajoutees au fn des bursts du mobile B (defaut 0 : le fn publie par
                     ↪  c54x_exe est "
                         "le bon ; l'ecart de 2 vu le 21/10 venait d'une comparaison au RA_R de la L1 TI). Env
                      ↪  AIR_B_FN_DELTA")
    ap.add_argument("--sans", default="", help="types a ne pas relayer, ex. « ?,RACH »")
    ap.add_argument("--periode", type=float, default=0.002, help="periode de scrutation (s)")
    ap.add_argument("--stats", type=float, default=5.0, help="periode des comptes (s)")
    args = ap.parse_args()
    noms = {v: k for k, v in TYPES.items()}
    sans = {noms[t.strip()] for t in args.sans.split(",") if t.strip() in noms}

    a_ring, b_ring = Anneau(args.a_ring, "A"), Anneau(args.b_ring, "B")
    vers_b = Cible(args.host, args.b_port, "B", args.tn)   # ce que A emet -> B recoit (TS0)
    vers_a = Cible(args.host, args.a_port, "A", args.tn)   # ce que B emet -> A recoit
    c_ring = Anneau(args.c_ring, "C") if args.c_ring else None
    vers_b1 = Cible(args.host, args.b_port, "B(TS1)", 1) if c_ring else None   # ce que C emet -> B recoit, tn 1
    vers_c = Cible(args.host, args.c_port, "C", 1) if (c_ring and args.c_port) else None
    dedie = Dedie(args.tch_cfg)
    decalage = Decalage(args.fn_delta if c_ring else None)
    c_sans_d = 0
   mobile_tch = 0.0         # [2026-10-21] instant du dernier burst du mobile sur un TS dedie (tn_m != 0) : avec le
                            # SDCCH chiffre l'air ne lit plus l'ASSIGNMENT COMMAND, c'est le mobile qui dit qu'il
                              ↪  est sur le TCH
    print("air : A %s -> udp %d (B) ; B %s -> udp %d (A) ; tn %d%s%s%s" %
          (args.a_ring, args.b_port, args.b_ring, args.a_port, args.tn,
           " ; C %s -> udp %d (B, tn 1) ; B -> udp %s (C) sur TCH" % (args.c_ring, args.b_port, args.c_port) if
            ↪  c_ring else "",
           " ; annonce TCH : %s" % args.tch_cfg if args.tch_cfg else "",
           " ; sans %s" % args.sans if sans else "")
          + (" ; fn de B %+d" % args.b_fn_delta if args.b_fn_delta else ""), flush=True)
    prochain = time.monotonic() + args.stats
    while True:
        bouge = False
        for seq, fn, typ, tsc, offset, src, bits in a_ring.nouveaux():
            bouge = True
            if typ in sans or offset == 0xff and typ == 0:
                continue
            fnc = a_ring.fn_continu(fn)
            tn_b = (src >> 4) & 7                      # le TS du burst (TS0 : BCCH/CCCH/SDCCH, TS1 : le TCH)
            vers_b.envoyer(fnc, typ, bits, tn_b)
            if tn_b == 0:
                dedie.descendant(fnc, typ, bits)
        if c_ring:
            d = decalage.lire()
            for seq, fn, typ, tsc, offset, src, bits in c_ring.nouveaux():
                bouge = True
                if typ in sans or offset == 0xff and typ == 0:
                    continue
               if d is None or (dedie.tn is None and not args.c_toujours and time.monotonic() - mobile_tch > 2.0):
```

```
                    c_sans_d += 1
                    continue
                vers_b1.envoyer((fn + d) % 2715648, typ, bits)
        for seq, fn, typ, tsc, offset, src, bits in b_ring.nouveaux():
            bouge = True
            if typ in sans or offset == 0xff and typ == 0:
                continue
            fnc = (b_ring.fn_continu(fn) + args.b_fn_delta) % 2715648
            tn_m = (src >> 4) & 7                       # le TS sur lequel le mobile a emis (montant_tn_dedie)
            if typ == 1:                                    # RACH : le mobile est (re)venu sur le TS0
                if dedie.tn is not None:
                    dedie.liberer("RACH du mobile")
                vers_a.envoyer(fnc, typ, bits, 0)
            elif tn_m and vers_c is not None:
                mobile_tch = time.monotonic()
                d = decalage.lire()
                if d is None:
                    c_sans_d += 1
                    continue
                vers_c.envoyer((fnc - d) % 2715648, typ, bits, args.c_tn)   # TCH du TS1 -> telephone 2 du TRX,
↪  dans SA numerotation
            else:
                vers_a.envoyer(fnc, typ, bits, tn_m)       # un seul telephone TRX : TS0 et TS1 par tn
        if not bouge:
            time.sleep(args.periode)
        if time.monotonic() >= prochain:
            prochain += args.stats
            def fmt(c):
                return " ".join("%s=%d" % (TYPES.get(t, str(t)), n) for t, n in sorted(c.par_type.items())) or "-"
            print("air : A->B %d (%s)  B->A %d (%s)%s  rates A %d B %d%s  SDCCH blocs %d/%d%s" %
                  (vers_b.envoyes, fmt(vers_b), vers_a.envoyes, fmt(vers_a),
                   "  C->B %d (%s)  B->C %d (%s)" % (vers_b1.envoyes, fmt(vers_b1), vers_c.envoyes if vers_c else
                    ↪  0, fmt(vers_c) if vers_c else "-") if c_ring else "",
                   a_ring.rate, b_ring.rate, " C %d" % c_ring.rate if c_ring else "",
                  dedie.decodes, dedie.decodes + dedie.rates, "  TCH TS%d" % dedie.tn if dedie.tn is not None else
↪  "")
                  + ("  D=%s%s" % (decalage.d, " (sans D : %d perdus)" % c_sans_d if c_sans_d else "") if c_ring
                   ↪  else ""), flush=True)


if __name__ == "__main__":
    try:
        sys.exit(main())
    except KeyboardInterrupt:
        pass
```

3.309 /opt/GSM/osmo-operator/pont/airmesh.py

29600 octets, 595 lignes → 595 lignes

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#
# airmesh.py - le maillage se fait SUR LES BURSTS, plus sur les appels.
#
# ------------------------------------------------------------------------
# CE QUE CA CHANGE
#
# Jusqu'ici deux noeuds du banc se parlaient tout en haut de la pile : un trunk
# SIP entre deux Asterisk, un relais SMS en TCP, et du M3UA vers un hub. Chaque
# noeud avait sa propre radio, hermetique : le mobile de l'operateur 1 ne
# POUVAIT PAS entendre la BTS de l'operateur 2, parce que les bursts des deux
# ne se croisaient nulle part. Ce qu'on appelait "roaming" etait un
# raccordement telephonique entre deux reseaux qui s'ignoraient.
```

```
#
# Ici, ce qui traverse le reseau, c'est le burst. Le milieu radio devient
# commun : la BTS du noeud 2 emet, le mobile du noeud 1 l'entend, se cale
# dessus, lit ses SI, et fait sa mise a jour de localisation POUR DE VRAI. Tout
# le reste - BSC, MSC, HLR, MGW, Asterisk - reste enferme dans son noeud. Seul
# l'air est partage.
#
# ──────────────────────────────────────────────────────────────────────────────
# QUI FAIT QUOI : QEMU emet les bursts, fake_trx traduit, airmesh transporte
#
# Les bursts ne sont pas fabriques ici. Ils viennent du Calypso qui tourne dans
# QEMU : le firmware module, pont.py recupere ce que l'ARM depose dans ses
# sidebands, l'encode en bursts et le pose en TRXD. Dans l'autre sens il decode
# le descendant de la BTS et le rend au firmware. C'est LA la radio.
#
# fake_trx.py est le TRADUCTEUR de noeud : un commutateur de bursts qui relie N
# transceivers et distribue ce que l'un EMET a tous ceux dont la frequence de
# RECEPTION correspond (burst_fwd.py, forward_msg). Il ne fabrique rien, il
# aiguille.
#
# airmesh est le transport entre noeuds. Il se branche sur le traducteur, prend
# ce qui y passe, l'envoie aux pairs, et injecte ce qui en revient.
#
#     ┌──────────────────────── un noeud ────────────────────────┐
#     │   QEMU/Calypso ──sidebands── pont.py ──TRXD 5700──┐     │
#     │   osmo-bts-trx ──────────────────────TRXD 5700────┤     │
#     │   trxcon/mobile ─────────────────────TRXD 6700────┤ fake  │
#     │   airmesh(cote mobile) ──────────────TRXD 6706────┤ _trx  │
#     │   airmesh(cote BTS) ─────────────────TRXD 6709────┘     │
#     └──────────────────────────┬──────────────────────────────┘
#                                │  UDP AIRM (defaut 4739)
#                ┌───────────────┴──────────────┐
#             noeud 2                        noeud 3
#
# POURQUOI DEUX BRANCHEMENTS. fake_trx apparie sur la FREQUENCE : un
# transceiver n'entend que ce qui est emis sur sa frequence de reception. Un
# seul point d'accroche n'entendrait donc qu'un sens, et un mobile qui n'entend
# pas la BTS ne s'y attache jamais.
#   - cote mobile : rx = descendant, tx = montant. Il ENTEND ce que la BTS
#     locale emet, et INJECTE le montant des mobiles distants vers elle.
#   - cote BTS    : rx = montant, tx = descendant. Il ENTEND ce que les mobiles
#     locaux emettent, et INJECTE le descendant des BTS distantes vers eux.
#
# ──────────────────────────────────────────────────────────────────────────────
# COMMENT ON S'Y BRANCHE SANS MODIFIER UNE LIGNE DE QEMU-SRC
#
# fake_trx cree un transceiver de plus par mobile declare (60-fake-trx.sh :
# --trx ue<n>@127.0.0.<g>:<port>, port = 6700 + (n-1) * 3). Ces transceivers ne
# sont "des mobiles" que par leur nom : fake_trx ne leur connait ni role ni
# bande tant qu'ils n'ont pas envoye RXTUNE et TXTUNE.
#
# On declare donc DEUX mobiles de plus que le banc n'en a - N_MS + 2 - et
# airmesh prend leurs deux places. Il accorde la premiere comme un mobile et la
# seconde a l'envers, comme une BTS. Aucune modification du lanceur, aucun
# fichier de qosmo-grgsm touche : deux emplacements deja prevus, occupes autrement.
#
# CE QU'IL FAUT SAVOIR EN LES LISANT : `pgrep -a fake_trx` montrera deux
# mobiles qui n'existent pas, et le journal de fake_trx annoncera ue3 et ue4.
# C'est voulu. Le nom vient du lanceur, pas de nous.
#
# ──────────────────────────────────────────────────────────────────────────────
# LES TROIS DIFFICULTES, ET CE QUI EST FAIT DE CHACUNE
#
# 1. L'HORLOGE. Une trame TDMA dure 4,615 ms et porte un numero (FN) sur 22
#    bits qui reboucle toutes les 3 h 28. Deux noeuds ont deux horloges sans
#    rapport : un burst etiquete fn=12345 chez l'un ne designe aucun instant
```

```
#    chez l'autre.
#    CE QUI EST FAIT : un suivi d'ECART. airmesh mesure en continu la difference
#    entre le FN local et le FN annonce par chaque pair, et REECRIT le FN a
#    l'injection. Les deux horloges avancent au meme rythme nominal (un timer
#    logiciel de 4,615 ms des deux cotes), l'ecart est donc quasi constant et
#    la derive se rattrape a chaque burst recu.
#    CE QUI SERAIT MIEUX : un noeud maitre qui DIFFUSE le FN, les autres
#    n'ayant plus d'horloge propre. C'est la bonne reponse, elle demande de
#    prendre la main sur clck_gen ; l'ecart suffit tant que les noeuds sont sur
#    le meme LAN. La fonction _fn_offset() est le seul endroit a remplacer.
#
# 2. LE DELAI. Un burst doit arriver AVANT que l'ordonnanceur local n'en ait
#    besoin. On l'injecte donc en avance de LOOKAHEAD trames (defaut 6, soit
#    ~28 ms) : un aller-retour LAN tient largement dedans, une liaison a 30 ms
#    de latence non. Les bursts arrives trop tard sont COMPTES, jamais jetes en
#    silence - un maillage radio qui perd 5 % de ses bursts se voit dans les
#    statistiques, pas dans les journaux du BSC.
#
# 3. QUI ENTEND QUI. Sans graphe, chaque mobile entendrait toutes les BTS du
#    banc a pleine puissance : plus de cellule, plus de reselection, plus rien
#    a demontrer. data/air-mesh.txt donne l'attenuation entre noeuds, en dB.
#    Au-dela d'un plancher, le burst n'est pas transmis - c'est ce qui fait un
#    MAILLAGE et non un bus.
#
# ──────────────────────────────────────────────────────────────────────────
# LES DEUX MODES (AIRMESH_MODE)
#
#   fake  (defaut) - le burst traverse tel quel. C'est du commutation pure :
#                    aucune demodulation, aucun codage canal, le train de bits
#                    ressort identique. Exact et peu couteux.
#   pont           - le burst passe par la chaine du Calypso : demodulation en
#                    L2 (libosmocoding), remise a QEMU par les sidebands, et ce
#                    que l'ARM repond est re-module en bursts avant de repartir
#                    sur le maillage. C'est pont.py qui tient cette chaine ;
#                    airmesh lui passe la main par une FIFO et n'y touche pas.
#
# ──────────────────────────────────────────────────────────────────────────
# CE QUI N'EST PAS VALIDE
#
# Ce fichier est ecrit contre les formats et les conventions de port lus dans
# osmocom-bb (trx_toolkit : transceiver.py, data_msg.py, burst_fwd.py) et
# contre le TRXDv0 deja implemente des deux cotes par pont.py. Il n'a PAS
# encore tourne sur le banc : la partie a eprouver en premier est le suivi
# d'ecart d'horloge (point 1), qui est la seule qui repose sur une hypothese -
# que les deux clck_gen avancent au meme rythme. Les compteurs sont la pour
# ca : "hors fenetre" qui monte, c'est le point 1 ou le point 2 qui lache.

import argparse, os, socket, struct, sys, threading, time, collections

# ── TRXD v0, tel qu'osmocom-bb l'ecrit ──────────────────────────────────────
# En-tete commun : octet 0 = (version << 4) | numero de slot, puis FN sur 4
# octets gros-boutiste.  (data_msg.py, parse_hdr / append_hdr_to)
#   TxMsg (L1 -> transceiver) : en-tete commun + pwr(1)            = 6 octets
#   RxMsg (transceiver -> L1) : en-tete commun + rssi(1) + toa(2)  = 8 octets
# Puis 148 octets, un par bit (durs a l'emission, souples a la reception).
TRXD_VER      = 0
CHDR_LEN      = 5
TX_HDR_LEN    = CHDR_LEN + 1
RX_HDR_LEN    = CHDR_LEN + 3
BURST_LEN     = 148

# ── Notre protocole entre noeuds ────────────────────────────────────────────
# Volontairement plat et lisible au tcpdump : on debogue un maillage radio en
# regardant passer les paquets, pas en relisant un schema de serialisation.
#   magie(4) ver(1) sens(1) noeud(2) arfcn(2) tn(1) reserve(1)
#   fn(4) pwr(1) rssi(1) toa(2 signe) nbits(2)   = 22 octets, puis nbits octets
```

```
AIRM_MAGIC    = b"AIRM"
AIRM_VER      = 1
AIRM_HDR      = ">4sBBHHBBIbbhH"
AIRM_HDR_LEN  = struct.calcsize(AIRM_HDR)
DIR_DL        = 0      # ce qu'une BTS emet
DIR_UL        = 1      # ce qu'un mobile emet

FN_MAX        = 2715648          # 26 * 51 * 2048, le rebouclage du FN GSM
FRAME_S       = 4.615e-3


def arfcn_to_khz(arfcn, uplink=False):
    """Frequence en kHz. GSM900 / DCS1800, les deux bandes du banc.

    fake_trx n'apparie pas sur l'ARFCN mais sur la FREQUENCE : se tromper de
    bande ne donne pas une erreur, cela donne un transceiver qui n'entend
    simplement jamais rien - la panne la plus longue a trouver.
    """
    if 512 <= arfcn <= 885:                     # DCS 1800
        dl = 1805200 + 200 * (arfcn - 512)
        return dl - 95000 if uplink else dl
    if 0 <= arfcn <= 124:                       # GSM 900
        dl = 935000 + 200 * arfcn
        return dl - 45000 if uplink else dl
    if 975 <= arfcn <= 1023:                    # E-GSM 900
        dl = 935000 + 200 * (arfcn - 1024)
        return dl - 45000 if uplink else dl
    raise ValueError("ARFCN hors des bandes connues : %d" % arfcn)


class Stats:
    """Ce qu'on regarde quand le maillage "marche a moitie"."""
    def __init__(self):
        self.lock = threading.Lock()
        self.c = collections.Counter()

    def bump(self, k, n=1):
        with self.lock:
            self.c[k] += n

    def render(self):
        with self.lock:
            return " ".join("%s=%d" % (k, v) for k, v in sorted(self.c.items()))


ST = Stats()


class Topologie:
    """Qui entend qui, et a quel prix.

    data/air-mesh.txt, une arete par ligne :
        <noeud a> <noeud b> <attenuation dB> [uni]
    Sans le mot "uni" l'arete vaut dans les deux sens. Une paire absente du
    fichier ne s'entend pas : c'est un maillage, pas un bus - deux noeuds ne se
    voient que si quelqu'un l'a ecrit.
    """

    def __init__(self, chemin, defaut_db):
        self.aretes = {}
        self.defaut = defaut_db
        self.ouvert = False
        if chemin and os.path.isfile(chemin):
            self._charger(chemin)
            self.ouvert = True
```

```
    def _charger(self, chemin):
        with open(chemin, "r", encoding="utf-8", errors="replace") as f:
            for ligne in f:
                ligne = ligne.split("#", 1)[0].strip()
                if not ligne:
                    continue
                ch = ligne.split()
                if len(ch) < 3:
                    continue
                try:
                    a, b, att = int(ch[0]), int(ch[1]), float(ch[2])
                except ValueError:
                    continue
                self.aretes[(a, b)] = att
                if len(ch) < 4 or ch[3] != "uni":
                    self.aretes[(b, a)] = att

    def attenuation(self, src, dst):
        """dB entre deux noeuds, ou None si l'un n'entend pas l'autre.

        Sans fichier, tout le monde s'entend a l'attenuation par defaut : c'est
        le comportement d'un bus, utile pour un premier essai a deux noeuds, et
        il est ANNONCE au demarrage pour qu'on ne le prenne pas pour une
        topologie.
        """
        if not self.ouvert:
            return self.defaut
        return self.aretes.get((src, dst))


class TrxLien:
    """Une L1 vue de fake_trx : CTRL et DATA, exactement comme trxcon.

    Conventions de port lues dans transceiver.py (Transceiver.__init__) :
    le transceiver BINDE base+1 (CTRL) et base+2 (DATA) ; la L1 binde base+101
    et base+102 et emet vers les premiers. On est du cote L1.
    """

    def __init__(self, nom, host, base, arfcn, rx_uplink):
        self.nom  = nom
        self.host = host
        self.base = base
        self.arfcn = arfcn
        self.rx_uplink = rx_uplink        # True : on ecoute le montant
        self.ctrl = self._sock(base + 101)
        self.data = self._sock(base + 102)
        self.ctrl_seq = 0
        self.slots = set()

    @staticmethod
    def _sock(port):
        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        s.bind(("127.0.0.1", port))
        return s

    # -- CTRL ----------------------------------------------------------------
    def cmd(self, texte, attente=1.0):
        """Envoie une commande CTRL et attend sa reponse.

        On ATTEND la reponse plutot que de tirer les commandes a la suite :
        fake_trx repond RSP <cmd> <code>, et un code non nul (frequence hors
        bande, slot refuse) passait inapercu. Un transceiver mal accorde
        n'emet aucune erreur ensuite - il se tait, simplement.
        """
        self.ctrl.sendto(("CMD " + texte + "\0").encode(), (self.host, self.base + 1))
```

```
        self.ctrl.settimeout(attente)
        try:
            rep, _ = self.ctrl.recvfrom(512)
        except socket.timeout:
            return None
        finally:
            self.ctrl.settimeout(None)
        return rep.rstrip(b"\0").decode(errors="replace")

    def accorder(self, slots):
        """POWERON complet : accord des deux frequences puis ouverture des slots.

        L'ordre compte. fake_trx refuse SETSLOT sur un transceiver qui n'a pas
        de frequence, et POWERON sur un transceiver sans slot ne sert a rien :
        il tourne et n'entend rien.
        """
        rx = arfcn_to_khz(self.arfcn, uplink=self.rx_uplink)
        tx = arfcn_to_khz(self.arfcn, uplink=not self.rx_uplink)
        for cmd in ("RXTUNE %d" % rx, "TXTUNE %d" % tx):
            rep = self.cmd(cmd)
            if rep is None or " 0" not in rep:
                raise RuntimeError("%s : %s refuse (%s)" % (self.nom, cmd, rep))
        for tn in slots:
            # type 1 = TCH/F pour fake_trx ; ce qui compte ici est que le slot
            # soit ACTIF, le contenu reel est porte par les bursts eux-memes.
            self.cmd("SETSLOT %d 1" % tn)
            self.slots.add(tn)
        rep = self.cmd("POWERON")
        if rep is None or " 0" not in rep:
            raise RuntimeError("%s : POWERON refuse (%s)" % (self.nom, rep))

    def eteindre(self):
        try:
            self.cmd("POWEROFF", attente=0.3)
        except OSError:
            pass

    # -- DATA ---------------------------------------------------------------
    def lire(self):
        """Un burst que fake_trx nous livre : ce que le voisin local a EMIS."""
        paquet, _ = self.data.recvfrom(1024)
        if len(paquet) < RX_HDR_LEN + BURST_LEN:
            ST.bump("rx_court")
            return None
        ver = paquet[0] >> 4
        if ver != TRXD_VER:
            ST.bump("rx_version_inattendue")
            return None
        tn   = paquet[0] & 0x0F
        fn   = struct.unpack(">L", paquet[1:5])[0]
        rssi = -paquet[5]
        toa  = struct.unpack(">h", paquet[6:8])[0]
        bits = paquet[RX_HDR_LEN:RX_HDR_LEN + BURST_LEN]
        return tn, fn, rssi, toa, bits

    def ecrire(self, tn, fn, pwr, bits):
        """Injecte un burst : fake_trx le distribuera a qui est accorde dessus."""
        hdr = bytearray(TX_HDR_LEN)
        hdr[0] = (TRXD_VER << 4) | (tn & 0x0F)
        hdr[1:5] = struct.pack(">L", fn % FN_MAX)
        hdr[5] = pwr & 0xFF
        self.data.sendto(bytes(hdr) + bytes(bits), (self.host, self.base + 2))


class Maillage:
    def __init__(self, cfg):
```

```
    self.cfg = cfg
    self.noeud = cfg.node
    self.topo = Topologie(cfg.topologie, cfg.attenuation_defaut)
    self.pairs = cfg.pairs                      # [(id, ip, port)]
    self.stop = threading.Event()

    # L'horloge : FN local vu en dernier, et ecart estime par pair.
    self._fn_local = 0
    self._fn_vu_a  = time.monotonic()
    self._ecart    = {}                         # id pair -> ecart en trames
    self._verrou   = threading.Lock()

    self.sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    self.sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    self.sock.bind((cfg.bind, cfg.port))

    # Cote mobile : entend le descendant local, injecte le montant distant.
    self.ms  = TrxLien("airmesh-ms",  "127.0.0.1", cfg.trx_ms,
                       cfg.arfcn, rx_uplink=False)
    # Cote BTS : entend le montant local, injecte le descendant distant.
    self.bts = TrxLien("airmesh-bts", "127.0.0.1", cfg.trx_bts,
                       cfg.arfcn, rx_uplink=True)

# ── Horloge ──────────────────────────────────────────────────────
def _note_fn_local(self, fn):
    with self._verrou:
        self._fn_local = fn
        self._fn_vu_a  = time.monotonic()

def _fn_maintenant(self):
    """Le FN local, extrapole depuis le dernier burst vu.

    On n'a pas d'horloge a nous : le seul FN de reference est celui que
    fake_trx nous livre avec chaque burst. Entre deux bursts on avance a la
    montre - 4,615 ms par trame. C'est suffisant pour une avance de six
    trames ; ce serait insuffisant pour dater quoi que ce soit.
    """
    with self._verrou:
        base, quand = self._fn_local, self._fn_vu_a
    return (base + int((time.monotonic() - quand) / FRAME_S)) % FN_MAX

def _fn_offset(self, pair, fn_distant):
    """Ecart entre l'horloge d'un pair et la notre, lisse.

    C'est LE point fragile du maillage, et le seul endroit a remplacer le
    jour ou un noeud maitre diffusera le FN : ici, on suppose seulement que
    les deux horloges avancent au meme rythme.
    """
    vu = (self._fn_maintenant() - fn_distant) % FN_MAX
    with self._verrou:
        ancien = self._ecart.get(pair)
        if ancien is None:
            self._ecart[pair] = vu
            return vu
        # Moyenne glissante sur l'ecart circulaire : sans le repliement,
        # un rebouclage du FN ferait sauter l'ecart de deux millions de
        # trames et le maillage se tairait pendant trois heures.
        d = (vu - ancien + FN_MAX // 2) % FN_MAX - FN_MAX // 2
        neuf = (ancien + d // 8) % FN_MAX
        self._ecart[pair] = neuf
        return neuf

# ── Emission vers les pairs ──────────────────────────────────────
def _diffuser(self, sens, tn, fn, rssi, toa, bits):
    for (pid, ip, port) in self.pairs:
        att = self.topo.attenuation(self.noeud, pid)
```

```
        if att is None:
            ST.bump("hors_portee")
            continue
        r = max(-120, min(-1, int(rssi - att)))
        if r <= self.cfg.plancher:
            ST.bump("sous_plancher")
            continue
        paquet = struct.pack(AIRM_HDR, AIRM_MAGIC, AIRM_VER, sens,
                             self.noeud, self.cfg.arfcn, tn, 0,
                             fn % FN_MAX, 0, r, toa, len(bits)) + bytes(bits)
        try:
            self.sock.sendto(paquet, (ip, port))
            ST.bump("emis_dl" if sens == DIR_DL else "emis_ul")
        except OSError:
            ST.bump("emission_echec")

def _boucle_locale(self, lien, sens):
    """Ce que le voisin LOCAL emet part vers les pairs."""
    while not self.stop.is_set():
        try:
            lu = lien.lire()
        except OSError:
            break
        if lu is None:
            continue
        tn, fn, rssi, toa, bits = lu
        self._note_fn_local(fn)
        self._diffuser(sens, tn, fn, rssi, toa, bits)

# ── Reception depuis les pairs ──────────────────────────────────
def _boucle_maillage(self):
    while not self.stop.is_set():
        try:
            paquet, _ = self.sock.recvfrom(2048)
        except OSError:
            break
        if len(paquet) < AIRM_HDR_LEN:
            ST.bump("mesh_court")
            continue
        (magie, ver, sens, src, arfcn, tn, _res,
         fn, _pwr, rssi, toa, nbits) = struct.unpack(AIRM_HDR, paquet[:AIRM_HDR_LEN])
        if magie != AIRM_MAGIC or ver != AIRM_VER:
            ST.bump("mesh_etranger")
            continue
        bits = paquet[AIRM_HDR_LEN:AIRM_HDR_LEN + nbits]
        if len(bits) != nbits or nbits != BURST_LEN:
            ST.bump("mesh_tronque")
            continue
        if arfcn != self.cfg.arfcn:
            # Un pair sur une autre porteuse : c'est normal et c'est meme
            # le but - deux operateurs voisins n'occupent pas le meme
            # canal. On ne le compte pas comme une erreur.
            ST.bump("autre_arfcn")
            continue

        # Ou ce burst tombe-t-il dans NOTRE temps.
        fn_local = (fn + self._fn_offset(src, fn)) % FN_MAX
        cible    = (self._fn_maintenant() + self.cfg.lookahead) % FN_MAX
        retard   = (cible - fn_local + FN_MAX // 2) % FN_MAX - FN_MAX // 2
        if retard > self.cfg.tolerance:
            ST.bump("hors_fenetre_tard")
            continue
        fn_inj = cible

        if sens == DIR_DL:
            # Un descendant distant : c'est une BTS d'ailleurs. On l'injecte
```

```
                # par le cote BTS, celui dont les mobiles locaux ecoutent le tx.
                self.bts.ecrire(tn, fn_inj, self._pwr(rssi), bits)
                ST.bump("injecte_dl")
            else:
                self.ms.ecrire(tn, fn_inj, self._pwr(rssi), bits)
                ST.bump("injecte_ul")

    @staticmethod
    def _pwr(rssi):
        """TxMsg porte une attenuation (0 = pleine puissance), pas un RSSI.

        On reporte donc l'attenuation deja appliquee par la topologie, bornee
        a l'octet : c'est ce qui donne au mobile une raison de preferer une
        cellule a une autre.
        """
        return max(0, min(255, -rssi - 40))

    # ── Vie du processus ──────────────────────────────────────────────
    def _boucle_stats(self):
        while not self.stop.wait(self.cfg.stats_s):
            ecarts = " ".join("n%d:%+d" % (k, v if v < FN_MAX // 2 else v - FN_MAX)
                              for k, v in sorted(self._ecart.items()))
            print("[airmesh] fn=%d %s | ecarts %s" %
                  (self._fn_maintenant(), ST.render(), ecarts or "aucun"),
                  flush=True)

    def demarrer(self):
        print("[airmesh] noeud %d  arfcn %d  pairs %s" %
              (self.noeud, self.cfg.arfcn,
               ", ".join("%d@%s:%d" % p for p in self.pairs) or "aucun"), flush=True)
        if not self.topo.ouvert:
            print("[airmesh] AUCUNE topologie (%s) : tous les pairs s'entendent "
                  "a %.0f dB - c'est un bus, pas un maillage."
                  % (self.cfg.topologie, self.cfg.attenuation_defaut), flush=True)
        if self.cfg.mode != "fake":
            print("[airmesh] mode '%s' : les bursts passent par la chaine "
                  "Calypso de pont.py (FIFO %s)" % (self.cfg.mode, self.cfg.fifo),
                  flush=True)

        slots = [int(x) for x in self.cfg.slots.split(",") if x != ""]
        self.ms.accorder(slots)
        self.bts.accorder(slots)

        fils = [
            threading.Thread(target=self._boucle_locale, args=(self.ms, DIR_DL),
                             name="local-dl", daemon=True),
            threading.Thread(target=self._boucle_locale, args=(self.bts, DIR_UL),
                             name="local-ul", daemon=True),
            threading.Thread(target=self._boucle_maillage, name="maillage", daemon=True),
            threading.Thread(target=self._boucle_stats, name="stats", daemon=True),
        ]
        for f in fils:
            f.start()
        try:
            while not self.stop.is_set():
                time.sleep(0.5)
        except KeyboardInterrupt:
            pass
        finally:
            self.stop.set()
            self.ms.eteindre()
            self.bts.eteindre()
            print("[airmesh] arret. %s" % ST.render(), flush=True)


def paire(txt):
```

```
    """--pair 2:192.168.1.12:4739"""
    ch = txt.split(":")
    if len(ch) != 3:
        raise argparse.ArgumentTypeError("attendu <noeud>:<ip>:<port>, recu %r" % txt)
    return (int(ch[0]), ch[1], int(ch[2]))


def main():
    p = argparse.ArgumentParser(
        prog="airmesh",
        description="Maillage de bursts GSM entre noeuds : le milieu radio devient commun.")
    p.add_argument("--node", type=int, required=True, help="numero de CE noeud")
    p.add_argument("--arfcn", type=int, default=int(os.environ.get("AIRMESH_ARFCN", "514")))
    p.add_argument("--pair", dest="pairs", action="append", type=paire, default=[],
                   metavar="NOEUD:IP:PORT", help="un pair du maillage (repetable)")
    p.add_argument("--bind", default=os.environ.get("AIRMESH_BIND", "0.0.0.0"))
    p.add_argument("--port", type=int, default=int(os.environ.get("AIRMESH_PORT", "4739")))
    # 6700 + (n-1)*3 : les deux emplacements de mobile qui suivent les vrais.
    # Avec deux MS reels, le banc doit tourner en N_MS=4 et airmesh prend 3 et 4.
    p.add_argument("--trx-ms", type=int, default=int(os.environ.get("AIRMESH_TRX_MS", "6706")),
                   help="base TRXD de l'emplacement pris cote mobile (defaut : MS 3)")
    p.add_argument("--trx-bts", type=int, default=int(os.environ.get("AIRMESH_TRX_BTS", "6709")),
                   help="base TRXD de l'emplacement pris cote BTS (defaut : MS 4)")
    p.add_argument("--slots", default=os.environ.get("AIRMESH_SLOTS", "0,1,2,3,4,5,6,7"))
    p.add_argument("--lookahead", type=int, default=int(os.environ.get("AIRMESH_LOOKAHEAD", "6")),
                   help="trames d'avance a l'injection (6 = ~28 ms)")
    p.add_argument("--tolerance", type=int, default=int(os.environ.get("AIRMESH_TOL", "2")),
                   help="trames de retard tolerees avant de compter le burst perdu")
    p.add_argument("--topologie", default=os.environ.get("AIRMESH_TOPO", "data/air-mesh.txt"))
    p.add_argument("--attenuation-defaut", type=float,
                   default=float(os.environ.get("AIRMESH_ATT", "20")))
    p.add_argument("--plancher", type=int, default=int(os.environ.get("AIRMESH_PLANCHER", "-110")),
                   help="dBm sous lesquels le burst n'est pas transmis")
    # fake : le burst traverse tel quel, airmesh n'est qu'un transport.
    # pont : il passe par la chaine Calypso de pont.py - demodulation, remise a
    # QEMU, remodulation - avant de repartir. C'est le mode qui fait vraiment
    # emettre le firmware pour le compte du maillage.
    p.add_argument("--mode", choices=("fake", "pont"),
                   default=os.environ.get("AIRMESH_MODE", "fake"))
    p.add_argument("--fifo", default=os.environ.get("AIRMESH_FIFO", "/dev/shm/airmesh_pont"))
    p.add_argument("--stats-s", type=float, default=float(os.environ.get("AIRMESH_STATS", "10")))
    cfg = p.parse_args()

    try:
        Maillage(cfg).demarrer()
    except RuntimeError as e:
        print("[airmesh] %s" % e, file=sys.stderr)
        return 1
    return 0


if __name__ == "__main__":
    sys.exit(main())
```

3.310 /opt/GSM/osmo-operator/pont/cipher.py

2451 octets, 86 lignes → 86 lignes

```
import logging
import os
import struct
import threading
import time
```

```
from . import gsm

log = logging.getLogger("pont")

KC_PATH = "/dev/shm/calypso_kc_l1"
KC_TTL = 0.1


class Cipher:
    def __init__(self, retention, stats):
        self.retention = retention
        self.stats = stats
        self._lock = threading.Lock()
        self._fd = None
        self._cache = None
        self._next = 0.0
        self._held = None
        # La BTS ne chiffre le descendant qu'apres avoir recu la premiere trame
        # montante chiffree (osmo-bts l1sap.c check_for_first_ciphrd). Tant que
        # ce n'est pas constate, le descendant du canal dedie est en clair.
        self.dl_active = False

    def _read(self):
        now = time.monotonic()
        if now < self._next:
            return self._cache
        self._next = now + KC_TTL
        try:
            if self._fd is None:
                self._fd = os.open(KC_PATH, os.O_RDONLY)
            b = os.pread(self._fd, 32, 0)
        except OSError:
            if self._fd is not None:
                os.close(self._fd)
                self._fd = None
            self._cache = None
            return None
        if len(b) < 14:
            self._cache = None
            return None
        algo = b[4]
        kc = b[6:14]
        self._cache = (algo, kc) if 1 <= algo <= 3 and any(kc) else None
        return self._cache

    def current(self):
        with self._lock:
            key = self._read()
            if key:
                self._held = key
                return key
            if self.retention and self._held:
                return self._held
            return None

    def release(self, reason):
        with self._lock:
            if self._held is None:
                return
            self._held = None
            self.dl_active = False
        log.info("Kc lache (%s) : le lien repart en clair", reason)

    def confirm_dl(self, fn):
        with self._lock:
            if self.dl_active:
```

```
                return
            self.dl_active = True
        log.info("chiffrement descendant confirme par la BTS (fn=%u)", fn)

    def apply(self, burst, fn, uplink):
        key = self.current()
        if key is None:
            return burst
        dl, ul = gsm.a5_keystream(key[0], key[1], fn)
        if uplink:
            self.stats.a5_ul += 1
        else:
            self.stats.a5_dl += 1
        return gsm.a5_xor(burst, ul if uplink else dl)
```

3.311 /opt/GSM/osmo-operator/pont/config.py

3758 octets, 92 lignes → 92 lignes

```
import argparse
import os
from dataclasses import dataclass


def _env(name, default):
    return os.environ.get(name, default)


def _flag(name, default):
    return _env(name, default) not in ("0", "", "no")


@dataclass(frozen=True)
class Config:
    trx_bind: str
    trx_base: int
    dsp_port: int          # [2026-09-17] >0 : copie chaque burst DL vers le DSP (BSP UDP, 6702)
    arfcn: int
    bsic: int
    ul_fn_advance: int
    window_tol: int
    window_essais: int
    ul_retard_max: int
    rssi: int
    gsmtap_port: int
    sch_port: int
    tap: bool
    tap_port: int
    bsc_cfg: str
    clock_period: int
    kc_retention: bool
    record: bool
    record_ul: bool
    record_dl_path: str
    record_ul_path: str
    record_dl_fifo: str
    record_ul_fifo: str
    record_max_mb: int
    record_min_free_mb: int
    record_osr: int


def parse(argv=None):
    p = argparse.ArgumentParser(prog="pont", description="Pont TRX entre le BTS et la couche 1 gr-gsm de QEMU")
```

```
    p.add_argument("--arfcn", type=int, default=int(_env("PONT_ARFCN", "514")))
    p.add_argument("--bsic", type=int, default=int(_env("PONT_BSIC", "7")))
    p.add_argument("--trx-bind", default=_env("PONT_TRX_BIND", "127.0.0.1"))
    p.add_argument("--trx-base", type=int, default=int(_env("PONT_TRX_BASE", "5700")))
    p.add_argument("--dsp-port", type=int, default=int(_env("PONT_DSP_PORT", "0")),
                   help="port UDP du BSP du DSP C54x (c54x_exe --arm / qosmo-dsp) : 6702 ; 0 = coupe")
    p.add_argument("--bsc-cfg", default=_env("PONT_BSC_CFG", "/etc/osmocom/osmo-bsc.cfg"))
    p.add_argument("--no-tap", action="store_true", default=not _flag("PONT_TAP", "1"))
    p.add_argument("--no-record", action="store_true", default=not _flag("PONT_AIRREC", "1"))
    a = p.parse_args(argv)
    return Config(
        trx_bind=a.trx_bind,
        trx_base=a.trx_base,
        dsp_port=a.dsp_port,
        arfcn=a.arfcn,
        bsic=a.bsic,
        ul_fn_advance=int(_env("PONT_UL_FN_ADVANCE", "3")),
        window_tol=int(_env("PONT_WINDOW_TOL", "1")),
        # Combien de fois re-attendre un burst montant reveille trop tot
        # (horloge asservie au DSP, cf. trx.py Transmitter.run).
        window_essais=int(_env("PONT_WINDOW_ESSAIS", "12")),
        # Retard maximal (trames) d'un burst montant encore envoye a la BTS,
        # cf. trx.py Transmitter.run. 0 = jeter tout burst en retard (defaut du
        # paquet ; pont_dsp.py, le point d'entree DSP, pose 26).
        ul_retard_max=int(_env("PONT_UL_RETARD_MAX", "0")),
        rssi=int(_env("PONT_RSSI", "60")),
        # [2026-10-03] 4730 reste la cible : c'est le port, fige, de la L1 gr-gsm de
        # QEMU (calypso_l1_grgsm.c). PONT_GSMTAP_PORT permet d'intercaler
        # qosmo/tools/injecteur_bruit.py (cible grgsm, 14730 -> 4730) ; pont.py le
        # pose seul quand BRUIT_MODE vaut ber ou relais.
        gsmtap_port=int(_env("PONT_GSMTAP_PORT", "4730")),
        sch_port=4731,
        tap=not a.no_tap,
        tap_port=int(_env("PONT_TAP_PORT", "4729")),
        bsc_cfg=a.bsc_cfg,
        clock_period=int(_env("PONT_CLCK_PERIOD_FN", "51")),
        kc_retention=_flag("PONT_KC_RETENTION", "1"),
        record=not a.no_record,
        record_ul=_flag("PONT_AIRREC_UL", "1"),
        record_dl_path=_env("PONT_AIRREC_DL_PATH", "/root/record.cfile"),
        record_ul_path=_env("PONT_AIRREC_UL_PATH", "/root/record_ul.cfile"),
        record_dl_fifo=_env("PONT_AIRREC_DL_FIFO", "/tmp/iq_fft.fifo"),
        record_ul_fifo=_env("PONT_AIRREC_UL_FIFO", "/tmp/iq_fft_ms.fifo"),
        record_max_mb=int(_env("PONT_AIRREC_MAX_MB", "4096")),
        record_min_free_mb=int(_env("PONT_AIRREC_MINFREE_MB", "8192")),
        record_osr=int(_env("PONT_AIRREC_OSR", "4")),
    )
```

3.312 /opt/GSM/osmo-operator/pont/downlink.py

16116 octets, 367 lignes → 367 lignes

```
import logging
import os
import socket
import struct

from . import gsm

_dcch_gen = [0]   # [2026-10-22] generation de calypso_dcch_cfg (IMM ASS), voir CHANNEL RELEASE
# [2026-10-24] RA des derniers RACH du firmware TI (dsp/uplink.py, noter_rach) et instant de la derniere IMM ASS
# retenue : voir _imm_ass_dcch.
_rach_ra = []
_imm_ass_retenue = [0.0]
```

```
IMM_ASS_GARDE_S = float(os.environ.get("PONT_IMM_ASS_GARDE", "2.0"))


def noter_rach(ra):
    """Un RACH du firmware TI : le mobile acceptera l'IMM ASS qui repond a l'un de ses 3 derniers (44.018
     ↪  3.3.1.1.3)."""
    _rach_ra.append(ra)
    del _rach_ra[:-3]

log = logging.getLogger("pont")

TCH_DL_PATH = "/dev/shm/calypso_tch_dl"
TCH_DL_SLOTS = 16
TCH_DL_SLOT = 48
GSMTAP_HOST = "127.0.0.1"


class Feeder:
    def __init__(self, cfg, stats):
        self.cfg = cfg
        self.stats = stats
        self.sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    def l2(self, fn, chan, l2, tn=0, uplink=False, to_qemu=True):
        pkt = gsm.gsmtap_header(self.cfg.arfcn, fn, chan, tn, uplink) + bytes(l2)
        if not uplink and to_qemu:
            self.sock.sendto(pkt, (GSMTAP_HOST, self.cfg.gsmtap_port))
        if self.cfg.tap:
            self.sock.sendto(pkt, (GSMTAP_HOST, self.cfg.tap_port))

    def sch(self, fn):
        self.sock.sendto(b"SCH2" + struct.pack("<iii", self.cfg.bsic, fn, self.cfg.arfcn), (GSMTAP_HOST,
         ↪  self.cfg.sch_port))


class TchRing:
    def __init__(self):
        self.fd = None
        self.w = 0

    def publish(self, fr33, fn):
        if self.fd is None:
            self.fd = os.open(TCH_DL_PATH, os.O_CREAT | os.O_RDWR, 0o644)
            os.ftruncate(self.fd, 8 + TCH_DL_SLOTS * TCH_DL_SLOT)
            os.pwrite(self.fd, (0).to_bytes(4, "little") + TCH_DL_SLOTS.to_bytes(4, "little"), 0)
            self.w = 0
        self.w += 1
        rec = self.w.to_bytes(4, "little") + (fn & 0xffffffff).to_bytes(4, "little") + bytes(fr33)[:gsm.FR_BYTES]
        rec += bytes(TCH_DL_SLOT - len(rec))
        os.pwrite(self.fd, rec, 8 + ((self.w - 1) % TCH_DL_SLOTS) * TCH_DL_SLOT)
        os.pwrite(self.fd, self.w.to_bytes(4, "little"), 0)


class Downlink:
    def __init__(self, cfg, stats, cipher, dedicated, tch, feeder, record, timeslots):
        self.cfg = cfg
        self.stats = stats
        self.cipher = cipher
        self.dedicated = dedicated
        self.tch = tch
        self.feed = feeder
        self.record = record
        self.timeslots = timeslots
        self.blocks = {}
        self.ring = TchRing()
        self.tch_acc = []
```

```
    self.tch_last_fn = None
    self.sacch_acc = {}
    self._deciphered = {}
    tch.on_close.append(self._tch_reset)

def _tch_reset(self):
    self.tch_acc = []
    self.tch_last_fn = None
    self.sacch_acc = {}
    self._deciphered = {}

def dispatch(self, tn, fn, burst):
    if self.record:
        self.record.feed_dl(fn, tn, burst)
    if tn == 0 and fn % 51 in gsm.SCH_SLOTS_51:
        if gsm.sch_decode(burst):
            self.feed.sch(fn)
        return
    if tn == self.tch.active_tn():
        self._tch(tn, fn, burst)
        return
    phys = self.timeslots.get(tn, "")
    plan = gsm.plan_for(phys)
    if plan is not None:
        self._signalling(tn, fn, burst, plan)

def _active_subchannel(self, tn, plan):
    ded = self.dedicated.plan()
    if ded is None:
        return None, False
    dplan, ss, dtn = ded
    if dtn != tn:
        return None, True
    if dplan is not plan:
        return None, False
    return ss % plan.n_sub, False

def _signalling(self, tn, fn, burst, plan):
    m51 = fn % 51
    base = next((b for b in plan.block_bases if b <= m51 <= b + 3), None)
    if base is None:
        return
    fn0 = fn - (m51 - base)
    ss, elsewhere = self._active_subchannel(tn, plan)
    if elsewhere and base in plan.ded_bases:
        self.stats.dl_skipped += 1
        return
    if ss is not None:
        if base in plan.sdcch_dl and base != plan.sdcch_dl[ss]:
            return
        if base in plan.sacch_dl and (fn0 % 102) != plan.sacch_dl102[ss]:
            return
    if len(self.blocks) > 64:
        self.blocks.clear()
    if gsm.is_dummy(burst):
        self.stats.dl_dummy += 1
        return
    acc = self.blocks.setdefault((tn, fn0), [])
    acc.append((fn, burst))
    self.stats.dl_bursts += 1
    if len(acc) < 4:
        return
    del self.blocks[(tn, fn0)]
    l2 = self._decode(acc, base in plan.ded_bases)
    self.stats.block(tn, base, l2 is not None)
    if l2 is None:
```

```
            self.stats.dl_crc_fail += 1
            return
        self.stats.dl_blocks += 1
        if base in plan.sdcch_dl:
            self.feed.l2(fn0, gsm.GSMTAP_SDCCH4, l2, tn)
            self._rr_downlink(l2, fn0)
            return
        if base in plan.sacch_dl:
            self.feed.l2(fn0, gsm.GSMTAP_SACCH, l2, tn)
            return
        mt = None
        if len(l2) >= 3 and l2[1] == gsm.RR_PD:
            mt = l2[2]
        elif len(l2) >= 5 and l2[3] == gsm.RR_PD:
            mt = l2[4]
        if mt in (gsm.RR_IMMEDIATE_ASSIGNMENT, gsm.RR_IMMEDIATE_ASSIGNMENT_EXT):
            self.cipher.release("IMMEDIATE ASSIGNMENT")
            if mt == gsm.RR_IMMEDIATE_ASSIGNMENT:
                self._imm_ass_dcch(l2, fn0)
        if mt in gsm.SI_TYPES:
            self.feed.l2(fn0, gsm.GSMTAP_BCCH, l2, tn)
        elif mt in gsm.CCCH_TYPES:
            # [2026-09-09] LES PAGINGS VIDES NE VONT PLUS AU L1 QEMU. Le L1
            # (calypso_l1_grgsm.c, feed_agch) ne garde QU UN bloc CCCH - le
            # dernier recu, 100 trames de validite. osmo-bts remplit chaque
            # bloc PCH libre d un PAGING REQUEST 1 sans identite : le vrai
            # paging (TMSI du mobile) etait ecrase ~50 ms plus tard, et le
            # firmware ne le voyait que si sa lecture de son groupe de paging
            # tombait dans la fenetre. Mesure sur le banc : 4 appels sur 5
            # rejetes en DEST_OOO (paging expire au bout de 10 s, BSC ayant
            # emis 19 tentatives), le mobile campe et n ayant journalise
            # AUCUN « PAGING REQUEST 1 ». Le tap Wireshark, lui, garde tout.
            self.feed.l2(fn0, gsm.GSMTAP_CCCH, l2, tn,
                         to_qemu=not gsm.is_empty_paging(l2))

    def _clear(self, bursts):
        return [gsm.coded_from_burst(b) for _, b in bursts]

    def _ciphered(self, bursts):
        if len(self._deciphered) > 64:
            self._deciphered.clear()
        out = []
        for fn, b in bursts:
            d = self._deciphered.get(fn)
            if d is None:
                d = self._deciphered[fn] = gsm.coded_from_burst(self.cipher.apply(b, fn, False))
            out.append(d)
        return out

    def _decode(self, bursts, dedicated):
        # Comme dans un vrai reseau : clair, puis chiffre. La BTS n'active le
        # chiffrement descendant qu'apres la premiere trame montante chiffree,
        # donc SI5/SI6 et les premieres trames du SDCCH arrivent en clair meme
        # quand le Kc est deja connu. On decode en clair tant que ca passe, et
        # on verrouille l'etat global "descendant chiffre" des qu'un bloc ne
        # decode qu'une fois dechiffre. Tous les canaux suivent cet etat.
        if not dedicated or self.cipher.current() is None:
            return gsm.xcch_decode(self._clear(bursts))
        if not self.cipher.dl_active:
            l2 = gsm.xcch_decode(self._clear(bursts))
            if l2 is not None:
                return l2
        l2 = gsm.xcch_decode(self._ciphered(bursts))
        if l2 is not None:
            self.cipher.confirm_dl(bursts[0][0])
        return l2
```

```
    def _decode_tch(self, bursts):
        # Meme regle pour le TCH : la parole suit l'etat global, une FACCH
        # valide (CRC Fire) sert de preuve pour basculer l'etat.
        if self.cipher.current() is None:
            return gsm.tch_fr_decode(self._clear(bursts))
        if self.cipher.dl_active:
            return gsm.tch_fr_decode(self._ciphered(bursts))
        rc, fr = gsm.tch_fr_decode(self._clear(bursts))
        if rc == gsm.MACBLOCK_LEN:
            return rc, fr
        rc2, fr2 = gsm.tch_fr_decode(self._ciphered(bursts))
        if rc2 == gsm.MACBLOCK_LEN:
            self.cipher.confirm_dl(bursts[0][0])
            return rc2, fr2
        return rc, fr

    def _imm_ass_dcch(self, l2, fn):
        """[2026-10-22] Firmware TI (Magnetite) : pas de tap QEMU pour dire au banc sur quel canal dedie le
        mobile va. L'IMMEDIATE ASSIGNMENT (44.018 9.1.18) le dit : octet 1 pseudo-longueur, 2 PD, 3 type,
        4 mode de page, 5 chan_nr. SDCCH/4 = 0b00100SST, SDCCH/8 = 0b01SSSTTT. Le dernier recu gagne : la BSC
        repond aux RACH dans l'ordre, et celui que le mobile accepte (sa reference) arrive en dernier.
        CALYPSO_PONT_IMM_DCCH=0 pour couper ; seulement avec MS_MAGNETITE (le firmware osmocom a son tap).
        [2026-10-24] FAUX : le mobile prend la PREMIERE qui repond a l'un de ses RACH et ignore les suivantes.
↪ Appel du
        24/10 18:59 : 5 IMM ASS (SS0..SS4), le C139 sur SS0 (SS1..4 tombes en T3101 a la BSC), le pont sur SS4 ;
        l'ASSIGNMENT COMMAND passait sur SS0 sans etre lue, TCH jamais arme, pas de SABM sur la FACCH, « Assignment
        Failure ». On garde donc la premiere dont la reference (octet 7, RA) est un RACH du mobile, puis on ignore
        les autres pendant PONT_IMM_ASS_GARDE s (le pont lit le descendant ~30 trames avant le mobile, qui peut
        encore emettre un RACH). Sans RA connu (decodage rate), l'ancienne regle."""
        if not os.environ.get("MS_MAGNETITE") or os.environ.get("CALYPSO_PONT_IMM_DCCH", "1") == "0":
            return
        if len(l2) < 5 or l2[1] != gsm.RR_PD:
            return
        cn = l2[4]
        if (cn >> 5) == 0b001:                              # SDCCH/4 + SACCH : 0b001SSTTT, sous-canal 0..3
            kind, ss = 0, (cn >> 3) & 3
        elif (cn >> 6) == 0b01:                             # SDCCH/8 + SACCH, sous-canal 0..7
            kind, ss = 1, (cn >> 3) & 7
        else:
            return
        if _rach_ra and len(l2) > 7:
            import time
            ra = l2[7]
            if ra not in _rach_ra or time.monotonic() - _imm_ass_retenue[0] < IMM_ASS_GARDE_S:
                log.info("IMMEDIATE ASSIGNMENT (fn=%u) : SDCCH TS%d SS%d pour RA 0x%02x, ignoree (RACH du mobile :
↪ %s)",
                         fn, cn & 7, ss, ra, " ".join("0x%02x" % r for r in _rach_ra) or "-")
                return
            _imm_ass_retenue[0] = time.monotonic()
            _rach_ra.clear()
        from .state import ecrire_dcch
        _dcch_gen[0] += 1                                   # annule une liberation differee en attente
        ecrire_dcch(kind, ss, cn)
        log.info("IMMEDIATE ASSIGNMENT (fn=%u) : SDCCH/%d TS%d SS%d -> calypso_dcch_cfg (firmware TI)",
                 fn, 8 if kind else 4, cn & 7, ss)

    def _rr_downlink(self, l2, fn):
        off, mt = gsm.rr_message_type(l2)
        if mt == gsm.RR_ASSIGNMENT_COMMAND and len(l2) > off + 4:
            b0, b1, b2 = l2[off + 2], l2[off + 3], l2[off + 4]
            if (b1 >> 4) & 1:
                log.warning("ASSIGNMENT COMMAND avec saut de frequence : TCH non arme (fn=%u)", fn)
                return
            self.tch.arm(b0 & 0x07, (b1 >> 5) & 0x07, ((b1 & 0x03) << 8) | b2)
        elif mt == gsm.RR_CHANNEL_RELEASE:
```

```
        self.tch.release_requested("CHANNEL RELEASE")
        self._liberer_dcch_ti(fn, "CHANNEL RELEASE")

def _liberer_dcch_ti(self, fn, quoi):
    """Firmware TI (Magnetite) : pas de tap QEMU pour dire que le mobile a quitte le dedie ; le pont libere
    lui-meme calypso_dcch_cfg (et le TCH) apres le CHANNEL RELEASE."""
    if not os.environ.get("MS_MAGNETITE") or os.environ.get("CALYPSO_PONT_IMM_DCCH", "1") == "0":
        return
    # [2026-10-22 14:10] DIFFEREE. Le pont lit le descendant ~30 trames AVANT le mobile (marge DL
    # +28..+35) : liberer tout de suite coupait le SDCCH au BSP avant que le mobile ait recu ce
    # CHANNEL RELEASE ; il restait sur un canal mort (SA_I en erreur) jusqu'a la perte du lien
    # (~13 s, DDS_R), sourd au paging pendant ce temps (MT SMS, appels entrants perdus).
    # PONT_DCCH_REL_DELAI (s, defaut 1.5) ; une IMM ASS entre-temps annule la liberation.
    # [2026-10-22 20:55] AUSSI SUR LA FACCH. Le CHANNEL RELEASE d'un appel arrive sur le TCH : rien n'etait
    # libere, TCH tenu 20 s (RELEASE_FALLBACK) et BSP route sur le dedie apres le raccroche (DESACCORD),
    # NP_I en erreur, RECOVERY de la L1 TI, DSP relance en 2,4 s, 444 trames perdues : antenne perdue.
    import threading
    from .state import ecrire_dcch, DCCH_RELEASED
    gen = _dcch_gen[0]
    delai = float(os.environ.get("PONT_DCCH_REL_DELAI", "1.5"))
    def _liberer():
        if _dcch_gen[0] == gen:
            self.tch.close("%s, liberation differee (firmware TI)" % quoi)
            ecrire_dcch(DCCH_RELEASED, 0, 0)
            log.info("%s (fn=%u) : calypso_dcch_cfg libere apres %.1f s (firmware TI)", quoi, fn, delai)
    t = threading.Timer(delai, _liberer)
    t.daemon = True
    t.start()

def _tch(self, tn, fn, burst):
    if gsm.is_dummy(burst):
        self.stats.dl_dummy += 1
        return
    if fn % 26 == gsm.sacch_tf_frame(tn):
        self._tch_sacch(tn, fn, burst)
        return
    if not gsm.is_tch_carrier(fn):
        return
    if self.tch_last_fn is not None and fn != gsm.next_tch_carrier(self.tch_last_fn):
        self.tch_acc = []
    self.tch_last_fn = fn
    self.tch_acc.append((fn, burst))
    if len(self.tch_acc) > 8:
        del self.tch_acc[0]
    if len(self.tch_acc) < 8 or gsm.tch_burst_index(fn) % 4 != 3:
        return
    rc, fr = self._decode_tch(self.tch_acc)
    if rc == gsm.FR_BYTES:
        self.stats.tch_dl += 1
        self.ring.publish(fr, fn)
        # [2026-09-23] REFERENCE POUR LA PAROLE DU DSP. Chaque trame FR que
        # le pont decode lui-meme des bursts de la BTS, avec son fn BTS :
        # tools/comparer_parole.py la met en regard de ce que la ROM livre
        # dans a_dd (/dev/shm/calypso_add_dl.bin, montant.c) pour compter
        # les bits faux par classe. 4000 trames au plus par session.
        self._noter_parole(fn, fr)
    elif rc == gsm.MACBLOCK_LEN:
        self.stats.tch_facch_dl += 1
        l2 = fr[:gsm.MACBLOCK_LEN]
        self.feed.l2(fn, gsm.GSMTAP_TCH_F, l2, tn)
        if gsm.rr_message_type(l2)[1] == gsm.RR_CHANNEL_RELEASE:
            self.tch.release_requested("CHANNEL RELEASE (FACCH)")
            self._liberer_dcch_ti(fn, "CHANNEL RELEASE (FACCH)")
    else:
        self.stats.tch_crc += 1
```

```
    def _noter_parole(self, fn, fr):
        n = getattr(self, "_parole_n", 0)
        if n >= 4000:
            return
        try:
            mode = "ab" if n else "wb"
            with open("/dev/shm/pont_tch_dl.bin", mode) as f:
                f.write(struct.pack("<I", fn & 0xffffffff) + bytes(fr[:gsm.FR_BYTES]))
        except OSError:
            return
        self._parole_n = n + 1

    def _tch_sacch(self, tn, fn, burst):
        start = gsm.sacch_tf_block_base(tn)
        block = fn - ((fn % 104 - start) % 104)
        acc = self.sacch_acc.setdefault(block, [])
        acc.append((fn, burst))
        if len(self.sacch_acc) > 4:
            for k in sorted(self.sacch_acc)[:-2]:
                del self.sacch_acc[k]
        if len(acc) < 4:
            return
        del self.sacch_acc[block]
        l2 = self._decode(acc, True)
        if l2 is None:
            self.stats.tch_crc += 1
            return
        self.stats.tch_sacch_dl += 1
        self.feed.l2(block, gsm.GSMTAP_TCH_ACCH, l2, tn)
```

3.313 /opt/GSM/osmo-operator/pont/gsm.py

8725 octets, 258 lignes → 259 lignes (1 groupes compactés)

```
import ctypes
import struct

HYPERFRAME = 26 * 51 * 2048
FRAME_DUR = 60.0 / 13000.0
MACBLOCK_LEN = 23
FR_BYTES = 33

GSMTAP_BCCH = 0x01
GSMTAP_CCCH = 0x04
GSMTAP_SDCCH4 = 0x07
GSMTAP_TCH_F = 0x09
GSMTAP_ACCH = 0x80
GSMTAP_SACCH = GSMTAP_SDCCH4 | GSMTAP_ACCH
GSMTAP_TCH_ACCH = GSMTAP_TCH_F | GSMTAP_ACCH

RR_PD = 0x06
RR_ASSIGNMENT_COMMAND = 0x2e
RR_ASSIGNMENT_COMPLETE = 0x29
RR_CHANNEL_RELEASE = 0x0d
RR_IMMEDIATE_ASSIGNMENT = 0x3f
RR_IMMEDIATE_ASSIGNMENT_EXT = 0x39
SI_TYPES = frozenset((0x19, 0x1a, 0x1b, 0x1c, 0x1d, 0x1e))
CCCH_TYPES = frozenset((0x3f, 0x39, 0x3a, 0x21, 0x22, 0x24))


def is_empty_paging(l2):
    """PAGING REQUEST TYPE 1 sans identite : le bloc de remplissage qu osmo-bts
    emet sur chaque bloc PCH libre (Mobile Identity de longueur 1, type 0 =
    « no identity »). Rien a en faire cote mobile."""
```

```
    return (len(l2) >= 6 and l2[1] == 0x06 and l2[2] == 0x21
            and l2[4] == 0x01 and (l2[5] & 0x07) == 0x00)

TSC7 = bytes((1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0))
RACH_SYNC = bytes(int(c) for c in "01001011011111111100110011010101000111100 0")
RACH_SLOTS_51 = frozenset([4, 5] + list(range(14, 37)) + [45, 46])
SCH_SLOTS_51 = frozenset((1, 11, 21, 31, 41))
DUMMY_BURST = bytes((
    0, 0, ⟨1⟩, ⟨2⟩, ⟨3⟩, 1, ⟨4⟩, ⟨5⟩, 0, ⟨6⟩, ⟨7⟩, 0, ⟨8⟩, ⟨9⟩, ⟨10⟩, ⟨11⟩, ⟨12⟩, ⟨13⟩, ⟨14⟩, ⟨15⟩, ⟨16⟩, ⟨17⟩, 0, ⟨18⟩, 0,
     ↪  ⟨19⟩, ⟨20⟩, ⟨21⟩, ⟨22⟩, 0, 0, 1, ⟨23⟩, ⟨24⟩, ⟨25⟩, ⟨26⟩, ⟨27⟩,  ×3
    ⟨⟩ = (0,1,1,1,1,1,1,1,1,0,1,1,0,0,0,0,1,1,0,0,1,1,1,0,0,0)
        (1,0,0,0,0,1,0,0,0,0,0,0,1,1,1,1,1,0,1,1,1,0,0,1,1,1,0)
        (1,0,1,1,0,0,1,1,0,1,1,1,0,1,0,0,1,1,0,0,1,1,1,0,0,1,1)
    1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0))
DUMMY_TOLERANCE = 8

_SOFT = bytes([127, 129] + [129] * 254)
_ONEBIT = bytes([0] + [1] * 255)

ubit = ctypes.c_int8
sbit = ctypes.c_int8
_cod = ctypes.CDLL("libosmocoding.so", use_errno=True)
_cod.gsm0503_xcch_encode.argtypes = [ctypes.POINTER(ubit), ctypes.POINTER(ctypes.c_uint8)]
_cod.gsm0503_xcch_decode.argtypes = [ctypes.POINTER(ctypes.c_uint8), ctypes.POINTER(sbit),
                                     ctypes.POINTER(ctypes.c_int), ctypes.POINTER(ctypes.c_int)]
_cod.gsm0503_rach_ext_encode.argtypes = [ctypes.POINTER(ubit), ctypes.c_uint16, ctypes.c_uint8, ctypes.c_bool]
_cod.gsm0503_rach_decode_ber.argtypes = [ctypes.POINTER(ctypes.c_uint8), ctypes.POINTER(sbit), ctypes.c_uint8,
                                       ctypes.POINTER(ctypes.c_int), ctypes.POINTER(ctypes.c_int)]
_cod.gsm0503_sch_decode.argtypes = [ctypes.POINTER(ctypes.c_uint8), ctypes.POINTER(sbit)]
_cod.gsm0503_tch_fr_encode.argtypes = [ctypes.POINTER(ubit), ctypes.POINTER(ctypes.c_uint8), ctypes.c_int,
 ↪  ctypes.c_int]
_cod.gsm0503_tch_fr_decode.argtypes = [ctypes.POINTER(ctypes.c_uint8), ctypes.POINTER(sbit), ctypes.c_int,
                                       ctypes.c_int, ctypes.POINTER(ctypes.c_int), ctypes.POINTER(ctypes.c_int)]
_gsm = ctypes.CDLL("libosmogsm.so", use_errno=True)
_gsm.osmo_a5.argtypes = [ctypes.c_int, ctypes.POINTER(ctypes.c_uint8), ctypes.c_uint32,
                         ctypes.POINTER(ubit), ctypes.POINTER(ubit)]


def normalize_bits(raw148):
    return raw148.translate(_ONEBIT)


def is_dummy(burst):
    errors = 0
    for a, b in zip(burst, DUMMY_BURST):
        if a != b:
            errors += 1
            if errors > DUMMY_TOLERANCE:
                return False
    return True


def coded_from_burst(burst):
    return burst[3:61] + burst[87:145]


def burst_from_coded(coded):
    return bytes(3) + bytes(coded[:58]) + TSC7 + bytes(coded[58:116]) + bytes(3)


def _soft(bursts):
    return b"".join(bytes(b).translate(_SOFT) for b in bursts)


def xcch_decode(bursts4):
    buf = (sbit * (4 * 116)).from_buffer_copy(_soft(bursts4))
```

```
    l2 = (ctypes.c_uint8 * MACBLOCK_LEN)()
    ne = ctypes.c_int()
    nb = ctypes.c_int()
    rc = _cod.gsm0503_xcch_decode(l2, buf, ctypes.byref(ne), ctypes.byref(nb))
    return bytes(l2) if rc == 0 else None


def xcch_encode(l2):
    data = (ctypes.c_uint8 * MACBLOCK_LEN)(*l2[:MACBLOCK_LEN])
    e = (ubit * (4 * 116))()
    _cod.gsm0503_xcch_encode(e, data)
    raw = bytes(e)
    return [burst_from_coded(raw[k * 116:(k + 1) * 116]) for k in range(4)]


def tch_fr_decode(bursts8):
    buf = (sbit * (8 * 116)).from_buffer_copy(_soft(bursts8))
    out = (ctypes.c_uint8 * FR_BYTES)()
    ne = ctypes.c_int()
    nb = ctypes.c_int()
    rc = _cod.gsm0503_tch_fr_decode(out, buf, 1, 0, ctypes.byref(ne), ctypes.byref(nb))
    return rc, bytes(out)


def tch_fr_encode_into(buf24, payload):
    if payload:
        data = (ctypes.c_uint8 * len(payload))(*payload)
        _cod.gsm0503_tch_fr_encode(buf24, data, len(payload), 1)
    else:
        _cod.gsm0503_tch_fr_encode(buf24, None, 0, 1)


def sch_decode(burst):
    d78 = burst[3:42] + burst[106:145]
    buf = (sbit * 78)(*[-127 if b else 127 for b in d78])
    sb = (ctypes.c_uint8 * 4)()
    return _cod.gsm0503_sch_decode(sb, buf) == 0


def rach_burst(ra, bsic):
    coded = (ubit * 40)()
    _cod.gsm0503_rach_ext_encode(coded, ctypes.c_uint16(ra), ctypes.c_uint8(bsic), False)
    return bytes(8) + RACH_SYNC + bytes(coded)[:36] + bytes(148 - 8 - 41 - 36)


def rach_ra(burst, bsic):
    """[2026-10-24] RA d'un access burst (bits durs, 8 de queue + 41 de synchro + 36 codes), None si le CRC
    (colore par le BSIC) ne colle pas."""
    d36 = burst[8 + 41:8 + 41 + 36]
    if len(d36) != 36:
        return None
    buf = (sbit * 36)(*[-127 if b else 127 for b in d36])
    ra = ctypes.c_uint8()
    nerr, nbits = ctypes.c_int(), ctypes.c_int()
    if _cod.gsm0503_rach_decode_ber(ctypes.byref(ra), buf, ctypes.c_uint8(bsic), ctypes.byref(nerr),
     ↪  ctypes.byref(nbits)):
        return None
    return ra.value


def a5_keystream(algo, kc, fn):
    key = (ctypes.c_uint8 * 8)(*kc)
    dl = (ubit * 114)()
    ul = (ubit * 114)()
    _gsm.osmo_a5(algo, key, ctypes.c_uint32(fn & 0xffffffff), dl, ul)
    return bytes(dl), bytes(ul)
```

```
def a5_xor(burst, ks):
    b = bytearray(burst)
    for i in range(57):
        b[3 + i] ^= ks[i]
        b[88 + i] ^= ks[57 + i]
    return bytes(b)


def gsmtap_header(arfcn, fn, chan, tn, uplink):
    a = arfcn | (0x4000 if uplink else 0)
    return struct.pack(">BBBBHbBIBBBB", 2, 4, 0x01, tn, a, 0, 0, fn, chan, 0, 0, 0)


def rr_message_type(l2):
    for off in (0, 3):
        if len(l2) > off + 1 and l2[off] == RR_PD:
            if off == 3 and (l2[2] >> 2) < 2:
                continue
            return off, l2[off + 1]
    return None, None


def tch_burst_index(fn):
    m = fn % 26
    return m if m < 12 else m - 1


def is_tch_carrier(fn):
    return fn % 26 not in (12, 25)


def next_tch_carrier(fn):
    n = (fn + 1) % HYPERFRAME
    while not is_tch_carrier(n):
        n = (n + 1) % HYPERFRAME
    return n


def sacch_tf_frame(tn):
    return 12 if tn % 2 == 0 else 25


def sacch_tf_block_base(tn):
    # 45.002 table 5 : le bloc SACCH/TF du TN commence a la trame 12 + 13*TN de la 104-multitrame
    return (12 + 13 * tn) % 104


def sacch_tf_block_start(fn, tn):
    return fn % 104 == sacch_tf_block_base(tn)


class SlotPlan:
    def __init__(self, name, sdcch_dl, sacch_dl, other_dl):
        self.name = name
        self.n_sub = len(sdcch_dl)
        self.sdcch_dl = tuple(sdcch_dl)
        self.sacch_dl = tuple(sacch_dl)
        half = len(sacch_dl)
        self.sacch_dl102 = tuple(sacch_dl[i % half] + (51 if i >= half else 0) for i in range(self.n_sub))
        self.ded_bases = tuple(sorted(set(self.sdcch_dl + self.sacch_dl)))
        self.block_bases = tuple(sorted(set(self.sdcch_dl + self.sacch_dl + tuple(other_dl))))

    def ul_base(self, ss):
        return (self.sdcch_dl[ss % self.n_sub] + 15) % 51
```

```
PLAN_SDCCH4 = SlotPlan("CCCH+SDCCH4", (22, 26, 32, 36), (42, 46), (2, 6, 12, 16))
PLAN_SDCCH8 = SlotPlan("SDCCH8", (0, 4, 8, 12, 16, 20, 24, 28), (32, 36, 40, 44), ())


def plan_for(phys):
    if phys.startswith("SDCCH8"):
        return PLAN_SDCCH8
    if phys.startswith("CCCH"):
        return PLAN_SDCCH4
    return None


def load_timeslots(path):
    ts = {}
    try:
        cur = None
        n_bts = 0
        for line in open(path, encoding="utf-8", errors="ignore"):
            s = line.strip()
            if s.startswith("bts "):
                n_bts += 1
                if n_bts > 1:
                    break
                cur = None
            elif s.startswith("timeslot "):
                cur = int(s.split()[1])
            elif s.startswith("phys_chan_config") and cur is not None:
                ts[cur] = s.split(None, 1)[1].strip()
                cur = None
    except OSError:
        pass
    return ts or {0: "CCCH+SDCCH4", 1: "SDCCH8", 2: "TCH/F", 3: "TCH/F"}
```

3.314 /opt/GSM/osmo-operator/pont/pont.py

1986 octets, 37 lignes → 37 lignes

```
#!/usr/bin/env python3
# pont.py - le pont du montage GRGSM (couche 1 gr-gsm dans QEMU).
#
# [2026-09-23] Le pont est coupe en deux points d'entree sur le meme paquet :
#   pont.py           montage grgsm : QEMU recoit les blocs L2 en GSMTAP
#                     (4730/4731), le pont dechiffre A5 pour lui.
#   pont_dsp.py       montage DSP (c54x_exe --arm, --dsp-port 6702) : les
#                     bursts partent vers le BSP du C54x.
# Les deux gardent le chiffrement A5 dans le pont. Ce qui change d'un montage
# a l'autre passe par les DEFAUTS poses ici, avant l'import : le chemin grgsm
# garde exactement le comportement qu'il avait avant le decoupage, et ce qu'on
# change pour le DSP ne peut plus le casser. Une variable posee par
# l'operateur gagne toujours (setdefault).
import os
import sys

# Un burst montant reveille apres sa trame est JETE, comme avant le 2026-09-23
# (le montage DSP l'envoie, cf. trx.py Transmitter.run).
os.environ.setdefault("PONT_UL_RETARD_MAX", "0")

# [2026-10-03] INJECTEUR DE BRUIT (qosmo/tools/injecteur_bruit.py, cible grgsm).
# La L1 gr-gsm de QEMU ecoute GSMTAP sur 4730 en dur : pour intercaler
# l'injecteur, c'est le pont qui vise ailleurs. BRUIT_MODE=ber ou relais (les
# seuls modes qui aient un sens sur des blocs L2 deja decodes) fait viser
# BRUIT_PORT_GRGSM (14730), ou l'injecteur ecoute (qosmo/run_modules/38-bruit.sh,
```

```
# c54x_exe/run.sh). Sans BRUIT_MODE, rien ne change. PONT_GSMTAP_PORT pose par
# l'operateur (ou par c54x_exe/run.sh) gagne.
if os.environ.get("BRUIT_MODE", "") in ("ber", "relais"):
    os.environ.setdefault("PONT_GSMTAP_PORT", os.environ.get("BRUIT_PORT_GRGSM", "14730"))

sys.path[0] = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
from pont import main

if __name__ == "__main__":
    if any(a == "--dsp-port" or a.startswith("--dsp-port=") for a in sys.argv[1:]):
        sys.exit("pont.py est le pont du montage grgsm : pour le DSP, lancer pont/pont_dsp.py --dsp-port 6702")
    main()
```

3.315 /opt/GSM/osmo-operator/pont/pont_dsp.py

1563 octets, 31 lignes → 31 lignes

```
#!/usr/bin/env python3
# pont_dsp.py - le pont du montage DSP (c54x_exe --arm, bursts vers le BSP du
# C54x sur --dsp-port 6702). Le montage grgsm a son propre point d'entree,
# pont.py, qui garde ses defauts d'avant le decoupage du 2026-09-23 ; les deux
# partagent le paquet (trx, descendant, montant, chiffrement A5).
#
# [2026-09-23] Ce point d'entree lance pont.dsp.main, le vrai pont DSP : memes
# briques, mais l'etat dedie suit le DSP au lieu du decodage du pont (voir
# pont/dsp/__init__.py). pont/__init__.py et pont.py n'en dependent pas.
# Lance par c54x_exe/run.sh en MODE=dsp (PONT_PY par defaut), donc aussi par
# start-native.sh --dsp.
import os
import sys

# Un burst montant reveille apres sa trame part quand meme, jusqu'a 26 trames
# de retard : la BTS le range par son fn (cf. trx.py Transmitter.run).
os.environ.setdefault("PONT_UL_RETARD_MAX", "26")
# Le Kc est garde apres la liberation (pont/dsp/uplink.py) : c'est la
# retention qui le rend encore lisible une fois montant.c revenu en clair.
os.environ.setdefault("PONT_KC_RETENTION", "1")
# [2026-10-23] Le descendant aussi vers le BSP de l instance 4 (banc-rssi, udp 6762) :
# en BTS classique, l appli rssi n a pas d autre source (l air ne relaie que
# l anneau TX du Calypso de l instance 1, ici le mobile). Sans rssi lance, la copie
# part vers un port vide. PONT_DSP_COPIES= (vide) la coupe.
os.environ.setdefault("PONT_DSP_COPIES", "6762")

sys.path[0] = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
from pont.dsp import main

if __name__ == "__main__":
    main()
```

3.316 /opt/GSM/osmo-operator/pont/record.py

8351 octets, 219 lignes → 219 lignes

```
import logging
import os
import threading
import time

import numpy as np

from . import gsm
```

```
log = logging.getLogger("pont")

SAMPLES_PER_FRAME = 1250
SAMPLES_PER_SLOT_X4 = 625


class Direction:
    def __init__(self, name, path, fifo_path, max_bytes, min_free, osr, taps):
        self.name = name
        self.path = path
        self.fifo_path = fifo_path
        self.max_bytes = max_bytes
        self.min_free = min_free
        self.osr = osr
        self.taps = taps
        self.lock = threading.Lock()
        self.frames = {}
        self.fh = None
        self.bytes = 0
        self.off = False
        self.restart = False
        self.fifo_fd = None
        self.fifo_pending = b""
        self.fifo_retry = 0
        self.n_frames = 0
        self.n_bursts = 0
        self.n_fifo_drop = 0
        self.last_fn = None

    def feed(self, fn, tn, burst):
        if self.off:
            return
        with self.lock:
            # [2026-10-15] UN BURST EN RETARD VA A LA PROCHAINE TRAME ECRITE.
            # Les bursts montants de la ROM (PONT_UL_RAW, pont/dsp/uplink.py)
            # partent a SON fn, 12 a ~200 trames derriere l horloge du pont :
            # ranges sous une trame deja ecrite, ils n etaient jamais relus
            # (86 bursts enregistres sur 18 000 emis, spectre montant vide en
            # data). Pour le spectre, l instant d arrivee compte, pas le fn.
            last = self.last_fn
            if last is not None and 0 <= (last - fn) % 2715648 < 1357824:
                fn = (last + 1) % 2715648
            self.frames.setdefault(fn, {})[tn] = burst

    def open(self):
        if self.fh is not None:
            self.fh.close()
            self.fh = None
        try:
            os.makedirs(os.path.dirname(self.path) or ".", exist_ok=True)
            st = os.statvfs(os.path.dirname(self.path) or ".")
            if st.f_bavail * st.f_frsize < self.min_free:
               log.warning("enregistrement %s coupe : moins de %d Mo libres", self.name, self.min_free // 1048576)
                return False
            self.fh = open(self.path, "wb", buffering=1024 * 1024)
        except OSError as e:
            log.warning("enregistrement %s impossible (%s)", self.name, e)
            return False
        self.bytes = 0
        log.info("enregistrement %s -> %s (buffer roulant de %d Mo)", self.name, self.path, self.max_bytes //
↪   1048576)
        return True

    def modulate(self, burst):
        nrz = 2.0 * np.frombuffer(bytes(burst), dtype=np.uint8).astype(np.float64) - 1.0
        prev = np.concatenate(([nrz[0]], nrz[:-1]))
```

```
    up = np.zeros(len(nrz) * self.osr)
    up[::self.osr] = nrz * prev
    phase = np.cumsum(np.convolve(up, self.taps, mode="same")) * (np.pi / 2.0)
    return np.exp(1j * phase).astype(np.complex64)

def _fifo(self, octets):
    if not self.fifo_path:
        return
    if self.fifo_fd is None:
        if self.n_frames < self.fifo_retry:
            return
        self.fifo_retry = self.n_frames + 217
        try:
            if not os.path.exists(self.fifo_path):
                os.mkfifo(self.fifo_path, 0o644)
            self.fifo_fd = os.open(self.fifo_path, os.O_WRONLY | os.O_NONBLOCK)
        except OSError:
            return
    data = self.fifo_pending + octets
    maxq = 3 * len(octets)
    if len(data) > maxq:
        data = data[len(data) - maxq:]
        self.n_fifo_drop += 1
    try:
        n = os.write(self.fifo_fd, data)
        self.fifo_pending = data[n:]
    except BlockingIOError:
        self.fifo_pending = data
    except OSError:
        os.close(self.fifo_fd)
        self.fifo_fd = None
        self.fifo_pending = b""

def write_frame(self, fn, spf, step, blen, silence):
    # LE FIFO FFT NE DEPEND PAS DU DISQUE. [2026-09-04] fh est None quand
    # l'enregistrement fichier est coupe (moins de PONT_AIRREC_MINFREE_MB
    # libres : le cas de toute cle live, dont l'overlay tient en RAM). Le
    # FIFO n'etait alimente qu'ici, apres l'ecriture fichier : sans .cfile,
    # plus de spectre I/Q dans le dashboard ni dans le Conky, et le bouton
    # record du dashboard (qui lit CE fifo) restait muet. Le fichier reste
    # coupe faute de place - c'est voulu - mais les trames passent au FIFO.
    if self.off or (self.fh is None and not self.fifo_path):
        return
    with self.lock:
        slots = self.frames.pop(fn, None)
        self.last_fn = fn
        if len(self.frames) > 512:
            self.frames.clear()
    if slots:
        frame = np.zeros(spf, dtype=np.complex64)
        for tn, burst in slots.items():
            o = tn * step
            frame[o:o + blen] = self.modulate(burst)
            self.n_bursts += 1
    else:
        frame = silence
    octets = frame.tobytes()
    self._fifo(octets)
    self.n_frames += 1
    if self.fh is None:
        return
    try:
        self.fh.write(octets)
    except OSError as e:
        # Le fichier se coupe, pas le FIFO : voir l'en-tete de la methode.
        log.warning("enregistrement %s : ecriture echouee (%s), coupe", self.name, e)
```

```
            try:
                self.fh.close()
            except OSError:
                pass
            self.fh = None
            return
        self.bytes += len(octets)
        if self.restart or self.bytes >= self.max_bytes:
            self.restart = False
            self.fh.flush()
            self.fh.seek(0)
            self.fh.truncate(0)
            self.bytes = 0


class Recorder(threading.Thread):
    def __init__(self, cfg, clock):
        super().__init__(name="record", daemon=True)
        self.cfg = cfg
        self.clock = clock
        osr = cfg.record_osr
        n = 4 * osr
        t = np.arange(-n, n + 1) / float(osr)
        g = np.exp(-2.0 * (np.pi ** 2) * (0.3 ** 2) * t ** 2 / np.log(2.0))
        taps = g / g.sum()
        max_bytes = cfg.record_max_mb * 1024 * 1024
        min_free = cfg.record_min_free_mb * 1024 * 1024
        self.dl = Direction("descendant", cfg.record_dl_path, cfg.record_dl_fifo, max_bytes, min_free, osr, taps)
        self.ul = Direction("montant", cfg.record_ul_path, cfg.record_ul_fifo, max_bytes, min_free, osr, taps)
        self.ul.off = not cfg.record_ul

    def feed_dl(self, fn, tn, burst):
        self.dl.feed(fn, tn, burst)

    def feed_ul(self, fn, tn, burst):
        self.ul.feed(fn, tn, burst)

    def split(self, *_):
        self.dl.restart = True
        self.ul.restart = True

    def summary(self):
        return " | IQ dl=%dtr/%dbu ul=%dtr/%dbu fifo_drop=%d/%d" % (
            self.dl.n_frames, self.dl.n_bursts, self.ul.n_frames, self.ul.n_bursts,
            self.dl.n_fifo_drop, self.ul.n_fifo_drop)

    def run(self):
        osr = self.cfg.record_osr
        if osr % 4:
            log.warning("enregistrement refuse : OSR=%d doit etre multiple de 4", osr)
            return
        # Un sens dont le fichier ne s'ouvre pas (place, droits) garde son FIFO :
        # il ne s'eteint (off) que s'il n'a pas de FIFO a servir non plus.
        if not self.dl.open() and not self.dl.fifo_path:
            self.dl.off = True
        if not self.ul.off and not self.ul.open() and not self.ul.fifo_path:
            self.ul.off = True
        if self.dl.off and self.ul.off:
            return
        spf = SAMPLES_PER_FRAME * osr
        step = (SAMPLES_PER_SLOT_X4 * osr) // 4
        blen = 148 * osr
        silence = np.zeros(spf, dtype=np.complex64)
        nxt = (self.clock.fn() + 2) % gsm.HYPERFRAME
        while True:
            d = (self.clock.fn() - nxt) % gsm.HYPERFRAME
```

```
            if d > gsm.HYPERFRAME // 2 or d < 2:
                time.sleep(gsm.FRAME_DUR / 2.0)
                continue
            if d > 650:
                with self.dl.lock:
                    self.dl.frames.clear()
                with self.ul.lock:
                    self.ul.frames.clear()
                nxt = (self.clock.fn() + 2) % gsm.HYPERFRAME
                continue
            self.dl.write_frame(nxt, spf, step, blen, silence)
            self.ul.write_frame(nxt, spf, step, blen, silence)
            nxt = (nxt + 1) % gsm.HYPERFRAME
```

3.317 /opt/GSM/osmo-operator/pont/state.py

4810 octets, 155 lignes → 155 lignes

```
import logging
import os
import struct
import threading
import time

from . import gsm

log = logging.getLogger("pont")

DCCH_CFG = "/dev/shm/calypso_dcch_cfg"
TCH_CFG = "/dev/shm/calypso_tch_cfg"
DCCH_TTL = 0.1
DCCH_RELEASED = 0xFF


def ecrire_dcch(kind, ss, chan_nr):
    """[2026-10-22] calypso_dcch_cfg ecrit par le pont, au format du tap QEMU (calypso_dcch_tap.c) :
    seq(4) kind(1) ss(1) tn(1) chan_nr(1), 16 octets. Le firmware TI (Magnetite) n'a pas de tap :
    sans annonce, montant.c n'armait jamais un SDCCH hors TS0 (SDCCH/8 sur TS1 : IMM ASS ignoree
    par le banc, LU en Timeout). seq repart au-dessus de celui du fichier."""
    seq = 0
    try:
        with open(DCCH_CFG, "rb") as f:
            b = f.read(4)
            if len(b) == 4:
                seq = struct.unpack("<I", b)[0]
    except OSError:
        pass
    b = struct.pack("<IBBBB", (seq + 1) & 0xFFFFFFFF, kind & 0xFF, ss & 7, chan_nr & 7, chan_nr & 0xFF) + bytes(8)
    # EN PLACE, pas de os.replace : montant.c et pont/state.py gardent le fichier ouvert (pread sur leur fd),
    # un nouvel inode ne serait jamais relu.
    fd = os.open(DCCH_CFG, os.O_WRONLY | os.O_CREAT, 0o666)
    try:
        os.pwrite(fd, b, 0)
    finally:
        os.close(fd)


class Dedicated:
    def __init__(self):
        self._fd = None
        self._next = 0.0
        self._value = None
        self._seq = 0
```

```
    def read(self):
        now = time.monotonic()
        if now < self._next:
            return self._value
        self._next = now + DCCH_TTL
        try:
            if self._fd is None:
                self._fd = os.open(DCCH_CFG, os.O_RDONLY)
            b = os.pread(self._fd, 16, 0)
        except OSError:
            if self._fd is not None:
                os.close(self._fd)
                self._fd = None
            return self._value
        if len(b) < 8:
            return self._value
        seq = struct.unpack_from("<I", b, 0)[0]
        if not seq or seq == self._seq:
            return self._value
        self._seq = seq
        self._value = None if b[4] == DCCH_RELEASED else (b[4] & 1, b[5] & 7, b[6] & 7)
        return self._value

    def refresh(self):
        """Forcer la prochaine lecture : le fichier vient peut-etre de changer."""
        self._next = 0.0

    def plan(self):
        v = self.read()
        if v is None:
            return None
        kind, ss, tn = v
        return (gsm.PLAN_SDCCH8 if kind else gsm.PLAN_SDCCH4), ss, tn


class Tch:
    def __init__(self, stats):
        self.stats = stats
        self.lock = threading.Lock()
        self.tn = None
        self.tsc = 0
        self.open = False
        self.seq = 0
        self.on_close = []
        self.release_seen = None

    def arm(self, tn, tsc, arfcn):
        with self.lock:
            self.seq += 1
            self.tn = tn
            self.tsc = tsc
            self.open = False
            self._write_cfg(tn, tsc, arfcn, self.seq)
        log.info("ASSIGNMENT COMMAND : TCH TN=%d TSC=%d ARFCN=%d arme, en attente du mobile", tn, tsc, arfcn)

    def prove(self, source):
        with self.lock:
            if self.tn is None or self.open:
                return
            self.open = True
            tn = self.tn
        log.info("le mobile est sur le TCH TN=%d (%s) : montant TCH ouvert", tn, source)

    def release_requested(self, reason):
        with self.lock:
            if self.release_seen is not None:
```

```
                return
            self.release_seen = time.monotonic()
            tn = self.tn
        if tn is None:
            log.info("%s : en attente de la liberation du canal par le mobile", reason)
        else:
            log.info("%s : TCH TN=%d maintenu jusqu'a la liberation par le mobile", reason, tn)

    def release_overdue(self, delay):
        with self.lock:
            return self.release_seen is not None and time.monotonic() - self.release_seen > delay

    def close(self, reason):
        with self.lock:
            self.release_seen = None
            if self.tn is None:
                return
            tn = self.tn
            self.tn = None
            self.open = False
            self._write_cfg(0, 0, 0, 0)
        log.info("TCH TN=%d ferme (%s)", tn, reason)
        for cb in self.on_close:
            cb()

    def active_tn(self):
        return self.tn

    def is_open(self):
        return self.open and self.tn is not None

    @staticmethod
    def _write_cfg(tn, tsc, arfcn, seq):
        buf = bytearray(16)
        buf[4] = tn & 0xff
        buf[5] = tsc & 0xff
        buf[6:8] = int(arfcn).to_bytes(2, "little")
        buf[0:4] = int(seq).to_bytes(4, "little")
        fd = os.open(TCH_CFG, os.O_CREAT | os.O_WRONLY, 0o644)
        try:
            os.pwrite(fd, bytes(buf), 0)
        finally:
            os.close(fd)
```

3.318 /opt/GSM/osmo-operator/pont/stats.py

5565 octets, 111 lignes → 111 lignes

```
import logging
import threading
import time

log = logging.getLogger("pont")


class Stats:
    def __init__(self):
        self.dl_bursts = 0
        self.dl_blocks = 0
        self.dl_crc_fail = 0
        self.dl_dummy = 0
        self.dl_skipped = 0
        self.ul_sent = 0
        self.ul_late = 0
```

```
        self.rach = 0
        self.tch_dl = 0
        self.tch_facch_dl = 0
        self.tch_sacch_dl = 0
        self.tch_crc = 0
        self.tch_ul = 0
        self.tch_ul_bursts = 0
        self.tch_ul_dropped = 0
        self.facch_ul = 0
        self.sacch_ul = 0
        self.a5_dl = 0
        self.a5_ul = 0
        # [2026-10-03] blocs xCCH montants (SDCCH/SACCH du SDCCH) : bursts de la ROM DSP
        # (0x3f8a, pont/dsp/uplink.py) ou codes par l'hote (gsm.xcch_encode) ; rom_ko = bursts ROM
        # presents mais qui ne redecodent pas en ce L2.
        self.xcch_ul_rom = 0
        self.xcch_ul_hote = 0
        self.xcch_ul_rom_ko = 0
        # [2026-10-15] CE QUI PART VRAIMENT A L'ANTENNE, par chemin (PONT_UL_RAW=1).
        # Les compteurs rach/xcch_ul_hote/a5_ul ci-dessus comptent aussi le chemin
        # hote « fantome » qui tombe dans TransmitterPuits : la couverture les lisait
        # comme de l'emission hote (lignes MIXTE). air_* = bursts remis au TRX.
        self.air_rom_rach = 0
        self.air_rom_xcch = 0      # NB de la ROM hors TCH et hors PDCH (SDCCH/SACCH, chiffres par la ROM)
        self.air_rom_pdch = 0
        self.air_rom_tch = 0
        self.air_hote_tch = 0      # TCH code par l'hote (PONT_UL_RAW_TCH=0)
        self.air_puits = 0         # bursts hote jetes (jamais emis)
        # TCH montant par sous-canal (dsp/uplink.py classe_tch) : F FACCH, S SACCH/T, P parole.
        # rom = bursts ROM emis, rom_jete = bursts ROM du TCH ecartes (PONT_UL_RAW_TCH=0),
        # hote = bursts codes par l'hote et emis.
        self.tch_ul_rom = {"F": 0, "S": 0, "P": 0}
        self.tch_ul_rom_jete = {"F": 0, "S": 0, "P": 0}
        self.tch_ul_hote = {"F": 0, "S": 0, "P": 0}
        # A5 montant : rom = bursts NB marques chiffres par le coprocesseur (calypso_a5.c),
        # hote_tch = bursts TCH chiffres par le pont (cipher.py).
        self.a5_ul_rom = 0
        self.a5_ul_hote_tch = 0
        # [2026-10-15] phase B : bursts ROM de type inconnu (emis), blocs PDTCH vus sur la voie
        # SDCCH hote et jetes (B1), blocs xCCH sans bursts ROM a temps, non recodes (B2),
        # retard fn ROM -> horloge BTS des bursts du TCH (mesure pour B5, en trames).
        self.air_rom_inconnu = 0
        self.pdtch_ul_vu_hote = 0
        self.xcch_shadow_ko = 0
        self.tch_rom_retard_min = None
        self.tch_rom_retard_max = None
        self.sacch_t_rom_blocs = 0
        self.by_base = {}

    def block(self, tn, base, ok):
        st = self.by_base.setdefault((tn, base), [0, 0])
        st[0 if ok else 1] += 1


class Reporter(threading.Thread):
    def __init__(self, stats, clock, dedicated, record, period=5.0):
        super().__init__(name="stats", daemon=True)
        self.stats = stats
        self.clock = clock
        self.dedicated = dedicated
        self.record = record
        self.period = period

    def run(self):
        s = self.stats
        while True:
```

```
            time.sleep(self.period)
            log.info("STATS fn=%u | DL bursts=%d blocs=%d crc=%d dummy=%d hors_voie=%d | UL bursts=%d tard=%d
↪  rach=%d"
                     " | TCH dl=%d facch=%d sacch=%d crc=%d ul=%d bursts=%d perdus=%d | FACCH ul=%d SACCH ul=%d"
                     " | A5 dl=%d ul=%d | xCCH ul rom=%d hote=%d rom_ko=%d"
                     " | AIR rom rach=%d xcch=%d pdch=%d tch=%d | AIR hote tch=%d puits=%d"
                     " | TCH ul rom F=%d S=%d P=%d jete F=%d S=%d P=%d hote F=%d S=%d P=%d"
                     " | A5 ul rom=%d hote_tch=%d"
                     " | B inconnu=%d pdtch_hote_jete=%d xcch_ombre_ko=%d | TCH rom retard=%s..%s
                      ↪  sacch_t_blocs=%d%s",
                     self.clock.fn(), s.dl_bursts, s.dl_blocks, s.dl_crc_fail, s.dl_dummy, s.dl_skipped,
                     s.ul_sent, s.ul_late, s.rach, s.tch_dl, s.tch_facch_dl, s.tch_sacch_dl, s.tch_crc,
                     s.tch_ul, s.tch_ul_bursts, s.tch_ul_dropped, s.facch_ul, s.sacch_ul, s.a5_dl, s.a5_ul,
                     s.xcch_ul_rom, s.xcch_ul_hote, s.xcch_ul_rom_ko,
                     s.air_rom_rach, s.air_rom_xcch, s.air_rom_pdch, s.air_rom_tch, s.air_hote_tch, s.air_puits,
                     s.tch_ul_rom["F"], s.tch_ul_rom["S"], s.tch_ul_rom["P"],
                     s.tch_ul_rom_jete["F"], s.tch_ul_rom_jete["S"], s.tch_ul_rom_jete["P"],
                     s.tch_ul_hote["F"], s.tch_ul_hote["S"], s.tch_ul_hote["P"],
                     s.a5_ul_rom, s.a5_ul_hote_tch,
                     s.air_rom_inconnu, s.pdtch_ul_vu_hote, s.xcch_shadow_ko,
                     "-" if s.tch_rom_retard_min is None else s.tch_rom_retard_min,
                     "-" if s.tch_rom_retard_max is None else s.tch_rom_retard_max, s.sacch_t_rom_blocs,
                     self.record.summary() if self.record else "")
            ded = self.dedicated.plan()
            if ded:
                plan, ss, tn = ded
                log.info("canal dedie : %s SS=%d TS=%d (DL fn%%51=%d, UL fn%%51=%d)",
                         plan.name, ss, tn, plan.sdcch_dl[ss % plan.n_sub], plan.ul_base(ss))
            if s.by_base:
                log.info("blocs par TS/base51 (ok/echec) : %s",
                         " ".join("TS%d/%d:%d/%d" % (k[0], k[1], v[0], v[1]) for k, v in sorted(s.by_base.items())))
```

3.319 /opt/GSM/osmo-operator/pont/trx.py

27224 octets, 505 lignes → 505 lignes

```
import heapq
import logging
import os
import socket
import struct
import threading
import time

from . import gsm

log = logging.getLogger("pont")

# [2026-09-22] L'HORLOGE DU BANC SUIT LE DSP, PAS LE MUR.
#
# En montage DSP le C54x est emule : il coute ~5,8 ms par trame TDMA contre
# 4,615 ms de temps reel. La BTS, elle, tourne sur l'horloge que ce fichier lui
# envoie (IND CLOCK, run_clock). Avec une horloge murale elle emettait 1,25
# trame pour chacune que le DSP consommait : le BSP rejouait un anneau de plus
# en plus vieux (mesure du 2026-09-22 : pont a fn=99674 pendant que le BSP
# jouait la trame BTS 80410, soit 87 s de retard au bout de huit minutes),
# l'IMMEDIATE ASSIGNMENT arrivait apres T3126 et le canal dedie sortait des
# blocs incomplets (« Dropping frame with 110 bit errors »).
#
# c54x_exe publie la trame BTS que son BSP reclame (calypso_bsp.c,
# bsp_horloge_publier). On s'y asservit EN FREQUENCE, pas par recalage : la
# duree effective d'une trame (self.dur) suit la cadence mesuree du DSP, avec
# une correction proportionnelle qui tient l'avance de phase a AVANCE trames.
#
```

```
# Pourquoi pas un simple recalage de t0 : essaye d'abord, mesure immediate --
# « UL bursts=11 tard=232 ». Transmitter.run() jette tout burst dont la trame
# s'ecarte de plus de window_tol (1 trame) de l'horloge au moment de l'envoi ;
# une horloge qui avance par a-coups de quelques trames sort de cette fenetre a
# chaque fois. Plus un seul SABM n'arrivait a la BTS, donc plus de UA, donc
# « MDL-ERROR-IND cause 1 » et plus aucune mise a jour de localisation. Une
# horloge asservie doit rester CONTINUE : on ne change que sa vitesse, et on
# rebase t0 a chaque changement pour que fn() ne saute pas.
#
# PONT_HORLOGE=0 revient a l'horloge murale ; le fichier absent (montage sans
# DSP, couche 1 gr-gsm) laisse aussi l'horloge libre, sans rien a configurer.
HORLOGE = "/dev/shm/calypso_horloge"
#
# [2026-09-22] REGLAGE DE LA BOUCLE. Les premieres valeurs (periode 0,25 s,
# rattrapage en 400 trames) donnaient une boucle a ~0,07 Hz, SOUS la cadence a
# laquelle la vitesse du DSP elle-meme bouge. L'asservissement n'arrivait plus
# a suivre : la phase partait en cycle limite de +/- 100 trames, et a chaque
# demi-tour ou l'ecart passait negatif le pont se retrouvait DERRIERE la trame
# reclamee -- 12000 trames entierement sautees sur 120201 ticks, 10 %. Les
# trames perdues ainsi ne sont comptees NULLE PART (ni « tard », ni
# « manques » cote BSP) : c'est ce silence qui m'a fait accuser la
# demodulation du DSP. Mesurer plus souvent et rattraper plus vite remonte la
# boucle a ~1,3 Hz, au-dessus de la perturbation.
# Ne PAS « corriger » ca en augmentant PONT_HORLOGE_AVANCE : ca masque le
# cycle limite derriere une marge, sans le supprimer, et ca retarde tout le
# montant d'autant.
# [2026-09-22, 16:10] REGLABLES PAR L'ENVIRONNEMENT. J'ai change ces valeurs a
# l'aveugle une fois de trop : la premiere mesure apres coup a donne
# manques=2000 sur 10490 ticks (19 %), PIRE que les 10 % d'avant. On les sort
# donc pour pouvoir balayer sans recompiler et choisir sur mesure.
HORLOGE_PERIODE = float(os.environ.get("PONT_HORLOGE_PERIODE", "0.05"))
HORLOGE_KP = float(os.environ.get("PONT_HORLOGE_KP", "1.0"))
HORLOGE_PHASE_N = float(os.environ.get("PONT_HORLOGE_PHASE_N", "100.0"))
# [2026-10-19] PLAFOND DE RALENTISSEMENT. L'horloge asservie ne descendait pas
# sous le quart du temps reel (4 x FRAME_DUR). Avec la trace asm complete de
# l'ARM (start-native.sh --assembly-logs, ASSEMBLY_LOGS_FLAGS=in_asm,exec,
# nochain) le couple ARM+DSP en pas-a-pas tourne a 66 ms par trame, 14 fois le
# temps reel : le pont prenait +1159 puis +3897 trames d'avance, le BSP ne
# trouvait plus les bursts (« le burst a existe puis a disparu ») et mobile ne
# voyait aucune cellule (FBSB result=255, run du 2026-10-19 02:06).
# PONT_HORLOGE_MAX (defaut 4) borne le facteur ; c54x_exe/run.sh pose 20 quand
# ASSEMBLY_LOGS=1. Le plafond reste bas en temps normal : apres un a-coup du
# DSP, la cadence mesuree sur 50 ms explose et l'horloge ne doit pas s'affaler.
# Au-dela de 4, run_clock raccourcit la periode des IND CLOCK, voir plus bas.
HORLOGE_MAX = float(os.environ.get("PONT_HORLOGE_MAX", "4.0"))
# [2026-10-19] IND CLOCK en horloge lente, voir run_clock : avance maximale
# laissee a la BTS avant l'IND (= pause demandee, limite BTS 50), fn-advance de
# la BTS (osmotrx fn-advance), et garde-fou si la BTS ne livre plus rien
# (osmo-bts-trx coupe apres TRX_LOSS_FRAMES = 400 trames sans IND).
CLCK_PAUSE_MAX = int(os.environ.get("PONT_CLCK_PAUSE_MAX", "20"))
CLCK_BTS_FN_ADVANCE = int(os.environ.get("PONT_BTS_FN_ADVANCE", "2"))
CLCK_SANS_IND_MAX = int(os.environ.get("PONT_CLCK_SANS_IND_MAX", "300"))
# [2026-10-23] PLAFOND DE LA PAUSE DEMANDEE A LA BTS. osmo-bts-trx (scheduler_trx.c) met son timer en pause
# de N+1 trames quand l'IND le dit N trames en avance (N <= MAX_FN_SKEW = 50 ; au-dela il se recale, « GSM
# clock skew », sans dommage) ; si deux expirations s'ecartent alors de plus de 50 trames (230,75 ms), il
# SORT (« PC clock skew too high »). Une pause de 46 a 50 trames plus une trame de gigue suffit : run du
# 23/10 17:37 (Magnetite, DSP lent apres une liberation SDCCH), BTS0 sortie a 48 trames d'avance, reseau
# perdu. Si l'avance de la BTS tombe dans la zone dangereuse (> PLAFOND et <= MAX_FN_SKEW + marge), l'IND
# porte fn_bts - PLAFOND au lieu de fn_trx : la pause est bornee et l'avance restante se resorbe aux IND
# suivants (les bursts arrivent seulement plus tot, l'anneau du BSP les garde). 0 = comportement d'avant.
CLCK_PAUSE_PLAFOND = int(os.environ.get("PONT_CLCK_PAUSE_PLAFOND", "38"))
CLCK_SKEW_ZONE = int(os.environ.get("PONT_CLCK_SKEW_ZONE", "60"))   # au-dela, la BTS se recale seule


class Clock:
    def __init__(self):
```

```
    self.t0 = time.monotonic()
    self.dur = gsm.FRAME_DUR
    self._verrou = threading.Lock()
    self._asservie = os.environ.get("PONT_HORLOGE", "1") not in ("0", "", "no")
    self._avance = int(os.environ.get("PONT_HORLOGE_AVANCE", "12"))
    self._fd = None
    self._echeance = 0.0
    self._t_prec = None
    self._dsp_prec = None
    self._dit = False

def _lire_dsp(self):
    """Trame BTS que le DSP reclame, ou None s'il n'est pas encore cale."""
    try:
        if self._fd is None:
            self._fd = os.open(HORLOGE, os.O_RDONLY)
        b = os.pread(self._fd, 16, 0)
    except OSError:
        if self._fd is not None:
            os.close(self._fd)
            self._fd = None
        return None
    if len(b) < 16:
        return None
    seq, cale, fn_bts, _tick = struct.unpack("<IIII", b)
    return fn_bts % gsm.HYPERFRAME if (seq and cale) else None

def _reguler(self, now):
    """Appele sous verrou, au plus une fois par HORLOGE_PERIODE."""
    if now < self._echeance:
        return
    self._echeance = now + HORLOGE_PERIODE
    dsp = self._lire_dsp()
    if dsp is None:
        self._t_prec = self._dsp_prec = None
        return
    if not self._dit:
        self._dit = True
        log.info("horloge asservie au DSP (%s, avance visee %d trames, ralentissement max x%g)",
                 HORLOGE, self._avance, HORLOGE_MAX)
    t_prec, dsp_prec = self._t_prec, self._dsp_prec
    self._t_prec, self._dsp_prec = now, dsp
    if t_prec is None:
        return
    avancees = (dsp - dsp_prec) % gsm.HYPERFRAME
    if avancees <= 0 or avancees > gsm.HYPERFRAME // 2:
        return                                  # DSP arrete ou recale : on garde la vitesse
    dur_dsp = (now - t_prec) / avancees         # cadence mesuree du DSP
    # Ecart de phase, en trames, signe : > 0 = l'horloge du banc devance.
    f = (now - self.t0) / self.dur
    ecart = (int(f) - (dsp + self._avance)) % gsm.HYPERFRAME
    if ecart > gsm.HYPERFRAME // 2:
        ecart -= gsm.HYPERFRAME
    cible = dur_dsp * (1.0 + HORLOGE_KP * ecart / HORLOGE_PHASE_N)
    cible = min(max(cible, gsm.FRAME_DUR * 0.5), gsm.FRAME_DUR * HORLOGE_MAX)
    dur = 0.7 * self.dur + 0.3 * cible
    self.t0 = now - f * dur                     # rebase : fn() reste continue
    self.dur = dur

def fn(self):
    now = time.monotonic()
    if not self._asservie:
        return int((now - self.t0) / self.dur) % gsm.HYPERFRAME
    with self._verrou:
        self._reguler(now)
        return int((now - self.t0) / self.dur) % gsm.HYPERFRAME
```

```
    def time_of(self, fn):
        now = time.monotonic()
        if self._asservie:
            with self._verrou:
                self._reguler(now)
                t0, dur = self.t0, self.dur
        else:
            t0, dur = self.t0, self.dur
        cur = int((now - t0) / dur)
        delta = (fn - cur) % gsm.HYPERFRAME
        if delta > gsm.HYPERFRAME // 2:
            delta -= gsm.HYPERFRAME
        return t0 + (cur + delta) * dur

    def sleep_until_fn(self, fn):
        dt = self.time_of(fn) - time.monotonic()
        if dt > 0:
            time.sleep(dt)


def _udp(bind_host, port):
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    s.bind((bind_host, port))
    return s


class Trx:
    def __init__(self, cfg, clock, stats, cipher, record, dedicated=None, tch=None):
        self.cfg = cfg
        self.clock = clock
        self.stats = stats
        self.cipher = cipher
        self.record = record
        self.dedicated = dedicated
        self.tch = tch
        self.sk_clck = _udp(cfg.trx_bind, cfg.trx_base)
        self.sk_ctrl = _udp(cfg.trx_bind, cfg.trx_base + 1)
        self.sk_data = _udp(cfg.trx_bind, cfg.trx_base + 2)
        self.bts_data = None
        self.bts_clck = None
        self.fn_bts_vu = None          # fn du dernier burst descendant recu de la BTS (run_data)
        self._clck_t = 0.0             # dernier IND CLOCK : instant, fn BTS vu, avance demandee
        self._clck_vu = None
        self._clck_avance = 0
        self._clck_lent = False
        # [2026-10-23] COPIES DU DESCENDANT vers d autres BSP (PONT_DSP_COPIES, ports UDP separes par
        # des virgules) : un Calypso qui ecoute la cellule sans etre le mobile du pont, l appli rssi de
        # l instance 4 (banc-rssi, BSP udp 6762) en BTS classique. Socket a part : un port sans
        # ecouteur (rssi pas lance) ne doit rien changer au chemin du mobile.
        self.dsp_copies = [int(p) for p in os.environ.get("PONT_DSP_COPIES", "").replace(",", " ").split()
                           if p.isdigit() and int(p) != cfg.dsp_port]
        self.sk_copies = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) if self.dsp_copies else None
        self.copies_ul = os.environ.get("PONT_DSP_COPIES_UL", "1") != "0"

    def run_ctrl(self):
        while True:
            data, addr = self.sk_ctrl.recvfrom(1500)
            req = data.decode("latin1").strip("\0").strip()
            if not req.startswith("CMD"):
                continue
            parts = req.split()
            cmd = parts[1] if len(parts) > 1 else ""
            args = (" " + " ".join(parts[2:])) if len(parts) > 2 else ""
            self.sk_ctrl.sendto(("RSP %s 0%s" % (cmd, args)).encode() + b"\0", addr)
            if self.bts_clck is None:
```

```
            self.bts_clck = (addr[0], self.cfg.trx_base + 100)
            log.info("BTS %s : horloge armee vers le port %d", addr[0], self.bts_clck[1])
        if cmd not in ("SETPOWER", "NOHANDOVER"):
            log.info("CTRL %s%s", cmd, args)

def run_clock(self):
    # [2026-10-19] IND CLOCK QUAND L'HORLOGE EST LENTE : ATTENDRE LA REPRISE DE
    # LA BTS. osmo-bts-trx (scheduler_trx.c) avance ses trames a sa propre
    # cadence murale entre deux IND CLOCK. A chaque IND il mesure
    # elapsed_fn = fn_trx - fn_bts ; negatif, il met son timer en PAUSE de
    # -elapsed_fn trames (« We were N FN faster than TRX, compensating »), et
    # cette pause est REARMEE DEPUIS MAINTENANT par tout IND recu pendant
    # qu'elle court. Son timer tue la BTS si deux expirations sont separees
    # de plus de MAX_FN_SKEW = 50 trames (« PC clock skew too high »).
    # Run du 2026-10-19 02:20 (horloge a k=14, IND toutes les 28 trames) :
    # pause de 31 trames, IND suivant arrive pendant la pause, rearmee a 25,
    # elapsed_us=267403 > 230750, BTS arretee. Avec la periode fixe de 51
    # trames et k <= 4 la pause valait 38 trames et finissait 13 trames
    # avant l'IND suivant ; a k=14 elle vaudrait 47 trames, a 3 trames de
    # l'IND suivant : intenable.
    # Regle. Tant que la pause de la periode fixe, periode x (1 - 1/k),
    # tient sous CLCK_PAUSE_MAX (k <= ~1,9), rien ne change : un IND toutes
    # les 51 trames, la pause finit bien avant le suivant. Au-dela, un IND
    # part seulement quand (1) la pause demandee par le precedent est
    # ecoulee ET la BTS a recommence a livrer des bursts (fn vu par run_data
    # avance), et (2) la BTS a repris CLCK_PAUSE_MAX trames d'avance sur
    # l'horloge. Apres une pause N l'avance vaut encore N (1 - 1/k) : la BTS
    # ne court que quelques trames avant l'IND suivant, la pause demandee
    # reste ~CLCK_PAUSE_MAX quel que soit k et n'est jamais rearmee. Pas de
    # retour a la periode fixe dans ce regime : c'est elle qui laissait
    # l'avance monter a 40-100 trames (simulation, scratch sim_clck.py).
    # Les bursts descendants portent fn_bts + clock_advance (osmotrx
    # fn-advance, 2 par defaut : main.c).
    periode = self.cfg.clock_period
    pause_max = CLCK_PAUSE_MAX
    fn_adv = CLCK_BTS_FN_ADVANCE
    demi = gsm.HYPERFRAME // 2
    lent = False
    while True:
        now = time.monotonic()
        if self.bts_clck:
            fn_trx = self.clock.fn()
            k = self.clock.dur / gsm.FRAME_DUR
            vu = self.fn_bts_vu
            attente = (now - self._clck_t) / gsm.FRAME_DUR
            lent = k > 1.0 and vu is not None and periode * (1.0 - 1.0 / k) > pause_max
            if lent != self._clck_lent:
                self._clck_lent = lent
                if lent:
                  log.info("IND CLOCK : horloge lente (x%.1f), on attend la reprise de la BTS entre deux IND "
                           "(pause max %d trames)", k, pause_max)
                else:
                    log.info("IND CLOCK : horloge x%.1f, retour a la periode fixe de %d trames", k, periode)
            if not lent:
                envoyer = attente >= periode
                avance = 0
            else:
                avance = ((vu - fn_adv - fn_trx) + demi) % gsm.HYPERFRAME - demi
                # Reprise de la BTS : des bursts arrivent a nouveau. Un ou deux
                # peuvent etre en vol juste apres l'IND, d'ou la pause attendue
                # (ou 3 bursts, si la BTS s'est recalee sans pause : « GSM
                # clock skew », ou IND precedent envoye avant son premier burst).
                if self._clck_vu is None:
                    livre = 3
                else:
                    livre = ((vu - self._clck_vu) + demi) % gsm.HYPERFRAME - demi
```

```
                    reprise = livre >= 3 or (livre >= 1 and attente >= self._clck_avance)
                    envoyer = (reprise and avance >= pause_max) or attente >= CLCK_SANS_IND_MAX
                if envoyer:
                    fn_ind = fn_trx
                    if CLCK_PAUSE_PLAFOND > 0 and vu is not None:
                        av_bts = ((vu - fn_adv - fn_trx) + demi) % gsm.HYPERFRAME - demi
                        if CLCK_PAUSE_PLAFOND < av_bts <= CLCK_SKEW_ZONE:
                            fn_ind = (fn_trx + av_bts - CLCK_PAUSE_PLAFOND) % gsm.HYPERFRAME
                            self._clck_bornees = getattr(self, "_clck_bornees", 0) + 1
                            if self._clck_bornees <= 20 or self._clck_bornees % 100 == 0:
                                log.warning("IND CLOCK : BTS %d trames devant, pause bornee a %d (fn %u au lieu de
↪  %u, "
                                            "%d fois)", av_bts, CLCK_PAUSE_PLAFOND, fn_ind, fn_trx, self._clck_bornees)
                    self.sk_clck.sendto(("IND CLOCK %u" % fn_ind).encode() + b"\0", self.bts_clck)
                    self._clck_t = now
                    self._clck_vu = vu
                    self._clck_avance = max(avance, 0)
                    self.stats_clck(avance)
            # Horloge asservie (k > 1) : on scrute a chaque trame, sinon la premiere
            # mesure d'avance apres la mise en route arrive 51 trames trop tard
            # (avance 47 a k=14, pause > 50 : BTS arretee des le depart).
            # En regime lent, a la demi-trame : la pause d'equilibre vaut r (k - 1),
            # r etant le retard de detection de la reprise (simulation : r ~1,5
            # trame a la trame, soit 30 trames a k=20 ; ~1 a la demi-trame).
            if lent:
                time.sleep(gsm.FRAME_DUR / 2)
            else:
                time.sleep(gsm.FRAME_DUR if self.clock.dur > gsm.FRAME_DUR else gsm.FRAME_DUR * periode)

    def stats_clck(self, avance):
        """Compte les IND CLOCK envoyes horloge lente ; trace les pauses longues."""
        if not self._clck_lent:
            return
        self._clck_n = getattr(self, "_clck_n", 0) + 1
        if avance > CLCK_PAUSE_MAX + 10 and getattr(self, "_clck_longues", 0) < 20:
            self._clck_longues = getattr(self, "_clck_longues", 0) + 1
            log.warning("IND CLOCK : la BTS avait %d trames d'avance (pause demandee > %d, limite BTS 50)",
                        avance, CLCK_PAUSE_MAX)

    def run_data(self, on_burst):
        while True:
            data, addr = self.sk_data.recvfrom(2000)
            self.bts_data = addr
            if len(data) < 6 + 148:
                continue
            tn = data[0] & 0x07
            fn = struct.unpack_from(">L", data, 1)[0]
            self.fn_bts_vu = fn            # ou en est la BTS (run_clock, horloge lente)
            bits = gsm.normalize_bits(data[6:154])
            if self.cfg.dsp_port:
                # [2026-09-17] Le meme burst, tel que le BSP du DSP l'attend
                # (calypso_bsp.c bsp_trxd_readable) : 8 octets d'en-tete
                # [tn, fn BE32, attenuation, 0, 0] puis 148 bits 0/1. Le BSP les
                # convertit lui-meme en I/Q (table cos) et les depose en DARAM.
                #
                # [2026-09-22] LE DESCENDANT DU CANAL DEDIE EST DECHIFFRE ICI.
                # Asymetrie constatee : send_ul() CHIFFRE le montant (le pont
                # tient la place de la voie d'emission du DSP), mais le
                # descendant partait brut vers le DSP. Or il n'y a aucun A5
                # dans le modele Calypso -- `d_a5mode` n'existe que dans
                # l1-grgsm/, rien dans l1-dsp/ : personne ne dechiffre.
                # Mesure du 2026-09-22 (ENCRYPTION="a5 1") : la transaction
                # allait plus loin que jamais -- SABM/UA, IDENTITY, puis
                # AUTHENTICATION REQUEST/RESPONSE, tout en clair -- et la
                # tempete de « Dropping frame with 96 bit errors » commencait
                # a la ligne EXACTE du `CIPHERING MODE COMPLETE`, pour ne plus
```

```
            # s'arreter. En « a5 0 » la meme transaction va au bout.
            # A5 est symetrique : appliquer le flux descendant dechiffre.
            # SEULEMENT sur l'intervalle dedie : la BCCH et la CCCH ne sont
            # jamais chiffrees, les toucher detruirait le campement.
            bits_dsp = bits
            if self.dechiffre_dl and self.cipher.dl_active and self._burst_dedie(tn, fn):
                bits_dsp = self.cipher.apply(bits, fn, False)
            hdr = bytes([tn & 0x07]) + struct.pack(">L", fn) + bytes([data[5] if len(data) > 5 else 0, 0, 0])
            paquet = hdr + bytes(1 if b else 0 for b in bits_dsp)
            self.sk_data.sendto(paquet, ("127.0.0.1", self.cfg.dsp_port))
            for port in self.dsp_copies:
                try:
                    self.sk_copies.sendto(paquet, ("127.0.0.1", port))
                except OSError:
                    pass
        on_burst(tn, fn, bits)

# Dechiffrer le descendant avant de le passer au BSP : oui pour le shunt
# gr-gsm, non en montage DSP (TrxDsp) ou la ROM le fait elle-meme.
dechiffre_dl = True

def _burst_dedie(self, tn, fn):
    """Vrai si CE burst appartient au canal dedie du mobile.

    [2026-09-22] Le test portait sur le seul intervalle (`tn == tn_dedie`).
    En CCCH+SDCCH/4 le canal dedie vit sur TS0, celui qui porte AUSSI la
    FCCH, la SCH, la BCCH et la CCCH -- jamais chiffrees, par definition :
    le mobile doit pouvoir les lire avant d'avoir une cle. Des que A5
    s'activait, tout TS0 partait donc XORe vers le DSP et le campement se
    defaisait sous les pieds de la transaction en cours. En SDCCH/8 sur un
    intervalle a lui le defaut ne se voyait pas, d'ou son age.
    On refait ici le meme tri que Downlink._signalling : le bloc doit etre
    celui du sous-canal `ss` du mobile, SDCCH ou SACCH.
    Lecture mise en cache par Dedicated (DCCH_TTL), donc appelable a chaque
    burst."""
    if self.tch is not None and self.tch.is_open():
        # Sur un TCH assigne tout l'intervalle est a la transaction (trafic,
        # FACCH et SACCH) : pas de sous-canal a trier.
        if tn == self.tch.active_tn():
            return True
    if self.dedicated is None:
        return False
    ded = self.dedicated.plan()
    if ded is None:
        return False
    plan, ss, tn_ded = ded
    if tn != tn_ded:
        return False
    m51 = fn % 51
    base = next((b for b in plan.ded_bases if b <= m51 <= b + 3), None)
    if base is None:
        return False
    ss %= plan.n_sub
    if base in plan.sdcch_dl:
        return base == plan.sdcch_dl[ss]
    return (fn - (m51 - base)) % 102 == plan.sacch_dl102[ss]

def send_ul(self, tn, fn, burst, cipher):
    if self.bts_data is None:
        return
    if cipher:
        burst = self.cipher.apply(burst, fn, True)
    hdr = bytes([tn & 0x07]) + struct.pack(">L", fn) + bytes([self.cfg.rssi & 0xFF]) + struct.pack(">h", 0)
    self.sk_data.sendto(hdr + bytes(255 if b else 0 for b in burst), self.bts_data)
    self.stats.ul_sent += 1
    # [2026-10-23] le MONTANT aussi vers les copies (PONT_DSP_COPIES), marque 0xA5 a l octet 6 : le BSP
```

```
        # (calypso_bsp.c, CALYPSO_AIR_NIVEAUX) le range a part et ne le joue qu a une fenetre accordee sur la
        # frequence montante (rssi, vue UL). PONT_DSP_COPIES_UL=0 le coupe.
        if self.dsp_copies and self.copies_ul:
            paquet = bytes([tn & 0x07]) + struct.pack(">L", fn) + bytes([0, 0xA5, 0]) + bytes(1 if b else 0 for b
↪   in burst)
            for port in self.dsp_copies:
                try:
                    self.sk_copies.sendto(paquet, ("127.0.0.1", port))
                except OSError:
                    pass
        if self.record:
            self.record.feed_ul(fn, tn, burst)


class Transmitter(threading.Thread):
    def __init__(self, cfg, clock, trx, stats):
        super().__init__(name="tx", daemon=True)
        self.cfg = cfg
        self.clock = clock
        self.trx = trx
        self.stats = stats
        self.cond = threading.Condition()
        self.heap = []
        self.seq = 0

    def schedule(self, tn, fn_air, burst, cipher, essais=0):
        post = self.clock.time_of((fn_air - self.cfg.ul_fn_advance) % gsm.HYPERFRAME)
        with self.cond:
            self.seq += 1
            heapq.heappush(self.heap, (post, self.seq, tn, fn_air, burst, cipher, essais))
            self.cond.notify()

    def run(self):
        while True:
            with self.cond:
                while not self.heap:
                    self.cond.wait()
                post = self.heap[0][0]
                dt = post - time.monotonic()
                if dt > 0:
                    self.cond.wait(dt)
                    continue
                _, _, tn, fn_air, burst, cipher, essais = heapq.heappop(self.heap)
            cur = self.clock.fn()
            off = (fn_air - (cur + self.cfg.ul_fn_advance) + gsm.HYPERFRAME // 2) % gsm.HYPERFRAME -
↪   gsm.HYPERFRAME // 2
            # [2026-09-22] TROP TOT N'EST PAS TROP TARD.
            # `post` est calcule a la mise en file, a partir de la cadence de
            # l'horloge A CE MOMENT-LA. Depuis que celle-ci suit le DSP (voir
            # Clock plus haut), elle n'avance plus au rythme du mur : le DSP
            # marque le pas, l'horloge avec lui, et le reveil tombe avant que la
            # trame visee ne soit arrivee. Le burst etait alors compte « en
            # retard » et JETE, alors qu'il etait en avance.
            # Mesure du 2026-09-22 : « UL bursts=116 tard=18 », 13 % des bursts
            # montants perdus. Un bloc en demande quatre : ~40 % des blocs
            # montants n'arrivaient pas entiers a la BTS. C'est exactement ce
            # qu'on voyait en bout de chaine -- TMSI REALLOCATION COMPLETE et
            # CP-ACK jamais recus, SABM retransmis par T200.
            # On re-attend donc, avec un `post` recalcule sur l'horloge
            # courante ; seul un burst VRAIMENT en retard est perdu.
            if off > self.cfg.window_tol:
                if essais < self.cfg.window_essais:
                    self.schedule(tn, fn_air, burst, cipher, essais + 1)
                else:
                    self.stats.ul_late += 1
                continue
```

```
            if off < -self.cfg.window_tol:
                # [2026-09-23] EN RETARD N'EST PAS PERDU. osmo-bts-trx range un
                # burst montant par SON fn, compare au dernier fn traite du meme
                # canal logique (scheduler.c, trx_sched_route_burst_ind) ; l'heure
                # d'arrivee n'y entre pas -- un vrai transceiver livre toujours le
                # montant apres coup. Le jeter ici ne protegeait donc rien, et
                # coutait cher : 141 bursts emis pour 314 jetes (69 %) sur le run
                # de 11:04, chaque trame LAPDm retransmise 2 a 5 fois par T200,
                # le SMS MT a 12 s au lieu de 2. Le retard vient du reveil du
                # thread (GIL partage avec l'enregistrement I/Q) et des variations
                # de l'horloge asservie au DSP, pas du BTS.
                # On l'envoie tant qu'il reste dans la multitrame (la BTS refuse
                # au-dela de mf_period, « Too many contiguous TDMA frames »).
                # PONT_UL_RETARD_MAX=0 retablit l'ancien comportement (jeter).
                self.stats.ul_late += 1
                if -off > self.cfg.ul_retard_max:
                    continue
            self.trx.send_ul(tn, fn_air, burst, cipher)
```

3.320 /opt/GSM/osmo-operator/pont/uplink.py

17788 octets, 423 lignes → 423 lignes

```
import collections
import ctypes
import logging
import os
import struct
import threading
import time

from . import gsm

log = logging.getLogger("pont")

SB_RACH = "/dev/shm/calypso_rach"
SB_SDCCH_UL = "/dev/shm/calypso_sdcch_ul"
SB_FACCH_UL = "/dev/shm/calypso_tch_facch_ul"
SB_SACCH_UL = "/dev/shm/calypso_tch_sacch_ul"
SB_TCH_UL = "/dev/shm/calypso_tch_ul"
TCH_UL_SLOT = 64
TCH_UL_FR_OFS = 16
POLL = gsm.FRAME_DUR / 8
TX_BURSTS = 24
BPLEN = 116
RELEASE_FALLBACK = 20.0
# Combien de temps garder un bloc SDCCH montant en attendant de connaitre le
# canal dedie (cf. _poll_sdcch).
SDCCH_ATTENTE = 1.0


class Sideband:
    def __init__(self, path, size):
        self.path = path
        self.size = size
        self.fd = None
        self.seq = 0

    def read(self):
        try:
            if self.fd is None:
                self.fd = os.open(self.path, os.O_RDONLY)
            return os.pread(self.fd, self.size, 0)
        except OSError:
```

```
            if self.fd is not None:
                os.close(self.fd)
                self.fd = None
            return b""

    def new_record(self):
        b = self.read()
        if len(b) < self.size:
            return None
        seq = struct.unpack_from("<I", b, 0)[0]
        if not seq or seq == self.seq:
            return None
        self.seq = seq
        return b

    def skip_pending(self):
        b = self.read()
        if len(b) >= 4:
            self.seq = struct.unpack_from("<I", b, 0)[0]


class TchUplinkRing:
    def __init__(self):
        self.fd = None
        self.last = 0

    def drain(self):
        try:
            if self.fd is None:
                self.fd = os.open(SB_TCH_UL, os.O_RDONLY)
            hdr = os.pread(self.fd, 8, 0)
        except OSError:
            if self.fd is not None:
                os.close(self.fd)
                self.fd = None
            return [], 0
        if len(hdr) < 8:
            return [], 0
        w, n = struct.unpack("<II", hdr)
        if not w or not n:
            return [], 0
        lost = 0
        if self.last == 0:
            self.last = w
        elif (w - self.last) > n:
            lost = (w - self.last) - n
            self.last = w - n
        frames = []
        while self.last < w:
            self.last += 1
            sl = os.pread(self.fd, TCH_UL_SLOT, 8 + ((self.last - 1) % n) * TCH_UL_SLOT)
            if len(sl) >= TCH_UL_FR_OFS + gsm.FR_BYTES:
                frames.append(sl[TCH_UL_FR_OFS:TCH_UL_FR_OFS + gsm.FR_BYTES])
        return frames, lost


class Uplink(threading.Thread):
    def __init__(self, cfg, clock, stats, dedicated, tch, transmitter, feeder, cipher):
        super().__init__(name="uplink", daemon=True)
        self.cfg = cfg
        self.clock = clock
        self.stats = stats
        self.dedicated = dedicated
        self.tch = tch
        self.cipher = cipher
        self.ded_active = False
```

```
    self.tx = transmitter
    self.feed = feeder
    self.sb_rach = Sideband(SB_RACH, 12)
    self.sb_sdcch = Sideband(SB_SDCCH_UL, 39)
    self.sb_facch = Sideband(SB_FACCH_UL, 39)
    self.sb_sacch = Sideband(SB_SACCH_UL, 39)
    self.ring = TchUplinkRing()
    self.q_lock = threading.Lock()
    self.q_facch = collections.deque(maxlen=8)
    self.q_voice = collections.deque(maxlen=4)
    self.sacch_l2 = None
    self.tch_epoch = -1
    self.sdcch_attente = None
    for sb in (self.sb_rach, self.sb_sdcch, self.sb_facch, self.sb_sacch):
        sb.skip_pending()

def _next_fn(self, min_advance, accept):
    start = self.clock.fn() + min_advance
    for k in range(104):
        fn = (start + k) % gsm.HYPERFRAME
        if accept(fn):
            return fn
    return start % gsm.HYPERFRAME

def _queue_facch(self, l2, ouverture=False):
    # [2026-09-23] UNE UI SAPI 0 SUR LE TCH : UN BLOC QUI N'EST PAS UNE FACCH.
    # Le BSC signale « SAPI=0 UNIT DATA INDICATION: unimplemented » (RLL
    # 0x0b) sur lchan TCH_F, a l'ouverture du TCH (19:12:08, 20:22:54,
    # 20:32:29, 20:35:28, 21:00:40 -- avant meme le SABM du mobile). Ce
    # n'est pas le bourrage 01 03 01 2b : lapd_core.c:1127 jette une UI de
    # longueur 0. C'est lapdm.c:995 : un premier octet au bit EA a 0 n'est
    # pas une adresse, la trame est prise pour un en-tete court (Bter) et
    # remonte TELLE QUELLE en UNIT DATA IND. C'est la forme d'un bloc SACCH
    # montant : en-tete L1 d'abord (puissance, TA) -- 08 00 01 03 49 ... a
    # la puissance 8, alors qu'a la puissance 7 (07 ...) il passe pour du
    # SAPI 1 et disparait sans bruit, d'ou l'intermittence.
    # Une FACCH est toujours au format B et SAPI 0 : adresse 0x01 ou 0x03.
    # Tout autre bloc est ecarte ici, octets journalises pour en trouver
    # la source. PONT_FACCH_TOUT=1 retablit l'envoi sans tri.
    if l2[0] not in (0x01, 0x03) and os.environ.get("PONT_FACCH_TOUT", "0") != "1":
        log.info("FACCH montante ECARTEE (pas une adresse SAPI 0) : %s",
                 bytes(l2[:gsm.MACBLOCK_LEN]).hex(" "))
        return
    # Les trames non-I (SABM, RR, REJ, DISC...) sont rares : on les garde
    # au journal, comme la premiere FACCH de chaque TCH.
    if (l2[1] & 0x01) or ouverture:
        log.info("FACCH montante %s : %s", "non-I" if l2[1] & 0x01 else "I (ouverture)",
                 bytes(l2[:gsm.MACBLOCK_LEN]).hex(" "))
    with self.q_lock:
        self.q_facch.append(bytes(l2[:gsm.MACBLOCK_LEN]))
    self.stats.facch_ul += 1
    self.feed.l2(self.clock.fn(), gsm.GSMTAP_TCH_F, l2[:gsm.MACBLOCK_LEN], self.tch.active_tn() or 0, True)

def _poll_rach(self):
    b = self.sb_rach.new_record()
    if b is None:
        return
    _, ra, bsic, fn = struct.unpack_from("<IBBxxI", b, 0)
    burst = gsm.rach_burst(ra, bsic)
    self.tx.schedule(0, self._next_fn(4, lambda f: f % 51 in gsm.RACH_SLOTS_51), burst, False)
    self.stats.rach += 1

def _route_sdcch(self, l2):
    """Vrai si le bloc SDCCH montant part en FACCH (consomme ici).

    [2026-09-23] Extrait tel quel de _poll_sdcch pour que le pont DSP
```

```
    (pont/dsp/uplink.py) le surcharge ; le montage grgsm garde ce code."""
    tn_tch = self.tch.active_tn()
    if tn_tch is not None and not self.tch.is_open():
        if gsm.rr_message_type(l2)[1] == gsm.RR_ASSIGNMENT_COMPLETE:
            self.tch.prove("ASSIGNMENT COMPLETE")
            self._queue_facch(l2)
            return True
    elif self.tch.is_open():
        self._queue_facch(l2)
        return True
    return False

def _poll_sdcch(self):
    b = self.sb_sdcch.new_record()
    if b is not None:
        l2 = b[16:39]
        if self._route_sdcch(l2):
            return
        # [2026-09-21] Le bloc ATTEND que le canal soit connu, il n'est
        # plus jete. La couche 1 publie le SABM au moment meme ou elle
        # l'emet, et le canal dedie n'est lu ici qu'au plus toutes les
        # DCCH_TTL secondes : le premier bloc d'une connexion tombait
        # regulierement dans cette fenetre et disparaissait, le BTS
        # attendait son SABM jusqu'au Timeout. On relit le canal tout de
        # suite et on garde le bloc un instant s'il n'est pas encore arme.
        self.sdcch_attente = (l2, time.monotonic())
        self.dedicated.refresh()
    if self.sdcch_attente is None:
        return
    l2, depuis = self.sdcch_attente
    ded = self.dedicated.plan()
    if ded is None:
        if time.monotonic() - depuis > SDCCH_ATTENTE:
            log.info("SDCCH montant : aucun canal dedie apres %.1f s, bloc abandonne",
                     SDCCH_ATTENTE)
            self.sdcch_attente = None
        return
    plan, ss, tn = ded
    bursts = self._bursts_xcch(l2)
    if bursts is None:
        return          # le montage DSP attend encore les bursts de la ROM (pont/dsp/uplink.py)
    self.sdcch_attente = None

    # [2026-09-21] SDCCH ou SACCH ? La couche 1 ne peut pas le dire : le
    # firmware pose les deux dans le MEME tampon a_cu avec le MEME mot de
    # tache (prim_tx_nb.c:118, dsp_load_tx_task(DUL_DSP_TASK) dans les deux
    # cas) ; seul son drapeau interne MF_F_SACCH les separe, et il ne sort
    # jamais dans l'API RAM. C'est donc ici, sur la FORME du bloc, que ca se
    # tranche : un bloc SDCCH commence par l'adresse LAPDm (0x01 ou 0x03),
    # un bloc SACCH par ses deux octets d'en-tete L1 (puissance ordonnee,
    # TA), l'adresse ne venant qu'en position 2.
    #
    # Releve du banc, les deux blocs qui alternaient sur la voie SDCCH :
    #   01 3f 49 05 08 70 ...  -> SABM + LOCATION UPDATING REQUEST
    #   07 00 01 03 49 06 15   -> en-tete L1 (pwr 7, TA 0) + MEASUREMENT REPORT
    # Le second partait sur les trames SDCCH montantes, et la SACCH
    # montante restait vide : le BTS ne voyait plus rien sur sa liaison
    # lente et lachait le canal quelques secondes apres l'etablissement.
    #
    # [2026-09-23] Et le SAPI 3 ? Son adresse LAPDm est 0x0d / 0x0f, et une
    # trame SAPI 3 sans information (UA, SABM, RR, DISC) porte en position 2
    # un indicateur de longueur L=0, soit 0x01 -- exactement ce que le test
    # ci-dessus prenait pour l'adresse SAPI 0 d'un bloc SACCH. Releve du
    # banc, MT SMS vers le mobile du pont :
    #   0f 73 01 2b 2b ...     -> UA SAPI 3 (reponse au SABM du BTS)
    # parti sur la SACCH montante : le BTS le decodait en « Unknown (SS) »,
```

```
    # n'etablissait jamais le SAPI 3, et le CP-DATA du SMS ne partait pas
    # (MSC : WAIT_CP_ACK puis abandon au bout de 7 s, a chaque essai).
    # Octet 0 = 0x0d / 0x0f est ambigu (puissance 13 / 15 d'un en-tete
    # SACCH) : on ne tranche SACCH que si la lecture SACCH tient debout --
    # TA <= 63 (6 bits) et un vrai champ de controle LAPDm en position 3,
    # pas le bourrage 0x2b qui suit une trame SAPI 3 vide.
    sacch = l2[0] not in (0x01, 0x03) and l2[2] in (0x01, 0x03)
    if sacch and l2[0] in (0x0d, 0x0f):
        sacch = l2[1] <= 63 and l2[3] != 0x2b
    if sacch:
        base = (plan.sacch_dl102[ss % plan.n_sub] + 15) % 102
        fn0 = self._next_fn(4, lambda f: f % 102 == base)
        self.feed.l2(self.clock.fn(), gsm.GSMTAP_SACCH, l2, tn, True)
        self.stats.sacch_ul += 1
    else:
        base = plan.ul_base(ss)
        fn0 = self._next_fn(4, lambda f: f % 51 == base)
        self.feed.l2(self.clock.fn(), gsm.GSMTAP_SDCCH4, l2, tn, True)
    for j, burst in enumerate(bursts):
        self.tx.schedule(tn, (fn0 + j) % gsm.HYPERFRAME, burst, True)

def _bursts_xcch(self, l2):
    """Les 4 bursts d'un bloc SDCCH/SACCH montant. Ici codes par l'hote ; le pont DSP
    (pont/dsp/uplink.py) prend ceux de la ROM. None = pas encore prets, le bloc reste
    en attente et _poll_sdcch le represente au tour suivant."""
    self.stats.xcch_ul_hote += 1
    return gsm.xcch_encode(l2)

def _poll_facch(self):
    tn = self.tch.active_tn()
    if tn is None:
        self.tch_epoch = -1
        return
    if self.tch_epoch != self.tch.seq:
        # [2026-09-22] NE PLUS JETER LA PREMIERE FACCH APRES L'ARMEMENT.
        #
        # `skip_pending()` avance le curseur sur TOUT ce qui est deja ecrit
        # dans la bande laterale, et le `return` sortait sans rien traiter.
        # Or `tch.seq` est incremente par `tch.arm()`, appele quand le pont
        # decode l'ASSIGNMENT COMMAND descendante -- et l'ASSIGNMENT
        # COMPLETE est justement LA PREMIERE chose que le mobile emet sur le
        # nouveau TCH. Elle tombait donc exactement dans cette fenetre et
        # etait marquee « deja vue ».
        #
        # Mesure du 2026-09-22, appel vers 600 : le mobile emet bien
        # « ASSIGNMENT COMPLETE (cause #0) » (gsm48_rr.c:4720, 18:55:47),
        # le pont journalise « FACCH montante » SANS le suffixe
        # « , ASSIGNMENT COMPLETE » -- donc ce n'etait pas elle -- et le BSC
        # conclut « Assignment failed in state WAIT_RR_ASS_COMPLETE, cause
        # EQUIPMENT FAILURE: Timeout » (assignment_fsm.c:1057).
        #
        # On garde le saut pour la SACCH (un rapport de mesure perime ne
        # sert a rien) mais on TRAITE la FACCH en attente. Le risque
        # residuel -- rejouer une FACCH de la session precedente -- est
        # borne a un bloc et LAPDm le rejettera, la ou perdre l'ASSIGNMENT
        # COMPLETE fait echouer l'appel a tous les coups.
        # PONT_FACCH_SKIP=1 retablit l'ancien comportement.
        self.tch_epoch = self.tch.seq
        self.sb_sacch.skip_pending()
        if os.environ.get("PONT_FACCH_SKIP", "0") == "1":
            self.sb_facch.skip_pending()
            return
    b = self.sb_facch.new_record()
    if b is None:
        return
    l2 = b[16:39]
```

```
        # Un bloc qui n'est pas une FACCH (voir _queue_facch) n'ouvre pas le TCH.
        if l2[0] not in (0x01, 0x03) and os.environ.get("PONT_FACCH_TOUT", "0") != "1":
            self._queue_facch(l2)
            return
        ouverture = not self.tch.is_open()
        self.tch.prove("FACCH montante%s" % (", ASSIGNMENT COMPLETE" if gsm.rr_message_type(l2)[1] ==
        ↪  gsm.RR_ASSIGNMENT_COMPLETE else ""))
        self._queue_facch(l2, ouverture)

    def _poll_sacch(self):
        b = self.sb_sacch.new_record()
        if b is None:
            return
        self.sacch_l2 = bytes(b[16:39])
        self.stats.sacch_ul += 1

    def _poll_release(self):
        active = self.dedicated.plan() is not None
        if self.ded_active and not active:
            reason = "canal libere par le mobile"
        elif self.tch.release_overdue(RELEASE_FALLBACK):
            reason = "CHANNEL RELEASE sans liberation du mobile depuis %ds" % RELEASE_FALLBACK
        else:
            self.ded_active = active
            return
        self.ded_active = active
        self.tch.close(reason)
        self.cipher.release(reason)

    def _poll_voice(self):
        if not self.tch.is_open():
            return
        frames, lost = self.ring.drain()
        self.stats.tch_ul_dropped += lost
        for fr in frames:
            with self.q_lock:
                if len(self.q_voice) == self.q_voice.maxlen:
                    self.stats.tch_ul_dropped += 1
                self.q_voice.append(bytes(fr))
            self.stats.tch_ul += 1

    def run(self):
        while True:
            self._poll_release()
            self._poll_rach()
            self._poll_sdcch()
            self._poll_facch()
            self._poll_sacch()
            self._poll_voice()
            time.sleep(POLL)


class TchScheduler(threading.Thread):
    def __init__(self, cfg, clock, stats, tch, uplink, transmitter):
        super().__init__(name="tch-tx", daemon=True)
        self.cfg = cfg
        self.clock = clock
        self.stats = stats
        self.tch = tch
        self.uplink = uplink
        self.tx = transmitter
        self.buf = (gsm.ubit * (TX_BURSTS * BPLEN))()
        self.mask = 0
        self.sent_sacch_block = None
        tch.on_close.append(self._reset)
```

```
    def _reset(self):
        ctypes.memset(self.buf, 0, TX_BURSTS * BPLEN)
        self.mask = 0
        self.sent_sacch_block = None
        with self.uplink.q_lock:
            self.uplink.q_facch.clear()
            self.uplink.q_voice.clear()

    def _load_block(self):
        ctypes.memmove(self.buf, ctypes.byref(self.buf, 4 * BPLEN), 20 * BPLEN)
        ctypes.memset(ctypes.byref(self.buf, 20 * BPLEN), 0, 4 * BPLEN)
        self.mask = (self.mask << 4) & 0xFFFFFFFF
        with self.uplink.q_lock:
            facch = self.uplink.q_facch.popleft() if self.uplink.q_facch else None
            voice = self.uplink.q_voice.popleft() if self.uplink.q_voice else None
        gsm.tch_fr_encode_into(self.buf, facch if facch is not None else voice)

    def _sacch(self, tn, fn_air):
        if not gsm.sacch_tf_block_start(fn_air, tn):
            return
        block = fn_air - (fn_air % 104)
        if self.uplink.sacch_l2 is None or self.sent_sacch_block == block:
            return
        self.sent_sacch_block = block
        for j, burst in enumerate(gsm.xcch_encode(self.uplink.sacch_l2)):
            self.tx.schedule(tn, (fn_air + 26 * j) % gsm.HYPERFRAME, burst, True)

    def run(self):
        fn_air = None
        while True:
            tn = self.tch.active_tn()
            if tn is None or not self.tch.is_open():
                fn_air = None
                time.sleep(0.05)
                continue
            if fn_air is None:
                fn_air = (self.clock.fn() + self.cfg.ul_fn_advance + 1) % gsm.HYPERFRAME
            else:
                fn_air = (fn_air + 1) % gsm.HYPERFRAME
            self.clock.sleep_until_fn((fn_air - self.cfg.ul_fn_advance) % gsm.HYPERFRAME)
            self._sacch(tn, fn_air)
            if not gsm.is_tch_carrier(fn_air):
                continue
            bid = gsm.tch_burst_index(fn_air) % 4
            if bid == 0:
                self._load_block()
            elif not (self.mask & 0x01):
                continue
            base = bid * BPLEN
            burst = gsm.burst_from_coded(bytes(self.buf)[base:base + BPLEN])
            self.mask |= 1 << bid
            self.tx.schedule(tn, fn_air, burst, True)
            self.stats.tch_ul_bursts += 1
```

3.321 /opt/GSM/osmo-operator/pont/voix_hote.py

4887 octets, 107 lignes → 107 lignes

```
"""voix_hote.py - [2026-10-22] la voix du C139 Magnetite codee par l'hote (PONT_VOIX_HOTE=1).

Le vocodeur de la mask-ROM ne tient pas encore : le C54x emule tourne a ~9,5 ms par trame
quand le codec FR tourne (budget 4,62 ms), la ROM decode le TCH descendant en BFI (parole,
FACCH, SACCH) et ne remplit jamais a_du. Le pont, lui, decode deja chaque trame FR que la BTS
envoie (Downlink._tch) et code deja le TCH montant (TchScheduler). Il ne manquait que le PCM :
```

```
    DL : trame FR de la BTS --libgsm--> anneau /dev/shm/pont_voix_dl --pacat--> gsm_audio
    UL : gsm_mic.monitor --parec--> anneau /dev/shm/pont_voix_ul --libgsm--> uplink.q_voice

Le DSP ne fait plus que le canal : CALYPSO_VSP=0 (pas de port voix, pas de codec ROM). Active
par --gapk avec --ms-magnetite (startup_scripts/14-demarrage.sh). Les trames FR sont au format
de libosmocoding net_order=1 (TS 101 318, quartet 0xD en tete), celui de libgsm.
"""
import ctypes
import logging
import os
import struct
import threading
import time

from . import vsp_audio

log = logging.getLogger("pont")

SHM_UL = os.environ.get("PONT_VOIX_SHM_UL", "/dev/shm/pont_voix_ul")
SHM_DL = os.environ.get("PONT_VOIX_SHM_DL", "/dev/shm/pont_voix_dl")
N = vsp_audio.N
ECH = 160                                   # 20 ms a 8 kHz, une trame FR

_gsm = ctypes.CDLL("libgsm.so.1")
_gsm.gsm_create.restype = ctypes.c_void_p
_gsm.gsm_decode.argtypes = [ctypes.c_void_p, ctypes.c_char_p, ctypes.POINTER(ctypes.c_short)]
_gsm.gsm_encode.argtypes = [ctypes.c_void_p, ctypes.POINTER(ctypes.c_short), ctypes.c_char_p]


class VoixHote:
    """Remplace l'anneau TCH descendant du pont DSP (DownlinkDsp.ring) : publish() recoit chaque
    trame FR decodee de la BTS. Un fil code le micro et alimente le TCH montant."""

    def __init__(self, uplink, tch, stats):
        self.uplink = uplink
        self.tch = tch
        self.stats = stats
        self.dec = _gsm.gsm_create()
        self.enc = _gsm.gsm_create()
        self.ul = vsp_audio.anneau(SHM_UL)
        self.dl = vsp_audio.anneau(SHM_DL)
        self.pcm = (ctypes.c_short * ECH)()
        self.n_dl = self.n_dl_ko = self.n_ul = 0
        # une seule source pour la parole montante : plus celle de montant.c (a_du, /dev/shm/calypso_tch_ul)
        uplink._poll_voice = lambda: None

    # -- descendant ------------------------------------------------------------------------
    def publish(self, fr33, fn):
        if len(fr33) != 33 or _gsm.gsm_decode(self.dec, bytes(fr33), self.pcm) != 0:
            self.n_dl_ko += 1
            return
        w = struct.unpack_from("<I", self.dl, 0)[0]
        for i in range(ECH):
            struct.pack_into("<h", self.dl, 8 + 2 * ((w + i) % N), self.pcm[i])
        struct.pack_into("<I", self.dl, 0, (w + ECH) & 0xFFFFFFFF)   # compteur en dernier
        self.n_dl += 1

    # -- montant ---------------------------------------------------------------------------
    def _codeur(self):
        lu = struct.unpack_from("<I", self.ul, 0)[0]
        pcm = (ctypes.c_short * ECH)()
        out = ctypes.create_string_buffer(33)
        while True:
            w = struct.unpack_from("<I", self.ul, 0)[0]
            dispo = (w - lu) & 0xFFFFFFFF
```

```
            if dispo > N // 2:                      # retard (pont arrete, parec en rafale) : rattraper
                lu = (w - ECH) & 0xFFFFFFFF
                dispo = ECH
            if dispo < ECH:
                time.sleep(0.005)
                continue
            for i in range(ECH):
                pcm[i] = struct.unpack_from("<h", self.ul, 8 + 2 * ((lu + i) % N))[0]
            lu = (lu + ECH) & 0xFFFFFFFF
            if not self.tch.is_open():
                continue
            _gsm.gsm_encode(self.enc, pcm, out)
            with self.uplink.q_lock:
                if len(self.uplink.q_voice) == self.uplink.q_voice.maxlen:
                    self.stats.tch_ul_dropped += 1
                self.uplink.q_voice.append(out.raw)
            self.stats.tch_ul += 1
            self.n_ul += 1

    def _bilan(self):
        while True:
            time.sleep(30)
            log.info("voix hote : DL %d trames jouees (%d illisibles), UL %d trames codees en 30 s",
                     self.n_dl, self.n_dl_ko, self.n_ul)
            self.n_dl = self.n_dl_ko = self.n_ul = 0

    def start(self):
        vsp_audio.micro_reel()
        for nom, cible, args in (("voix-micro", vsp_audio.micro, (self.ul,)),
                                 ("voix-hp", vsp_audio.haut_parleur, (self.dl,)),
                                 ("voix-codeur", self._codeur, ()),
                                 ("voix-bilan", self._bilan, ())):
            threading.Thread(name=nom, target=cible, args=args, daemon=True).start()
        log.info("voix hote : FR de la BTS -> %s -> %s ; %s -> %s -> TCH montant (libgsm)",
                 SHM_DL, vsp_audio.SINK, vsp_audio.SOURCE, SHM_UL)
```

3.322 /opt/GSM/osmo-operator/pont/vsp_audio.py

7623 octets, 178 lignes → 178 lignes

```
"""vsp_audio.py - le port voix (VSP) du DSP Calypso relie a PulseAudio.

[2026-10-15] c54x_exe modelise le port serie SP0 de la ROM (qosmo
hw/arm/calypso/l1-dsp/calypso_vsp.c, CALYPSO_VSP=1) : a chaque echantillon il
pose DRR depuis l'anneau /dev/shm/calypso_pcm_ul et releve DXR dans
/dev/shm/calypso_pcm_dl, a 8 kHz exacts par rapport au fn de la BTS. Ce module
fait le reste, hors de la boucle temps reel du DSP :

    gsm_mic.monitor --parec--> calypso_pcm_ul      (le micro du mobile)
    calypso_pcm_dl  --pacat--> gsm_audio            (son haut-parleur)

Ce sont les points ou GAPK branchait le mobile (alsa gsm_in / gsm_out) et ceux
que l'etage « voix » de banc-max injecte et ecoute : le test reste le meme juge.

Anneaux : en-tete uint32 ecrit (compteur d'echantillons ecrits), uint32 taille,
puis taille x int16. Un seul ecrivain par anneau. Lance par c54x_exe/run.sh
(etape 5b) quand CALYPSO_VSP=1 :  python3 -m pont.vsp_audio
"""
import fcntl
import mmap
import os
import struct
import subprocess
import sys
```

```
import termios
import threading
import time

SHM_UL = "/dev/shm/calypso_pcm_ul"
SHM_DL = "/dev/shm/calypso_pcm_dl"
N = 16000                       # = VSP_SHM_N (calypso_vsp.h)
TAILLE = 8 + 2 * N
TAUX = 8000
SOURCE = os.environ.get("VSP_SOURCE", "gsm_mic.monitor")
SINK = os.environ.get("VSP_SINK", "gsm_audio")
GIGUE = int(os.environ.get("VSP_GIGUE_MS", "60")) * TAUX // 1000
DL_CIBLE = int(os.environ.get("VSP_DL_CIBLE_MS", "150")) * TAUX // 1000   # tuyau vers pacat


def anneau(chemin):
    fd = os.open(chemin, os.O_RDWR | os.O_CREAT, 0o666)
    os.ftruncate(fd, TAILLE)
    m = mmap.mmap(fd, TAILLE)
    os.close(fd)
    struct.pack_into("<I", m, 4, N)
    return m


def journal(*a):
    print("[vsp-audio]", *a, flush=True)


def micro(ul):
    """parec -> anneau UL (ecrivain unique)."""
    cmd = ["parec", f"--device={SOURCE}", "--format=s16le", f"--rate={TAUX}", "--channels=1",
           "--raw", "--latency-msec=20"]
    p = subprocess.Popen(cmd, stdout=subprocess.PIPE)
    journal("micro :", " ".join(cmd))
    w = struct.unpack_from("<I", ul, 0)[0]
    while True:
        bloc = p.stdout.read(320)                    # 20 ms
        if not bloc:
            journal("parec s'est arrete"); return
        for (ech,) in struct.iter_unpack("<h", bloc[: len(bloc) // 2 * 2]):
            struct.pack_into("<h", ul, 8 + 2 * (w % N), ech)
            w = (w + 1) & 0xFFFFFFFF
        struct.pack_into("<I", ul, 0, w)             # compteur en dernier


def en_attente(fd):
    """Octets deja ecrits dans le tuyau vers pacat et pas encore lus par lui."""
    try:
        return struct.unpack("i", fcntl.ioctl(fd, termios.FIONREAD, b"\0\0\0\0"))[0]
    except OSError:
        return 0


def haut_parleur(dl):
    """anneau DL -> pacat, a retard borne.

    [2026-10-16] LA VOIX REVENAIT 4,5 S APRES. L'ancienne boucle ecrivait dans
    pacat tout ce que l'anneau contenait, ET du silence chaque fois qu'il etait
    vide (DSP en retard). pacat consomme a 8 kHz exacts : chaque silence ajoute
    puis les echantillons du DSP qui arrivent ensuite en rafale s'empilaient
    dans le tuyau (64 Kio = 4 s), sans jamais se resorber. Mesure sur la pile
    vivante (FIONREAD sur le stdin de pacat) : 3,30 s en attente. Desormais le
    tuyau n'est rempli que jusqu'a DL_CIBLE (defaut 150 ms) ; le silence ne
    sert que si le tuyau va se vider ; l'anneau ne garde pas plus de GIGUE
    echantillons de retard (le surplus est jete, une fois, et compte)."""
    cmd = ["pacat", "--playback", f"--device={SINK}", "--format=s16le", f"--rate={TAUX}",
```

```
            "--channels=1", "--raw", "--latency-msec=60"]
    p = subprocess.Popen(cmd, stdin=subprocess.PIPE)
    journal("haut-parleur :", " ".join(cmd))
    fd = p.stdin.fileno()
    cible = DL_CIBLE
    lu = None
    n_total = n_silence = n_jete = 0
    file_max = 0
    t_journal = time.monotonic()
    while True:
        w = struct.unpack_from("<I", dl, 0)[0]
        if lu is None:
            lu = (w - GIGUE) & 0xFFFFFFFF
        dispo = (w - lu) & 0xFFFFFFFF
        if dispo > N or dispo > GIGUE + cible:           # retard de l'anneau : on rattrape
            n_jete += (dispo - GIGUE) if dispo <= N else 0
            lu = (w - GIGUE) & 0xFFFFFFFF
            dispo = GIGUE
        file = en_attente(fd) // 2
        file_max = max(file_max, file)
        manque = cible - file
        if manque >= 160:
            k = min(dispo, manque)
            out = bytearray()
            for i in range(k):
                out += dl[8 + 2 * ((lu + i) % N): 10 + 2 * ((lu + i) % N)]
            lu = (lu + k) & 0xFFFFFFFF
            n_total += k
            if file + k < 320:                           # le tuyau va se vider : silence
                out += bytes(2 * (320 - file - k)); n_silence += 320 - file - k
            if out:
                p.stdin.write(out); p.stdin.flush()
        time.sleep(0.01)
        if time.monotonic() - t_journal > 30:
            journal(f"DL : {n_total} echantillons du DSP, {n_silence} de silence, {n_jete} jetes (retard), "
                    f"tuyau vers pacat <= {file_max * 1000 // TAUX} ms (cible {cible * 1000 // TAUX} ms) en 30 s")
            n_total = n_silence = n_jete = file_max = 0; t_journal = time.monotonic()


def pactl(*args):
    r = subprocess.run(["pactl", *args], capture_output=True, text=True)
    return r.stdout if r.returncode == 0 else ""


def micro_reel():
    """[2026-10-16] Le vrai micro de l'operateur jusqu'a gsm_mic, donc jusqu'a DRR.

    Le cablage existe (lib/audio.sh : annuleur d'echo osmo_mic_ec, puis
    module-loopback osmo_mic_ec -> gsm_mic), mais le 16/10 l'entree materielle
    etait a 0 % et osmo_mic_ec en sourdine : le DSP ne recevait que le ton
    injecte par banc-max (« crete UL 0 » dans les lignes [vsp]). lib/audio.sh
    ne regle le volume qu'a la creation de l'annuleur. On le reverifie ici, a
    chaque lancement du port voix. VSP_MICRO_REEL=0 laisse le micro tel quel."""
    if os.environ.get("VSP_MICRO_REEL", "1") == "0":
        return
    sources = [l.split("\t")[1] for l in pactl("list", "short", "sources").splitlines() if "\t" in l]
    materiel = [s for s in sources if s.startswith("alsa_input") and not s.endswith(".monitor")]
    micro = "osmo_mic_ec" if "osmo_mic_ec" in sources else (materiel[0] if materiel else None)
    if not micro:
        journal("micro reel : aucune entree materielle")
        return
    volume = os.environ.get("VSP_MICRO_VOLUME", os.environ.get("EC_MIC_VOLUME", "25%"))
    for s in dict.fromkeys(materiel[:1] + [micro]):
        pactl("set-source-mute", s, "0")
        if " 0%" in pactl("get-source-volume", s) or "/   0%" in pactl("get-source-volume", s):
            pactl("set-source-volume", s, volume)
```

```
    boucles = [l for l in pactl("list", "short", "modules").splitlines()
               if "module-loopback" in l and "sink=gsm_mic" in l]
    if not any(f"source={micro} " in l + " " for l in boucles):
        pactl("load-module", "module-loopback", f"source={micro}", "sink=gsm_mic",
              "sink_dont_move=true", "source_dont_move=true", "latency_msec=40")
        journal(f"micro reel : bouclage {micro} -> gsm_mic pose")
    journal(f"micro reel : {micro} -> gsm_mic (sourdine levee, volume {volume} si a 0)")


def main():
    ul, dl = anneau(SHM_UL), anneau(SHM_DL)
    journal(f"{SOURCE} -> {SHM_UL} ; {SHM_DL} -> {SINK} ; gigue {GIGUE} echantillons")
    micro_reel()
    threading.Thread(target=micro, args=(ul,), daemon=True).start()
    haut_parleur(dl)


if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        sys.exit(0)
```

3.323 /opt/GSM/osmo-operator/pont/dsp/init.py

5162 octets, 112 lignes → 112 lignes

```
# pont/dsp - le pont du montage DSP (c54x_exe --arm, BSP du C54x sur udp/6702).
#
# [2026-09-23] Le paquet pont/ servait tel quel aux deux montages, et tout son
# etat dedie (TCH, chiffrement, Kc) vivait a l'heure du pont, c'est-a-dire de
# la BTS. En DSP le BSP joue les trames a l'heure du C54x, 12 a ~200 trames
# plus tard : chaque decision prise au decodage frappait des trames que le
# mobile n'avait pas encore entendues. Ce sous-paquet assemble les memes
# briques avec des sous-classes propres au DSP :
#   ClockDsp    l'avance de l'horloge suit l'avance reelle de la BTS
#               (osmotrx fn-advance et ses a-coups), mesuree en continu.
#   TchDsp      l'ASSIGNMENT COMMAND n'est qu'une annonce ; c54x_exe bascule
#               le BSP sur la tache TCH du firmware (montant.c) ; abandon()
#               rend le SDCCH apres une ASSIGNMENT FAILURE.
#   TrxDsp      l'intervalle du TCH est dechiffre des l'annonce.
#   UplinkDsp   un bloc SDCCH montant n'est jamais une FACCH ; le Kc n'est
#               plus lache a la liberation vue par le mobile.
#   FeederDsp,
#   DownlinkDsp pas de GSMTAP 4730/4731 ni d'anneau calypso_tch_dl (grgsm).
# pont/__init__.py (et donc pont.py, le montage grgsm) n'importe jamais ce
# sous-paquet.
import gc
import logging
import os
import signal
import sys
import time

from .. import _guarded
from .. import config as _config
from ..cipher import Cipher
from ..gsm import load_timeslots, plan_for
from ..record import Recorder
from ..stats import Reporter, Stats
from ..trx import Transmitter
from ..uplink import TchScheduler
from .clock import ClockDsp
from .dedicated import DedicatedDsp
```

```
from .downlink import DownlinkDsp, FeederDsp
from .tch import TchDsp
from .trx import TrxDsp
from .uplink import UplinkDsp

log = logging.getLogger("pont")


def main(argv=None):
    cfg = _config.parse(argv)
    if not cfg.dsp_port:
        sys.exit("pont DSP : --dsp-port (ou PONT_DSP_PORT) est obligatoire, 6702 pour c54x_exe ; "
                 "pour le montage grgsm, lancer pont/pont.py")
    logging.basicConfig(stream=sys.stdout, level=logging.INFO,
                        format="%(asctime)s [pont] %(message)s", datefmt="%H:%M:%S")
    gc.disable()
    gc.freeze()

    stats = Stats()
    clock = ClockDsp()
    cipher = Cipher(cfg.kc_retention, stats)
    dedicated = DedicatedDsp()     # [2026-09-23] genres TCH du tap, voir dsp/dedicated.py
    tch = TchDsp(stats)
    record = Recorder(cfg, clock) if cfg.record else None
    trx = TrxDsp(cfg, clock, stats, cipher, record, dedicated, tch)
    transmitter = Transmitter(cfg, clock, trx, stats)
    feeder = FeederDsp(cfg, stats)
    timeslots = load_timeslots(cfg.bsc_cfg)
    downlink = DownlinkDsp(cfg, stats, cipher, dedicated, tch, feeder, record, timeslots)
    uplink = UplinkDsp(cfg, clock, stats, dedicated, tch, transmitter, feeder, cipher)
    scheduler = TchScheduler(cfg, clock, stats, tch, uplink, transmitter)
    if os.environ.get("PONT_VOIX_HOTE", "0") == "1":
        # [2026-10-22] la voix du Magnetite par l'hote (libgsm), voir pont/voix_hote.py
        from ..voix_hote import VoixHote
        voix = VoixHote(uplink, tch, stats)
        downlink.ring = voix
        voix.start()
    if os.environ.get("PONT_UL_RAW", "0") == "1":
        # [2026-10-04] tuyau unique : les bursts de la ROM partent par `transmitter`, ceux que l'hote
        # code encore (RACH, SDCCH, FACCH, SACCH, parole) vont au puits. Voir dsp/uplink.py RomTxRing.
        from .uplink import TransmitterPuits
        uplink.tx_rom = transmitter
        uplink.tx = TransmitterPuits(stats, transmitter, tch)   # le TCH passe au vrai Transmitter (UL_RAW_TCH)
        scheduler.tx = uplink.tx
       log.info("PONT_UL_RAW=1 : montant = bursts finaux de la ROM (calypso_tx_rom) ; codage et A5 hote inactifs")

    log.info("pont TRX : ports %d/%d/%d, ARFCN %d, BSIC %d, avance UL %d trames",
             cfg.trx_base, cfg.trx_base + 1, cfg.trx_base + 2, cfg.arfcn, cfg.bsic, cfg.ul_fn_advance)
    log.info("pont DSP : bursts vers le BSP udp/%d, bascule TCH suivie par le firmware (montant.c), "
             "retard UL max %d trames, retention Kc %s",
             cfg.dsp_port, cfg.ul_retard_max, "oui" if cfg.kc_retention else "non")
    for tn in sorted(timeslots):
        plan = plan_for(timeslots[tn])
        log.info("TS%d %-12s %s", tn, timeslots[tn], ("plan %s, %d sous-voies" % (plan.name, plan.n_sub)) if plan
↪   else "")
    log.info("QEMU : pas de GSMTAP 4730/4731 (couche 1 gr-gsm desactivee) ; tap GSMTAP %s",
             ("udp/%d" % cfg.tap_port) if cfg.tap else "coupe")

    def _dl(tn, fn, bits):
        clock.note_dl(fn)       # avance reelle de la BTS, voir dsp/clock.py
        downlink.dispatch(tn, fn, bits)

    threads = [
        _guarded("ctrl", trx.run_ctrl),
        _guarded("clock", trx.run_clock),
        _guarded("data", lambda: trx.run_data(_dl)),
```

```
        transmitter, uplink, scheduler,
        Reporter(stats, clock, dedicated, record),
    ]
    if record:
        signal.signal(signal.SIGUSR1, record.split)
        threads.append(record)
    for t in threads:
        t.start()
    while True:
        time.sleep(3600)
```

3.324 /opt/GSM/osmo-operator/pont/dsp/clock.py

5929 octets, 116 lignes → 116 lignes

```
# pont/dsp/clock.py - l'horloge du pont DSP : l'avance suit la BTS.
#
# [2026-09-23] L'AVANCE DE LA BTS EST REPORTEE EN DYNAMIQUE SUR LE DSP.
#
# Clock (pont/trx.py) asservit l'horloge du banc a « trame reclamee par le DSP
# + PONT_HORLOGE_AVANCE (12) ». C'est la BTS qui emet, elle, et ses bursts
# descendants partent `osmotrx fn-advance` trames AVANT cette horloge (2 par
# defaut), plus ses a-coups : « We missed 6 timers », « 5 FN slower than
# TRX », puis rattrapage d'un bloc. La marge vraiment utile au BSP -- derniere
# trame recue de la BTS moins trame que le DSP reclame -- valait donc
# 12 + fn-advance - gigue, et tombait sous zero : run de 13:41, 1000 trames
# sans burst (« le DSP COURT DEVANT le BTS », ecarts +1 a +6), chacune jouee en
# effacement. Tombee sur un UA ou une FACCH, c'est un bloc perdu : « I frame
# ignored in state SABM_SENT » (l'UA manque, les I suivantes arrivent),
# T200 sur la FACCH de l'assignation.
#
# ClockDsp mesure a chaque regulation l'avance reelle de la BTS sur l'horloge
# (derniere trame descendante recue - trame de l'horloge), en garde le MINIMUM
# sur une fenetre glissante (les calages de la BTS sont des creux), et pose
#     avance = PONT_MARGE_DL - avance_BTS_min
# pour que « derniere trame recue - trame reclamee » reste >= PONT_MARGE_DL.
# Changer `osmotrx fn-advance` a chaud sur la BTS est suivi sans rien relancer.
# L'avance est bornee [PONT_AVANCE_MIN, PONT_AVANCE_MAX] : au-dela, le montant
# part d'autant plus en retard (Transmitter, PONT_UL_RETARD_MAX).
# PONT_MARGE_DL=0 revient a l'avance fixe de Clock.
# Defaut 16 : 8 a d'abord ete essaye (run de 13:48) et a DOUBLE les trames sans
# burst (6 % contre 1,6 % avec l'avance fixe 12 + fn-advance 2, soit ~14) --
# l'echantillonnage toutes les 50 ms ne voit pas les pires creux de la BTS.
#
# [2026-09-23 19:06] L'AVANCE BTS SE MESURE SUR LE POINT VISE, PAS SUR L'HORLOGE.
# Mesuree contre l'horloge reelle (derniere recue - horloge), elle supposait
# horloge == DSP + avance. Faux des que le DSP attend la BTS
# (CALYPSO_BSP_ATTENTE_MS) : le DSP colle alors a la derniere trame recue,
# l'horloge traine ~15 trames derriere son but, l'« avance BTS » lue monte a
# +12 et l'avance visee tombe au plancher (4). Marge reelle min -3 moy +1.2,
# une attente de 40 ms echue sur six, trame du banc a 11,5 ms au lieu de 4,6,
# SDCCH descendant troue : UA perdu, le mobile repete son SABM (« SABM frame
# with information not allowed in this state » au BSC, « Dropping frame with
# ~96 bit errors » au mobile). On mesure donc derniere recue - (DSP + avance) :
# la marge reelle, moins l'avance deja demandee -- boucle fermee sur ce qu'on
# veut tenir. Plancher releve a 10 (l'avance fixe 12 tenait a 1,6 %).
import collections
import logging
import os
import time

from .. import gsm
from ..trx import Clock
```

```
log = logging.getLogger("pont")

MARGE_DL = int(os.environ.get("PONT_MARGE_DL", "16"))
AVANCE_MIN = int(os.environ.get("PONT_AVANCE_MIN", "10"))
AVANCE_MAX = int(os.environ.get("PONT_AVANCE_MAX", "40"))
FENETRE_S = float(os.environ.get("PONT_MARGE_FENETRE", "3.0"))


class ClockDsp(Clock):
    def __init__(self):
        super().__init__()
        self._fn_dl = None                  # derniere trame descendante recue de la BTS
        self._mesures = collections.deque() # (instant, avance BTS sur l'horloge)
        self._avance_dite = None

    def note_dl(self, fn):
        """Appele par le fil TRXD a chaque burst descendant (sans verrou :
        une affectation d'entier)."""
        prec = self._fn_dl
        if prec is None or 0 < (fn - prec) % gsm.HYPERFRAME < gsm.HYPERFRAME // 2:
            self._fn_dl = fn

    def _reguler(self, now):
        du = now >= self._echeance
        super()._reguler(now)                   # rafraichit _dsp_prec (trame reclamee)
        dsp = self._dsp_prec
        if MARGE_DL > 0 and du and dsp is not None and self._fn_dl is not None:
            # Avance de la BTS sur le point VISE (DSP + avance), pas sur
            # l'horloge reelle -- voir l'en-tete, releve du 2026-09-23 19:06.
            avance_bts = (self._fn_dl - dsp - self._avance) % gsm.HYPERFRAME
            if avance_bts > gsm.HYPERFRAME // 2:
                avance_bts -= gsm.HYPERFRAME
            m = self._mesures
            m.append((now, avance_bts))
            while m and now - m[0][0] > FENETRE_S:
                m.popleft()
            creux = min(a for _, a in m)
            avance = min(max(MARGE_DL - creux, AVANCE_MIN), AVANCE_MAX)
            if avance != self._avance:
                self._avance = avance
                if self._avance_dite is None or abs(avance - self._avance_dite) >= 3:
                    self._avance_dite = avance
                    log.info("horloge : avance BTS mesuree %+d trames (creux sur %.0f s), "
                             "avance visee %d pour une marge DL >= %d",
                             creux, FENETRE_S, avance, MARGE_DL)
        self._mesurer_marge(now)

    def _mesurer_marge(self, now):
        """Marge REELLE vue par le BSP : derniere trame recue - trame reclamee
        par le DSP (calypso_horloge). Journalisee toutes les 10 s (min/moy) :
        c'est elle qui doit rester > 0, pas l'avance visee."""
        dsp = self._dsp_prec
        if dsp is None or self._fn_dl is None:
            return
        marge = (self._fn_dl - dsp) % gsm.HYPERFRAME
        if marge > gsm.HYPERFRAME // 2:
            marge -= gsm.HYPERFRAME
        st = getattr(self, "_marges", None)
        if st is None:
            st = self._marges = [now, marge, 0, 0]
        st[1] = min(st[1], marge)
        st[2] += marge
        st[3] += 1
        if now - st[0] >= 10.0:
            log.info("horloge : marge DL reelle (recu - reclame par le DSP) min %+d moy %+.1f "
                     "sur %d mesures, avance visee %d", st[1], st[2] / st[3], st[3], self._avance)
```

```
            self._marges = [now, marge, 0, 0]
```

3.325 /opt/GSM/osmo-operator/pont/dsp/dedicated.py

4371 octets, 102 lignes → 102 lignes

```
# pont/dsp/dedicated.py - calypso_dcch_cfg tel que l'ecrit le tap QEMU du
# montage DSP (qosmo hw/arm/calypso/calypso_dcch_tap.c).
#
# [2026-09-23] Le tap annonce maintenant aussi le TCH : genre 2 = TCH/F,
# 3 = TCH/H. Il ne connaissait que le SDCCH, et pendant tout un appel le
# fichier disait encore SDCCH/8 TS1. Dedicated.read (pont/state.py) ne garde
# que le bit 0 du genre : un genre 2 y deviendrait un SDCCH/4 sur le TS du TCH
# -- le montant SDCCH d'une ASSIGNMENT FAILURE serait alors planifie sur TS2.
#
# Ici le TCH est note a part et plan() continue de rendre le DERNIER SDCCH tant
# que la connexion vit : c'est exactement ce que voyaient DownlinkDsp, TrxDsp et
# UplinkDsp quand le tap ignorait le TCH (aucun changement de comportement), et
# c'est le bon canal si le mobile y revient (ASSIGNMENT FAILURE). Seul 0xFF
# (retour sur les voies communes) termine la connexion : UplinkDsp._poll_release
# ne voit donc pas l'arrivee sur le TCH comme une liberation.
# pont.py (grgsm) garde state.Dedicated : sa couche 1 n'ecrit jamais de genre 2.
import logging
import os
import struct
import time

from ..state import DCCH_CFG, DCCH_RELEASED, DCCH_TTL, Dedicated

log = logging.getLogger("pont")

DCCH_TCH_F = 2
DCCH_TCH_H = 3
# [2026-10-15] genre 4 = PDCH (GPRS) : le firmware l1gprs_fw.c est sur un PDCH ; le
# tap QEMU (calypso_dcch_tap.c) le publie a partir des L1CTL_GPRS_DL_BLOCK_IND.
DCCH_PDCH = 4


class DedicatedDsp(Dedicated):
    def __init__(self):
        super().__init__()
        self._tch = None        # (genre, ss, tn) : le firmware est sur le TCH
        self._pdch = None       # tn : le firmware est sur un PDCH (GPRS)

    def read(self):
        now = time.monotonic()
        if now < self._next:
            return self._value
        self._next = now + DCCH_TTL
        try:
            if self._fd is None:
                self._fd = os.open(DCCH_CFG, os.O_RDONLY)
            b = os.pread(self._fd, 16, 0)
        except OSError:
            if self._fd is not None:
                os.close(self._fd)
                self._fd = None
            return self._value
        if len(b) < 8:
            return self._value
        seq = struct.unpack_from("<I", b, 0)[0]
        if not seq or seq == self._seq:
            return self._value
        self._seq = seq
```

```
        genre, ss, tn = b[4], b[5] & 7, b[6] & 7
        if genre == DCCH_RELEASED:
            self._value = None
            self._tch = None
            self._pdch = None
        elif genre == DCCH_PDCH:
            self._pdch = tn
            log.info("canal dedie : PDCH TS=%d annonce par le firmware (GPRS)", tn)
        elif genre in (DCCH_TCH_F, DCCH_TCH_H):
            self._pdch_quitte("TCH", tn)
            self._tch = (genre, ss, tn)
            if self._value is None:
                log.info("canal dedie : TCH/%s TS=%d SS=%d annonce par le firmware, aucun SDCCH connu",
                         "F" if genre == DCCH_TCH_F else "H", tn, ss)
            else:
                log.info("canal dedie : TCH/%s TS=%d SS=%d annonce par le firmware "
                         "(SDCCH SS=%d TS=%d garde pour un retour)",
                         "F" if genre == DCCH_TCH_F else "H", tn, ss, self._value[1], self._value[2])
        else:
            self._pdch_quitte("SDCCH", tn)
            if self._tch is not None:
                log.info("canal dedie : le firmware a quitte le TCH TS=%d pour le SDCCH SS=%d TS=%d",
                         self._tch[2], ss, tn)
            self._tch = None
            self._value = (genre & 1, ss, tn)
        return self._value

    def _pdch_quitte(self, quoi, tn):
        # [2026-10-15] Le firmware ne peut pas etre sur un PDCH ET sur un SDCCH/TCH : _pdch
        # n'etait efface qu'au DCCH_RELEASED, et tn_dedie() (dsp/uplink.py), qui teste le PDCH
        # en premier, aurait envoye les bursts ROM du canal dedie sur le TS du PDCH.
        if self._pdch is not None:
            log.info("canal dedie : le firmware a quitte le PDCH TS=%d pour un %s TS=%d", self._pdch, quoi, tn)
            self._pdch = None

    def tch(self):
        """(genre, ss, tn) si le firmware est sur un TCH d'apres le tap, sinon None."""
        self.read()
        return self._tch

    def pdch(self):
        """tn si le firmware est sur un PDCH (GPRS) d'apres le tap, sinon None."""
        self.read()
        return self._pdch
```

3.326 /opt/GSM/osmo-operator/pont/dsp/downlink.py

999 octets, 29 lignes → 29 lignes

```
# pont/dsp/downlink.py - le descendant du pont DSP, sans les sorties grgsm.
#
# [2026-09-23] En montage DSP, QEMU tourne sous CALYPSO_DSP_EXTERN=1 : la
# couche 1 gr-gsm y est desactivee, et c'est la seule a ecouter le GSMTAP
# udp/4730, le SCH udp/4731 et l'anneau de parole /dev/shm/calypso_tch_dl
# (calypso_l1_grgsm.c:34). Le DSP recoit, lui, les bursts bruts (Trx.run_data
# -> udp/6702). On garde le tap Wireshark (udp/4729) ; le decodage local du
# pont (statistiques, confirmation du chiffrement, ASSIGNMENT COMMAND, CHANNEL
# RELEASE) est inchange.
from ..downlink import Downlink, Feeder


class FeederDsp(Feeder):
    def l2(self, fn, chan, l2, tn=0, uplink=False, to_qemu=True):
        super().l2(fn, chan, l2, tn, uplink, to_qemu=False)
```

```
    def sch(self, fn):
        pass


class _NullRing:
    def publish(self, fr33, fn):
        pass


class DownlinkDsp(Downlink):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.ring = _NullRing()
```

3.327 /opt/GSM/osmo-operator/pont/dsp/tch.py

2225 octets, 52 lignes → 52 lignes

```
# pont/dsp/tch.py - le TCH vu du pont DSP : une ANNONCE, pas une bascule.
#
# [2026-09-23] En montage DSP, /dev/shm/calypso_tch_cfg est lu par c54x_exe
# (src/montant.c, scruter_tch). Jusqu'ici il y armait le BSP sur le TCH des
# que le pont decodait l'ASSIGNMENT COMMAND, a l'heure de la BTS, alors que le
# BSP joue les trames a l'heure du DSP, 12 a ~200 trames plus tard. Le BSP
# remplacait alors toutes les trames pas encore jouees -- l'ASSIGNMENT COMMAND
# comprise, et TS0 avec -- par des bursts du TS du TCH. Run de 12:22, appels 2
# et 3 : le mobile n'a jamais recu l'ASSIGNMENT COMMAND, 80-100 bit errors,
# LOS. Desormais c54x_exe ne bascule qu'a la premiere tache TCH posee par le
# firmware (montant.c, suivre_tache_tch) ; l'ecriture faite ici n'est plus
# qu'une annonce (intervalle, TSC).
#
# Meme fichier, meme format, meme moment d'ecriture que Tch.arm : la L1 grgsm
# (calypso_l1_grgsm.c poll_tch_cfg) n'est pas concernee, pont.py n'utilise pas
# cette classe. Le seul cas nouveau, seq non nul avec tn=0, est l'abandon.
import logging

from ..state import Tch

log = logging.getLogger("pont")


class TchDsp(Tch):
    def arm(self, tn, tsc, arfcn):
        with self.lock:
            self.seq += 1
            self.tn = tn
            self.tsc = tsc
            self.open = False
            self._write_cfg(tn, tsc, arfcn, self.seq)
        log.info("ASSIGNMENT COMMAND : TCH TN=%d TSC=%d ARFCN=%d annonce au DSP, "
                 "bascule du BSP a la premiere tache TCH du firmware", tn, tsc, arfcn)

    def abandon(self, reason):
        """Le mobile est reste (ou revenu) sur le SDCCH : retirer l'annonce.

        seq+1 avec tn=0 : c54x_exe rend le SDCCH au BSP SANS lacher le Kc (le
        SDCCH vit encore, la BTS le chiffre toujours). seq=0 reste la
        liberation complete (Tch.close)."""
        with self.lock:
            if self.tn is None:
                return
            tn = self.tn
            self.seq += 1
```

```
            self.tn = None
            self.open = False
            self.release_seen = None
            self._write_cfg(0, 0, 0, self.seq)
        log.info("TCH TN=%d abandonne (%s) : retour sur le SDCCH", tn, reason)
        for cb in self.on_close:
            cb()
```

3.328 /opt/GSM/osmo-operator/pont/dsp/trx.py

2032 octets, 39 lignes → 39 lignes

```
# pont/dsp/trx.py - le Trx du pont DSP.
#
# La copie de chaque burst vers le BSP du C54x (UDP dsp_port, 8 octets
# d'en-tete + 148 bits 0/1) reste dans Trx.run_data : pont.dsp.main refuse de
# demarrer sans --dsp-port, donc ce chemin y est toujours pris.
import os

from ..trx import Trx


class TrxDsp(Trx):
    # [2026-09-23] LE DSP DECHIFFRE LUI-MEME. La mask-ROM programme le
    # coprocesseur A5 du Calypso (ports XIO 0x2800..0x2818, voir
    # l1-dsp/calypso_a5.c) avec a_kc et a_a5fn, et applique le flux au burst
    # demodule. Ce coprocesseur n'etait pas modelise : le flux restait nul, d'ou
    # le dechiffrement ici. Il l'est desormais, et dechiffrer ici en plus
    # appliquerait le flux deux fois (= renvoyer le burst chiffre au decodeur).
    # Le BSP recoit donc le descendant tel que la BTS l'a emis.
    # PONT_DSP_DECHIFFRE=1 retablit l'ancien comportement (coeur sans modele A5,
    # ou c54x_exe lance avec CALYPSO_A5=0).
    dechiffre_dl = os.environ.get("PONT_DSP_DECHIFFRE", "0") == "1"

    def _burst_dedie(self, tn, fn):
        """[2026-09-23] LE TCH EST DECHIFFRE DES L'ANNONCE.

        Trx._burst_dedie ne comptait l'intervalle du TCH comme dedie qu'une
        fois Tch.prove() passe, c'est-a-dire apres une premiere trame montante
        du mobile sur le TCH. Or osmo-bts chiffre le TCH dans les deux sens
        des l'activation du canal (osmo-bts-trx l1_if.c:479-483) : entre la
        bascule du firmware et prove(), le BSP jouait des bursts TS2 chiffres,
        et les premiers blocs de la BTS -- l'UA qui repond au SABM de
        l'ASSIGNMENT -- etaient perdus cote ROM (appel 1 du run de 12:22).
        Le dechiffrement reste garde par cipher.dl_active (Trx.run_data),
        confirme sur le SDCCH de la MEME session et remis a zero a l'IMMEDIATE
        ASSIGNMENT : un Kc retenu d'une session precedente ne touche jamais un
        TCH en clair."""
        if self.tch is not None and tn == self.tch.active_tn():
            return True
        return super()._burst_dedie(tn, fn)
```

3.329 /opt/GSM/osmo-operator/pont/dsp/uplink.py

23248 octets, 446 lignes → 444 lignes (1 groupes compactés)

```
# pont/dsp/uplink.py - le montant du pont DSP.
#
# Les bandes laterales sont celles de c54x_exe (src/montant.c), memes chemins
# et meme format qu'en grgsm. Deux regles changent, parce qu'en DSP on sait
# d'ou vient chaque bloc et que la liberation se lit a l'heure du DSP.
import collections
```

```
import logging
import os
import struct
import time

from .. import gsm
from ..uplink import POLL, RELEASE_FALLBACK, Uplink

log = logging.getLogger("pont")

RR_ASSIGNMENT_FAILURE = 0x2f

# [2026-10-03] Bursts xCCH montants emis par la ROM DSP, publies par c54x_exe (src/pont.c,
# tx_rom_publier) : la ROM copie chaque burst du tampon entrelace 0x4280 vers data[0x3f8a] a sa
# trame d'emission. Anneau : en-tete w(u32, nombre ecrit) n(u32, cases) ; case de 1024 octets :
# seq(u32) fn_burst0(u32) l2(23) .(1), 4 x 116 ubits (57 donnees, hl, hu, 57 donnees), puis 4 x 116
# ubits du flux de cle A5 montant que la ROM XORe sur la rafale en mode chiffre (data[0x3f9b],
# calypso_a5.c). La ROM chiffre a l'heure du DSP (12 a ~200 trames apres la BTS) : on defait son XOR et
# le pont rechiffre au fn de l'air (Cipher.apply), comme pour un bloc code par l'hote.
SB_XCCH_UL_ROM = "/dev/shm/calypso_xcch_ul_rom"
ROM_SLOT = 1024
# Les bursts de la ROM sortent 1 a ~5 trames apres la publication du L2 par montant.c (codage
# puis une trame par burst). Au-dela, le bloc part code par l'hote. PONT_UL_ROM=0 : toujours l'hote.
ROM_ATTENTE = float(os.environ.get("PONT_UL_ROM_ATTENTE", "0.12"))

# [2026-10-04] LE TUYAU UNIQUE : LES BURSTS MONTANTS FINAUX DE LA ROM, TELS QUELS, A SON NUMERO DE TRAME.
#
# c54x_exe (src/tsp_tx.c, pont.c tx_tsp_publier) publie dans calypso_tx_rom chaque burst que la ROM
# a fini de construire dans son script TSP pour l'ABB : sequence d'apprentissage, queues, inversion
# I/Q et chiffrement A5 deja appliques par la ROM, 148 bits, avec le numero de trame que l'ARM lui a
# donne pour ce burst (= celui du chiffrement : dsp_tester tx-sdcch-a5, 24/24). Avec PONT_UL_RAW=1 le
# pont les envoie a la BTS tels quels, a ce fn, sans codage ni A5 cote hote : RACH sur TS0, bursts
# normaux sur le TN du canal dedie (TCH si ouvert, sinon le SDCCH annonce). La BTS accepte un burst
# montant quel que soit son retard (osmo-bts trx_sched_ul_burst ne rejette pas sur le fn) ; le
# chemin hote classique continue de tourner pour les journaux, les stats et l'etat (GSMTAP), mais
# ses bursts vont dans un puits (Transmitter remplace par TransmitterPuits). PONT_UL_RAW_DECAL=<n>
# ajoute n trames au fn de la ROM (0 par defaut : la ROM chiffre a ce fn, la BTS dechiffre a ce fn).
SB_TX_ROM = "/dev/shm/calypso_tx_rom"
TX_ROM_SLOT = 160
UL_RAW = os.environ.get("PONT_UL_RAW", "0") == "1"
PDCH_FILTRE = os.environ.get("PONT_PDCH_FILTRE", "1") != "0"
# [2026-10-15] B5 : SACCH/T montant par la ROM. 0 (defaut) = mesure seulement (retard fn,
# blocs ROM disponibles) ; la bascule viendra une fois la gigue mesuree.
UL_RAW_SACCH_T = os.environ.get("PONT_UL_RAW_SACCH_T", "0") == "1"
UL_RAW_DECAL = int(os.environ.get("PONT_UL_RAW_DECAL", "0"))
# [2026-10-04 soir] LE TCH RESTE A L'HOTE. Avec montant.c qui laisse B_BLUD a la ROM, la ROM emet bien
# le SABM FACCH et la SACCH/T sur le TS2 (appel de 19:18 : SABM recu par la BTS, connect ack), mais
# AUCUN burst de parole (rejeu : 1290 trames a_du consommees, zero burst hl=hu=0 ; le serialiseur
# `cc 0x8900,tc` de 0x8726 n'est jamais appele pour la parole). Envoyer le TCH de la ROM, c'est donc
# couper la voix montante ; melanger bursts ROM et hote sur le meme TCH casse la FACCH (osmo-bts
# decode au bid 3 de chaque burst recu et decale son tampon a chaque bid 0). Tant que la ROM n'emet
# pas la parole : RACH et SDCCH/SACCH par la ROM, le TCH entier (FACCH, SACCH/T, parole) par l'hote.
# PONT_UL_RAW_TCH=1 : tout par la ROM (voix montante coupee).
UL_RAW_TCH = os.environ.get("PONT_UL_RAW_TCH", "0") == "1"
# [2026-10-21] Magnetite (firmware TI) : la ROM n'emet pas ses bursts par le script TSP que lit calypso_tx_rom,
# sauf le RACH que c54x_exe y publie lui-meme (tx_rach_vers_air). Le SABM et la SACCH, captures dans a_cu par
# montant.c (calypso_sdcch_ul), partaient au puits (« DL: >N200 of SABM »). PONT_UL_RAW_DCCH_HOTE=1 : SDCCH et
# SACCH codes par l'hote et envoyes a la BTS, le reste du tuyau ROM inchange.
UL_RAW_DCCH_HOTE = os.environ.get("PONT_UL_RAW_DCCH_HOTE", "0") == "1"
UL_TN_DEFAUT = int(os.environ["PONT_UL_TN_DEFAUT"]) if os.environ.get("PONT_UL_TN_DEFAUT", "") != "" else None


class RomTxRing:
    """Lecteur de calypso_tx_rom : (seq, fn, type, tsc, bits[148]) dans l'ordre d'ecriture."""
    def __init__(self):
        self.fd = None
```

```
        self.seq = 0            # dernier seq consomme

    def nouveaux(self):
        try:
            if self.fd is None:
                self.fd = os.open(SB_TX_ROM, os.O_RDONLY)
            hdr = os.pread(self.fd, 8, 0)
        except OSError:
            if self.fd is not None:
                os.close(self.fd)
                self.fd = None
            return []
        if len(hdr) < 8:
            return []
        w, n = struct.unpack("<II", hdr)
        if w <= self.seq:
            return []
        debut = max(self.seq + 1, w - n + 1)
        out = []
        for seq in range(debut, w + 1):
            case = os.pread(self.fd, TX_ROM_SLOT, 8 + ((seq - 1) % n) * TX_ROM_SLOT)
            if len(case) < TX_ROM_SLOT or struct.unpack_from("<I", case, 0)[0] != seq:
                continue
            fn, typ, tsc, _off, drap = struct.unpack_from("<IBBBB", case, 4)
            # drap bit 7 : burst chiffre par la ROM (c54x_exe pont.c tx_tsp_publier, 2026-10-15)
            out.append((seq, fn, typ, tsc, bytes(case[12:12 + 148]), bool(drap & 0x80)))
        self.seq = w
        return out


def classe_tch(fn, burst):
    """[2026-10-15] Burst montant d'un TCH/F : 'S' SACCH/T (fn%26 = 12, ou 25 selon le TN),
    'F' FACCH (un stealing flag leve : bits 60 / 87, de part et d'autre de la TSC -- meme
    place dans les bursts de la ROM, verifie sur calypso_tx_rom), 'P' parole sinon."""
    if fn % 26 in (12, 25):
        return "S"
    try:
        if burst[60] or burst[87]:
            return "F"
    except (IndexError, TypeError):
        pass
    return "P"


class TransmitterPuits:
    """Puits pour les bursts codes par l'hote quand PONT_UL_RAW=1 : rien ne part, sauf ceux du TCH
    (PONT_UL_RAW_TCH=0, voir UL_RAW_TCH), qui vont au vrai Transmitter."""
    def __init__(self, stats, vrai=None, tch=None):
        self.stats = stats
        self.vrai = vrai
        self.tch = tch
        self.n = 0

    def schedule(self, tn, fn_air, burst, cipher, essais=0):
        if not UL_RAW_TCH and self.vrai is not None and self.tch is not None and tn == self.tch.active_tn():
            self.stats.air_hote_tch += 1
            self.stats.tch_ul_hote[classe_tch(fn_air, burst)] += 1
            if cipher:
                self.stats.a5_ul_hote_tch += 1
            return self.vrai.schedule(tn, fn_air, burst, cipher, essais)
        self.n += 1
        self.stats.air_puits += 1
  ×3
class RomXcchRing:
    def __init__(self):
        self.fd = None
```

```
        self.vus = set()
        self.dechiffres = 0     # blocs que la ROM avait chiffres (XOR defait ici)

    def chercher(self, l2):
        """Les 4 bursts (116 ubits chacun) du bloc le plus recent dont les bits REDECODES donnent
        `l2` ; (None, n_ko) sinon. n_ko = blocs ROM recents qui ne redecodent pas (CRC faux)."""
        try:
            if self.fd is None:
                self.fd = os.open(SB_XCCH_UL_ROM, os.O_RDONLY)
            hdr = os.pread(self.fd, 8, 0)
        except OSError:
            if self.fd is not None:
                os.close(self.fd)
                self.fd = None
            return None, 0
        if len(hdr) < 8:
            return None, 0
        w, n = struct.unpack("<II", hdr)
        ko = 0
        for seq in range(w, max(0, w - n), -1):
            case = os.pread(self.fd, ROM_SLOT, 8 + ((seq - 1) % n) * ROM_SLOT)
            if len(case) < 32 + 8 * 116 or struct.unpack_from("<I", case, 0)[0] != seq or seq in self.vus:
                continue
            bits = case[32:32 + 4 * 116]
            flux = case[32 + 4 * 116:32 + 8 * 116]
            coded = [bits[k * 116:(k + 1) * 116] for k in range(4)]
            dec = gsm.xcch_decode(coded)
            chiffre = False
            if dec is None and any(flux):
                coded = [bytes(a ^ b for a, b in zip(bits[k * 116:(k + 1) * 116], flux[k * 116:(k + 1) * 116]))
                         for k in range(4)]
                dec = gsm.xcch_decode(coded)
                chiffre = True
            if dec is None:
                ko += 1
                self.n_ko = getattr(self, "n_ko", 0) + 1
                if self.n_ko <= 5:
                    log.info("xCCH ROM (0x3f8a) seq=%d : CRC faux en clair%s", seq,
                             " et apres XOR du flux A5" if any(flux) else ", aucun flux A5 joint")
                continue
            if bytes(dec[:gsm.MACBLOCK_LEN]) == bytes(l2[:gsm.MACBLOCK_LEN]):
                self.dechiffres += chiffre
                self.vus.add(seq)
                if len(self.vus) > 64:
                    self.vus = set(x for x in self.vus if x > w - n)
                return [gsm.burst_from_coded(c) for c in coded], ko
        return None, ko


class UplinkDsp(Uplink):
    rom = None
    rom_attente = None

    def _bursts_xcch(self, l2):
        """[2026-10-03] LES BURSTS MONTANTS SDCCH/SACCH SONT CEUX DE LA ROM.

        La ROM du DSP fait le codage canal (FIRE, convolutif, entrelacement) ; c54x_exe publie les
        bursts qu'elle emet (data[0x3f8a]) dans calypso_xcch_ul_rom. On les prend s'ils se
        redecodent en ce L2 ; tant qu'ils ne sont pas sortis (ROM_ATTENTE), le bloc attend ; au-dela,
        ou si PONT_UL_ROM=0, il est code par l'hote comme avant (compteur xCCH ul hote)."""
        if os.environ.get("PONT_UL_ROM", "1") == "0":
            return Uplink._bursts_xcch(self, l2)
        if self.rom is None:
            self.rom = RomXcchRing()
        cle = bytes(l2[:gsm.MACBLOCK_LEN])
        if self.rom_attente is None or self.rom_attente[0] != cle:
```

```
            self.rom_attente = (cle, time.monotonic())
        bursts, ko = self.rom.chercher(l2)
        if bursts is not None:
            self.rom_attente = None
            self.stats.xcch_ul_rom += 1
            if self.stats.xcch_ul_rom <= 3 or self.rom.dechiffres == 1 and not getattr(self, "_dech_dit", False):
                self._dech_dit = self.rom.dechiffres > 0
                log.info("xCCH montant : bursts de la ROM (0x3f8a) pour %s%s", cle[:6].hex(" "),
                         " (chiffres par la ROM, XOR A5 defait, rechiffres par le pont au fn de l'air)"
                         if self.rom.dechiffres else "")
            return bursts
        if time.monotonic() - self.rom_attente[1] < ROM_ATTENTE:
            return None
        self.rom_attente = None
        self.stats.xcch_ul_rom_ko += ko
        if UL_RAW and not UL_RAW_DCCH_HOTE:
            # [2026-10-15] B2 : sous PONT_UL_RAW=1 la copie hote finissait au puits ; on ne la
            # code plus. Liste vide = _poll_sdcch garde ses effets (GSMTAP, sacch_ul, routage)
            # sans rien programmer. Ce qui part a l'antenne reste le burst ROM de calypso_tx_rom.
            self.stats.xcch_shadow_ko += 1
            if self.stats.xcch_shadow_ko <= 5:
                log.info("xCCH montant : %s sans bursts 0x3f8a apres %.0f ms (%d bloc(s) sans CRC) -- "
                         "non recode (PONT_UL_RAW=1, l'antenne a ceux de calypso_tx_rom)",
                         cle[:6].hex(" "), ROM_ATTENTE * 1000, ko)
            return []
        if self.stats.xcch_ul_hote < 5:
            log.info("xCCH montant : pas de bursts ROM pour %s apres %.0f ms (%d bloc(s) ROM sans CRC), code par
↪ l'hote",
                     cle[:6].hex(" "), ROM_ATTENTE * 1000, ko)
        return Uplink._bursts_xcch(self, l2)

    def _route_sdcch(self, l2):
        """[2026-09-23] UN BLOC DE calypso_sdcch_ul N'EST JAMAIS UNE FACCH.

        En DSP, montant.c ne publie sur cette voie que les blocs des taches
        SDCCH (DUL/AUL, capture_sdcch_ul) ; ceux du TCH passent par
        calypso_tch_facch_ul (capture_tch_ul, TCHT). Uplink._route_sdcch les
        envoyait en FACCH des que tch.is_open() : l'ASSIGNMENT FAILURE de
        l'appel 1 du run de 12:22, emise sur TS1, partait donc sur TS2 et la
        BTS ne l'a jamais vue en SDCCH.
        Un tel bloc prouve au contraire que le mobile est sur le SDCCH : si un
        TCH est annonce et que c'est une ASSIGNMENT FAILURE, ou que le mobile
        etait deja passe sur le TCH, l'annonce est retiree (TchDsp.abandon ;
        c54x_exe rend TS1 au BSP s'il ne l'a pas deja fait sur la tache du
        firmware). Le bloc part toujours en SDCCH/SACCH.

        [2026-10-15] B1 : sur un PDCH, montant.c publie aussi les blocs a_cu du GPRS (tache DUL)
        sur cette voie. Ils ne sont pas du SDCCH : on les jette (compteur pdtch_hote_jete).
        PONT_PDCH_FILTRE=0 : ancien comportement."""
        if PDCH_FILTRE and self.dedicated.pdch() is not None:
            self.stats.pdtch_ul_vu_hote += 1
            if self.stats.pdtch_ul_vu_hote == 1:
                log.info("voie SDCCH montante : blocs du PDCH TS%d (GPRS) jetes, la ROM les emet
↪ (PONT_PDCH_FILTRE=0 pour les garder)",
                         self.dedicated.pdch())
            return True
        if self.tch.active_tn() is not None:
            if gsm.rr_message_type(l2)[1] == RR_ASSIGNMENT_FAILURE:
                self.tch.abandon("ASSIGNMENT FAILURE sur le SDCCH")
            elif self.tch.is_open():
                self.tch.abandon("bloc montant du mobile sur le SDCCH")
        return False

    def _poll_release(self):
        """[2026-09-23] LE Kc N'EST PLUS LACHE A LA LIBERATION.
```

```
        La liberation se lit dans calypso_dcch_cfg, publiee quand le firmware
        relit la BCCH : a l'heure du DSP, en retard sur la BTS, qui chiffre
        encore le SDCCH. Uplink._poll_release lachait le Kc (et dl_active) a
        cet instant ; run de 12:22, apres la 4e session : blocs TS1/8
        80/1793 et TS1/40 40/897 en echec cote pont jusqu'a 12:34. Le Kc et
        dl_active tombent desormais a l'IMMEDIATE ASSIGNMENT suivante, decodee
        a l'heure de la BTS (Downlink._signalling), ou avec un nouveau Kc."""
        active = self.dedicated.plan() is not None
        if self.ded_active and not active:
            reason = "canal libere par le mobile"
        elif self.tch.release_overdue(RELEASE_FALLBACK):
            reason = "CHANNEL RELEASE sans liberation du mobile depuis %ds" % RELEASE_FALLBACK
        else:
            self.ded_active = active
            return
        self.ded_active = active
        self.tch.close(reason)
        log.info("%s : Kc garde jusqu'a la prochaine IMMEDIATE ASSIGNMENT (la BTS chiffre encore)", reason)

    def run(self):
        """[2026-10-04] PONT_UL_RAW=1 : voir RomTxRing. Les _poll_* continuent (journaux, stats, etat),
        leurs bursts vont au puits ; ceux de la ROM partent par self.tx_rom.trx.send_ul, TOUT DE SUITE :
        leur fn est definitif (c'est celui du chiffrement de la ROM), la BTS les range par fn quel que
        soit leur retard (osmo-bts trx_sched_ul_burst), et la fenetre du Transmitter (PONT_UL_RETARD_MAX,
        26 trames) en jetait une partie des que l'avance BTS depassait ~23 trames (run de 17:36 : tard=702
        sur 2000). Le canal dedie est relu de force (refresh) quand un burst normal arrive sans TN connu
        -- le cache de 100 ms faisait perdre les premiers bursts du SDCCH, donc le SABM -- et les bursts
        attendent jusqu'a 0,3 s que le TN soit connu."""
        if not UL_RAW:
            return Uplink.run(self)
        ring = RomTxRing()
        n = {0: 0, 1: 0, 2: 0}
        attente = collections.deque()
        perdus = 0
        log.info("PONT_UL_RAW=1 : bursts montants = ceux de la ROM (%s), emis a son fn%s, sans codage ni A5 cote
↪ hote",
                 SB_TX_ROM, (" %+d" % UL_RAW_DECAL) if UL_RAW_DECAL else "")

        def tn_dedie():
            # [2026-10-04, run de 18:19] Le TN est celui ou le FIRMWARE est, d'apres le tap QEMU
            # (calypso_dcch_cfg) : des l'ASSIGNMENT COMMAND il passe sur le TCH et y envoie le SABM
            # FACCH, l'ASSIGNMENT COMPLETE, la parole ; plan() garde le SDCCH « pour un retour » et
            # tch.is_open() n'est vrai qu'apres la preuve cote pont -- les premiers bursts du TCH
            # partaient donc sur le TS du SDCCH et l'assignation echouait (ASSIGNMENT FAILURE cause 1).
            # [2026-10-15] PDCH (GPRS) d abord : les blocs PDTCH montants codes par la ROM
            # (tache DUL du firmware l1gprs_fw.c) partent sur le TN du PDCH, a leur fn.
            pdch = self.dedicated.pdch()
            if pdch is not None:
                return pdch
            tchf = self.dedicated.tch()
            if tchf is None:
                self.dedicated.refresh()
                tchf = self.dedicated.tch()
                pdch = self.dedicated.pdch()
                if pdch is not None:
                    return pdch
            if tchf is not None:
                return tchf[2]
            if self.tch.is_open() and self.tch.active_tn() is not None:
                return self.tch.active_tn()
            ded = self.dedicated.plan()
            if ded is None:
                self.dedicated.refresh()
                ded = self.dedicated.plan()
            # [2026-10-21] Magnetite : pas de tap L1CTL, donc jamais de calypso_dcch_cfg. Ses bursts SDCCH
            # (TSC 7, 4 trames de suite, deja dans calypso_tx_rom avec leur fn) partaient « aucun canal dedie
```

```
            # connu, perdu ». PONT_UL_TN_DEFAUT=<tn> : l'intervalle a prendre faute d'annonce (TS0, SDCCH/4).
            if ded is None and UL_TN_DEFAUT is not None:
                return UL_TN_DEFAUT
            return ded[2] if ded else None

        tch_dit = set()
        tch_rom = [0]

        def tn_tch():
            # TN du TCH tel que le firmware le voit (tap QEMU), sinon celui du pont
            t = self.dedicated.tch()
            return t[2] if t is not None else self.tch.active_tn()

        def envoyer(seq, fn, typ, tsc, bits, tn, a5=False):
            st = self.stats
            est_tch = typ != 1 and tn is not None and tn == tn_tch()
            if est_tch:
                # B5 : retard du fn ROM sur l'horloge BTS (trames), et SACCH/T disponibles
                r = (self.clock.fn() - fn) % gsm.HYPERFRAME
                if r > gsm.HYPERFRAME // 2:
                    r -= gsm.HYPERFRAME
                if st.tch_rom_retard_min is None or r < st.tch_rom_retard_min:
                    st.tch_rom_retard_min = r
                if st.tch_rom_retard_max is None or r > st.tch_rom_retard_max:
                    st.tch_rom_retard_max = r
                if classe_tch(fn, bits) == "S":
                    self._s_rom = getattr(self, "_s_rom", 0) + 1
                    st.sacch_t_rom_blocs = self._s_rom // 4
            if typ != 1 and not UL_RAW_TCH and tn == self.tch.active_tn():
                st.tch_ul_rom_jete[classe_tch(fn, bits)] += 1
                tch_rom[0] += 1          # voir UL_RAW_TCH : le TCH est code et emis par l'hote
                if tch_rom[0] == 1:
                   log.info("bursts ROM du TCH TS%d non emis : FACCH, SACCH/T et parole restent codes par l'hote "
                            "(PONT_UL_RAW_TCH=0), la ROM n'emet pas encore la parole montante", tn)
                return
            self.tx_rom.trx.send_ul(tn, (fn + UL_RAW_DECAL) % gsm.HYPERFRAME, list(bits), False)
            if typ == 1:
                st.air_rom_rach += 1
                ra = gsm.rach_ra(bits, self.cfg.bsic)      # [2026-10-24] voir downlink._imm_ass_dcch
                if ra is not None:
                    from ..downlink import noter_rach
                    noter_rach(ra)
                elif st.air_rom_rach <= 3:
                    log.info("RACH ROM #%d : RA illisible (BSIC %d), IMM ASS choisie sans reference", seq,
↪   self.cfg.bsic)
            elif tn == self.dedicated.pdch():
                st.air_rom_pdch += 1
            elif typ not in (1, 2):
                st.air_rom_inconnu += 1      # B4 : emis quand meme (voisin du PDCH en GPRS)
                if st.air_rom_inconnu <= 5:
                    log.info("burst ROM #%d de type inconnu (%d) : TS%d fn=%u fn%%13=%d fn%%26=%d", seq, typ, tn,
                             fn, fn % 13, fn % 26)
            elif est_tch:
                st.air_rom_tch += 1
                st.tch_ul_rom[classe_tch(fn, bits)] += 1
            else:
                st.air_rom_xcch += 1
            if a5 and typ != 1:
                st.a5_ul_rom += 1
            if typ != 1 and self.dedicated.tch() is not None and tn not in tch_dit:
                tch_dit.add(tn)
                log.info("bursts ROM sur le TCH TS%d : parole, FACCH et SACCH montants codes et chiffres par la
↪   ROM", tn)
            k = typ if typ in n else 0
            n[k] += 1
            total = n[0] + n[1] + n[2]
```

```
            if total <= 5 or (typ == 1 and n[1] <= 5) or total % 1000 == 0:
                log.info("burst ROM #%d %s fn=%u TS%d%s (RACH %d, NB %d, ? %d, attendus %d, perdus %d)", seq,
                         "RACH" if typ == 1 else "NB" if typ == 2 else "?", fn, tn,
                         (" TSC%d" % tsc) if typ == 2 else "", n[1], n[2], n[0], len(attente), perdus)

        while True:
            self._poll_release()
            self._poll_rach()
            if UL_RAW_DCCH_HOTE:
                # [2026-10-21] SDCCH et SACCH codes par l'hote, envoyes pour de vrai (le RACH reste a la ROM)
                puits, self.tx = self.tx, self.tx_rom
                try:
                    self._poll_sdcch()
                    self._poll_sacch()
                finally:
                    self.tx = puits
                self._poll_facch()
            else:
                self._poll_sdcch()
                self._poll_facch()
                self._poll_sacch()
            self._poll_voice()
            if attente:
                tn = tn_dedie()
                now = time.monotonic()
                while attente:
                    seq, fn, typ, tsc, bits, a5, t0 = attente[0]
                    if tn is not None:
                        attente.popleft()
                        envoyer(seq, fn, typ, tsc, bits, tn, a5)
                    elif now - t0 > 0.3:
                        attente.popleft()
                        perdus += 1
                        if perdus <= 5:
                            log.info("burst ROM #%d fn=%u (TSC %d) : aucun canal dedie connu apres 300 ms, perdu",
↪  seq, fn, tsc)
                    else:
                        break
            for seq, fn, typ, tsc, bits, a5 in ring.nouveaux():
                if typ == 1:
                    envoyer(seq, fn, typ, tsc, bits, 0)
                    continue
                tn = tn_dedie()
                if tn is None:
                    attente.append((seq, fn, typ, tsc, bits, a5, time.monotonic()))
                    continue
                envoyer(seq, fn, typ, tsc, bits, tn, a5)
            time.sleep(POLL)
```

3.330 /opt/GSM/osmo-operator/scripts/audio-chain.sh

6488 octets, 129 lignes → 129 lignes

```
#!/usr/bin/env bash
# ============================================================================
#  scripts/audio-chain.sh - ferme la chaine audio locale, une fois pulse debout
# ============================================================================
#  [2026-08-14] POURQUOI CE SCRIPT EXISTE.
#  Trois maillons de la chaine audio n'etaient poses par AUCUN chemin execute :
#
#   1. /etc/asound.conf - absent de la VM au boot. Sans lui les PCM ALSA
#      `gsm_out` / `gsm_in` que `mobile` ouvre n'existent pas, donc la voix TCH
#      n'atteint jamais gsm_audio. Il est bien deploye par ensure_pulse() de
#      lib/audio.sh... mais lib/audio.sh n'est source par PERSONNE (son appelant
```

```
#      annonce, run_modules/25-audio.sh, n'existe pas).
#   2. le sink `gsm_mic` - build-iso.sh n'ecrivait que `gsm_audio` dans
#      system.pa. Or configs/asound.conf fait pointer `pcm.gsm_in` sur
#      `gsm_mic.monitor` : sans ce sink, gapk_io echoue a ouvrir la capture et
#      ABANDONNE LES DEUX SENS (gapk_io.c:468) → appel etabli, TOTALEMENT muet.
#   3. le `module-loopback` gsm_audio.monitor → carte. gsm_audio est un
#      null-sink : sans consommateur la voix descendante est jetee par
#      construction (sortie ALSA SUSPENDED, pcm0p/sub0/status = "closed").
#
#  Ce script pose les trois, dans l'ordre, de facon idempotente.
#
#  Usage : audio-chain.sh [timeout_s]   (defaut 30)
#  Neutralise par AUDIO=0 ; le seul loopback l'est par AUDIO_LOCAL_LOOPBACK=0.
#  Latence du loopback : LOOPBACK_LATENCY_MSEC (defaut 20).
#  Toujours exit 0 - l'audio ne doit jamais empecher la pile de monter.
# ------------------------------------------------------------------------------
set -uo pipefail

# [2026-09-23] --reinit : decharge annuleur, boucles et osmo_rec, puis repose
# la chaine (lib/audio.sh, reinit_audio). Le reste des arguments : inchange
# (delai d attente de PulseAudio, 30 s par defaut).
REINIT=0
if [ "${1:-}" = "--reinit" ]; then REINIT=1; shift; fi
# [2026-09-23] --appel debut|fin : la chaine lourde (annuleur, boucles,
# osmo_rec) seulement pendant un appel. Appele par gapk-start.sh (mode auto)
# sur « Endpoint actif » / « Endpoint disparu ». AUDIO_PENDANT_APPEL=0 : la
# chaine reste posee en permanence, comme avant.
APPEL=""
if [ "${1:-}" = "--appel" ]; then APPEL="${2:-}"; shift 2 2>/dev/null || shift; fi

HERE="${HERE:-$(cd "$(dirname "${BASH_SOURCE[0]}")/.." && pwd)}"
export HERE
: "${PULSE_SOCK:=/var/run/pulse/native}"
export PULSE_SERVER="unix:${PULSE_SOCK}"

# shellcheck source=../lib/audio.sh
. "${HERE}/lib/audio.sh"

[ "${AUDIO:-1}" = "1" ] || { echo "[audio] desactive (AUDIO=0)"; exit 0; }

# ── 1. Le mapping ALSA, AVANT tout le reste ──────────────────────────────
# `mobile` ouvre gsm_out/gsm_in : si /etc/asound.conf manque, il tombe sur
# "Unknown PCM gsm_out" et rien ne remonte jamais dans gsm_audio.
if [ -f "${HERE}/configs/asound.conf" ] \
   && ! cmp -s "${HERE}/configs/asound.conf" /etc/asound.conf 2>/dev/null; then
    if cp -f "${HERE}/configs/asound.conf" /etc/asound.conf 2>/dev/null; then
        echo "[audio] /etc/asound.conf deploye (PCM gsm_out/gsm_in)"
    else
        echo "[audio] /etc/asound.conf NON deploye (droits ?)" >&2
    fi
fi

# ── 2. Attendre le demon ─────────────────────────────────────────────────
deadline="${1:-30}"
while [ "$deadline" -gt 0 ] && ! pactl info >/dev/null 2>&1; do
    sleep 1
    deadline=$(( deadline - 1 ))
done
if ! pactl info >/dev/null 2>&1; then
    echo "[audio] PulseAudio injoignable @ ${PULSE_SOCK} - chaine non fermee" >&2
    exit 0
fi

# ── 2 bis. Attendre une SORTIE MATERIELLE, pas seulement le demon ────────
# [2026-08-31] Au premier demarrage d'un disque fraichement installe, ce script
# concluait "aucune sortie materielle - loopback local ignore" et rendait la
```

```
# main : l'appel montait, et il etait MUET. Le demon repondait pourtant deja -
# c'est bien lui qu'attend l'etape 2 - mais les modules snd_hda_* finissaient
# de se charger, et module-udev-detect n'avait encore declare aucun sink ALSA.
# La cle live n'en souffrait pas : elle demarre plus lentement, la course se
# jouait dans l'autre sens, et le bug ne se voyait qu'apres installation.
#
# module-udev-detect rattrape les cartes qui arrivent apres coup, mais
# ensure_local_loopback, lui, ne passe qu'UNE fois : s'il passe trop tot, le
# loopback n'est jamais pose et plus rien ne le posera. On lui laisse donc la
# meme fenetre qu'au demon. Non bloquant au-dela : sans carte du tout (un banc
# sans son, une VM sans peripherique audio), on continue - les null-sinks
# suffisent a gapk_io, seule l'ecoute locale manque.
hw_deadline="${1:-30}"
while [ "$hw_deadline" -gt 0 ] \
      && ! pactl list short sinks 2>/dev/null | grep -q 'module-alsa-card'; do
    sleep 1
    hw_deadline=$(( hw_deadline - 1 ))
done

# ── 3. Les DEUX null-sinks, puis le loopback ──────────────────────────
# load_gsm_sinks() boucle sur GSM_SINKS = gsm_audio + gsm_mic : c'est ce qui
# rattrape un system.pa qui n'en declarerait qu'un.
[ "$REINIT" = 1 ] && reinit_audio
load_gsm_sinks
for s in gsm_audio gsm_mic; do
    if pactl list short sinks 2>/dev/null | grep -qw "$s"; then
        echo "[audio] sink $s present"
    else
        echo "[audio] sink $s MANQUANT - gapk_io va abandonner les deux sens" >&2
    fi
done

# [2026-09-09] L annuleur d echo d abord : les HP de l operateur sont
# osmo_hp_ec des le boot, comme avec la VM (lib/audio.sh, « L AFFECTATION DES
# HAUT-PARLEURS »). Sans VM, le bouclage direct suit ; avec, il est retire.
alleger_audio
if [ "${AUDIO_PENDANT_APPEL:-0}" = "1" ] && ! pmos_vm_audio_present; then
    case "$APPEL" in
    debut) ;;                                  # on pose la chaine, plus bas
    fin)   reinit_audio; echo "[audio] fin d appel : chaine retiree"; exit 0 ;;
    *)     # au start / au boot : un banc neuf n a pas d appel en cours ; le
           # premier « Endpoint actif » de gapk-auto posera la chaine.
           reinit_audio
           echo "[audio] chaine posee a la demande, au premier appel (AUDIO_PENDANT_APPEL=0 pour la garder)"
           exit 0 ;;
    esac
fi
ensure_echo_cancel
ensure_local_loopback
ensure_local_mic
ensure_record_mix
exit 0
```

3.331 /opt/GSM/osmo-operator/scripts/audio-diag.sh

8325 octets, 177 lignes → 177 lignes

```
#!/bin/bash
# audio-diag.sh - Diagnostic audio complet dans le container
# Usage : docker exec -ti osmo-operator-1 bash /etc/osmocom/audio-diag.sh
#
# Verifie toute la chaine : /dev/snd → ALSA → PulseAudio → mobile l1phy

set -euo pipefail
```

```
GREEN='\033[0;32m'; RED='\033[0;31m'; YELLOW='\033[1;33m'
CYAN='\033[0;36m'; NC='\033[0m'; BOLD='\033[1m'

ok()   { echo -e "  ${GREEN}✓${NC} $*"; }
fail() { echo -e "  ${RED}✗${NC} $*"; }
warn() { echo -e "  ${YELLOW}⚠${NC} $*"; }

echo -e "${CYAN}${BOLD}═══════════════════════════════════════════════${NC}"
echo -e "${CYAN}${BOLD}  Diagnostic audio - container $(hostname)${NC}"
echo -e "${CYAN}${BOLD}═══════════════════════════════════════════════${NC}"
echo ""

# ── 1. /dev/snd ──────────────────────────────────────────
echo -e "${BOLD}[1/7] /dev/snd${NC}"
if [ -d /dev/snd ]; then
    ok "/dev/snd present"
    ls -la /dev/snd/ 2>/dev/null | sed 's/^/       /'
else
    fail "/dev/snd ABSENT - le container n'a pas --device /dev/snd"
    echo "       Fix: ajouter --device /dev/snd au docker run"
fi
echo ""

# ── 2. ALSA ──────────────────────────────────────────────
echo -e "${BOLD}[2/7] ALSA${NC}"
if command -v aplay >/dev/null 2>&1; then
    ok "aplay disponible"
    echo "       Cartes:"
    cat /proc/asound/cards 2>/dev/null | sed 's/^/       /' || fail "pas de /proc/asound/cards"
    echo ""
    echo "       Devices playback:"
    aplay -l 2>/dev/null | grep "^card" | sed 's/^/       /' || warn "aucun device playback"
    echo ""
    echo "       Devices capture:"
    arecord -l 2>/dev/null | grep "^card" | sed 's/^/       /' || warn "aucun device capture"
else
    fail "aplay non installe"
fi
echo ""

# ── 3. PulseAudio ────────────────────────────────────────
echo -e "${BOLD}[3/7] PulseAudio${NC}"
if [ -n "${PULSE_SERVER:-}" ]; then
    ok "PULSE_SERVER=${PULSE_SERVER}"
    # Tester la connexion
    if command -v pactl >/dev/null 2>&1; then
        if pactl info >/dev/null 2>&1; then
            ok "PulseAudio accessible"
            pactl info 2>/dev/null | grep -E "Server|Default Sink|Default Source" | sed 's/^/       /'
        else
            fail "PulseAudio inaccessible (socket monte mais serveur non joignable)"
        fi
    else
        warn "pactl non installe - impossible de verifier PulseAudio"
        echo "       Le socket est monte mais on ne peut pas tester sans pactl"
        if [ -S "${PULSE_SERVER#unix:}" ] 2>/dev/null; then
            ok "Socket PulseAudio existe : ${PULSE_SERVER#unix:}"
        else
            fail "Socket PulseAudio introuvable"
        fi
    fi
else
    warn "PULSE_SERVER non defini"
    echo "       Sans PulseAudio, ALSA utilise directement hw:X,Y"
    echo "       Sur un desktop Linux moderne, le son passe par PulseAudio/PipeWire"
```

```
fi
echo ""

# ── 4. Variables ALSA ──────────────────────────────────────────────
echo -e "${BOLD}[4/7] Variables d'environnement${NC}"
echo "       ALSA_CARD     = ${ALSA_CARD:-<non defini>}"
echo "       GAPK_ALSA_DEV = ${GAPK_ALSA_DEV:-<non defini>}"
echo "       ALSA_OUTPUT   = ${ALSA_OUTPUT:-<non defini>}"
echo "       ALSA_INPUT    = ${ALSA_INPUT:-<non defini>}"
echo "       PULSE_SERVER  = ${PULSE_SERVER:-<non defini>}"
echo ""

# ── 5. mobile.cfg tch-voice ────────────────────────────────────────
echo -e "${BOLD}[5/7] mobile.cfg tch-voice${NC}"
for cfg in /root/.osmocom/bb/mobile_group*.cfg /root/.osmocom/bb/mobile.cfg; do
    [ -f "$cfg" ] || continue
    echo "       ${cfg}:"
    if grep -q "io-handler l1phy" "$cfg" 2>/dev/null; then
        ok "io-handler l1phy"
        grep -E "alsa-(output|input)-dev" "$cfg" 2>/dev/null | sed 's/^/       /'
    elif grep -q "io-handler gapk" "$cfg" 2>/dev/null; then
        warn "io-handler gapk (audio via osmo-gapk, pas ALSA direct)"
    else
        warn "pas de bloc tch-voice - pas d'audio MS"
    fi
    break
done
echo ""

# ── 6. Test ALSA rapide ────────────────────────────────────────────
echo -e "${BOLD}[6/7] Test ALSA (bip 440Hz 0.2s)${NC}"
if [ -d /dev/snd ]; then
    if command -v speaker-test >/dev/null 2>&1; then
        echo "       Tentative speaker-test (2 secondes max)..."
        timeout 2 speaker-test -t sine -f 440 -l 1 -P 1 2>&1 | head -5 | sed 's/^/       /' || warn "speaker-test
        ↪  echoue"
    elif command -v aplay >/dev/null 2>&1; then
        # Generer un bip court
        python3 -c "
import struct, sys
rate=8000; dur=0.2; freq=440
samples = [int(32767 * __import__('math').sin(2*3.14159*freq*t/rate)) for t in range(int(rate*dur))]
header = struct.pack('<4sI4s4sIHHIIHH4sI', b'RIFF', 36+len(samples)*2, b'WAVE', b'fmt ', 16, 1, 1, rate, rate*2,
↪  2, 16, b'data', len(samples)*2)
sys.stdout.buffer.write(header + struct.pack('<%dh'%len(samples), *samples))
" 2>/dev/null | aplay -q 2>/dev/null && ok "Bip joue sur ALSA" || warn "aplay a echoue"
    fi
else
    fail "Pas de /dev/snd - test impossible"
fi
echo ""

# ── 7. Processus audio ─────────────────────────────────────────────
echo -e "${BOLD}[7/7] Processus audio actifs${NC}"
if pgrep -fa "mobile.*mobile" >/dev/null 2>&1; then
    ok "Processus mobile actif"
    pgrep -fa "mobile" 2>/dev/null | head -3 | sed 's/^/       /'
else
    warn "Processus mobile non detecte (pas encore demarre ?)"
fi
if pgrep -fa "osmo-gapk" >/dev/null 2>&1; then
    ok "osmo-gapk actif"
    pgrep -fa "osmo-gapk" 2>/dev/null | head -3 | sed 's/^/       /'
else
    echo "       osmo-gapk non actif (normal si l1phy utilise)"
fi
```

```
echo ""

# ── Resume ─────────────────────────────────────────────────────────
echo -e "${BOLD}═══ Resume ═══${NC}"
ISSUES=0

if [ ! -d /dev/snd ]; then
    fail "BLOQUANT: /dev/snd absent → pas d'audio possible"
    echo "       Fix: dans start-docker.sh, verifier que --device /dev/snd est dans le docker run"
    ISSUES=$((ISSUES+1))
fi

if [ -z "${PULSE_SERVER:-}" ] && [ -d /dev/snd ]; then
    warn "PulseAudio non configure"
    echo "       Sur un desktop moderne, le device 'default' ALSA necessite PulseAudio"
    echo "       Fix 1: monter le socket PulseAudio dans le container :"
    echo "         -v /run/user/\$(id -u)/pulse/native:/run/user/1000/pulse/native"
    echo "         -e PULSE_SERVER=unix:/run/user/1000/pulse/native"
    echo "       Fix 2: utiliser un device ALSA direct (bypass PulseAudio) :"
    echo "         Trouver le device: aplay -l"
    echo "         Puis: ALSA_OUTPUT=hw:0,0 ALSA_INPUT=hw:0,0 dans start-docker.sh"
    ISSUES=$((ISSUES+1))
fi

if [ -n "${PULSE_SERVER:-}" ]; then
    pulse_sock="${PULSE_SERVER#unix:}"
    if [ ! -S "$pulse_sock" ] 2>/dev/null; then
        fail "Socket PulseAudio monte mais fichier inexistant: $pulse_sock"
        echo "       Le socket PulseAudio de l'hote n'est pas accessible dans le container"
        echo "       Verifier le -v dans le docker run"
        ISSUES=$((ISSUES+1))
    fi
fi

if [ $ISSUES -eq 0 ]; then
    echo -e "  ${GREEN}${BOLD}Audio devrait fonctionner${NC}"
    echo "       Tester: depuis le VTY mobile → call 1 600"
    echo "       Le son sort sur les HP de l'hote via /dev/snd"
fi
echo ""
```

3.332 /opt/GSM/osmo-operator/scripts/banc-repro.sh

8283 octets, 171 lignes → 171 lignes

```
#!/bin/bash
# ============================================================================
#  banc-repro.sh — UNE EXPERIENCE REPRODUCTIBLE, DU DEMARRAGE AU VERDICT
# ============================================================================
#  POURQUOI CE SCRIPT EXISTE. Le defaut qu'on chasse est INTERMITTENT : sur une
#  serie d'appels, certains aboutissent et d'autres non. Diagnostiquer ca demande
#  de rejouer la meme sequence apres chaque changement, et de comparer des
#  journaux comparables. Fait a la main, chaque iteration coute plusieurs minutes
#  et produit des runs qui ne se comparent pas (nombre d'appels different, delais
#  differents, journaux qui se melangent avec les precedents).
#
#  Ce script fige la sequence : demarrage detache, attente du camp, N appels
#  identiques, collecte horodatee, verdict par appel.
#
#  IL NE CORRIGE RIEN ET NE DEVINE RIEN. Il produit une mesure.
#
#  Usage :
#     scripts/banc-repro.sh                    3 appels, avec redemarrage
#     scripts/banc-repro.sh --calls 5          5 appels
```

```
#     scripts/banc-repro.sh --no-restart         reutilise la pile en cours
#     scripts/banc-repro.sh --dest 600           numero appele (defaut 600)
#
#  Sortie : un repertoire /root/repro-<horodatage>/ contenant tous les journaux
#  du run et un resume.txt. Rien n'est ecrase d'un run a l'autre.
# ------------------------------------------------------------------------------
set -uo pipefail

HERE="$(cd "$(dirname "${BASH_SOURCE[0]}")/.." && pwd)"
CALLS=3
DEST=600
RESTART=1
VTY=4247                # MS#1 ; MS#2 est sur 4248
CALL_S=20               # duree d'un appel
GAP_S=10                # silence entre deux appels (le canal doit se liberer)
BOOT_MAX=180            # attente maximale du camp

while [ $# -gt 0 ]; do
    case "$1" in
        --calls)      CALLS="${2:?}"; shift ;;
        --dest)       DEST="${2:?}"; shift ;;
        --vty)        VTY="${2:?}"; shift ;;
        --call-s)     CALL_S="${2:?}"; shift ;;
        --gap-s)      GAP_S="${2:?}"; shift ;;
        --no-restart) RESTART=0 ;;
        -h|--help)    sed -n '2,30p' "$0" | sed 's/^# \{0,1\}//'; exit 0 ;;
        *) echo "option inconnue : $1" >&2; exit 2 ;;
    esac
    shift
done

STAMP="$(date +%Y%m%d-%H%M%S)"
OUT="/root/repro-$STAMP"
mkdir -p "$OUT"
LOGDIR=/run/user/0/osmo-nitb/logs

say() { printf '  %s\n' "$*"; }

# -- VTY : une commande, une reponse ------------------------------------------
# `enable` puis la commande. On laisse une seconde au demon pour repondre : nc
# ferme des que stdin se termine, et sans cette pause la commande part mais la
# reponse n'est jamais lue (symptome : "aucune sortie" sur une commande qui a
# pourtant bien ete executee).
vty() {
    { printf 'enable\n%s\n' "$1"; sleep 1; } | timeout 6 nc 127.0.0.1 "$VTY" 2>/dev/null
}

# -- 1. Demarrage -------------------------------------------------------------
# CALYPSO_NO_ATTACH=1 : run.sh ne s'attache pas a tmux a la fin (run.sh:262).
# Sans ca le script resterait bloque sur une session interactive.
if [ "$RESTART" = 1 ]; then
    say "arret de la pile en cours"
    ( cd "$HERE" && CALYPSO_NO_ATTACH=1 ./start-native.sh --stop ) >"$OUT/stop.log" 2>&1
    sleep 3
    say "demarrage detache (CALYPSO_NO_ATTACH=1)"
    ( cd "$HERE" && CALYPSO_NO_ATTACH=1 setsid ./start-native.sh ) >"$OUT/start.log" 2>&1 &
else
    say "pile en cours reutilisee (--no-restart)"
fi

# -- 2. Attendre le camp, ne pas le supposer ----------------------------------
# On interroge le VTY plutot que d'attendre un delai fixe : selon la machine, le
# camp arrive en 40 s ou en 2 min, et un sleep fige donne soit du gaspillage soit
# des appels lances avant que le mobile ne soit pret — donc des "echecs" qui ne
# mesurent que notre impatience.
say "attente du camp (C3), maximum ${BOOT_MAX}s"
```

```
camp=0
for i in $(seq 1 "$BOOT_MAX"); do
    if vty "show ms" 2>/dev/null | grep -q "C3 camped normally"; then camp=1; break; fi
    sleep 1
done
if [ "$camp" != 1 ]; then
    say "PAS DE CAMP apres ${BOOT_MAX}s — on s'arrete, les appels ne mesureraient rien"
    vty "show ms" > "$OUT/show-ms-echec.txt" 2>&1
    cp "$LOGDIR"/*.log "$OUT/" 2>/dev/null
    exit 1
fi
say "campe apres ${i}s"

# Marqueur : tout ce qui suit dans pont.log appartient a l'experience.
PONT_DEBUT=$(wc -l < /dev/shm/pont.log 2>/dev/null || echo 0)

# ── 3. Les appels ──────────────────────────────────────────────────
: > "$OUT/appels.txt"
for n in $(seq 1 "$CALLS"); do
    say "appel $n/$CALLS  -> $DEST"
    avant=$(wc -l < /dev/shm/pont.log 2>/dev/null || echo 0)
    vty "call 1 $DEST" > "$OUT/appel-$n-vty.txt" 2>&1
    sleep "$CALL_S"
    vty "call 1 hangup" >> "$OUT/appel-$n-vty.txt" 2>&1
    sleep 2

    # Verdict : dans la tranche de pont.log produite par CET appel, un
    # "DESARME (CHANNEL RELEASE" signe un TCH qui a vecu ; un "SANS ASSIGNMENT
    # COMPLETE" signe un mobile qui n'a pas bascule. On ne conclut pas d'apres
    # les compteurs cumulatifs, qui melangeraient les appels precedents.
    tranche=$(tail -n +$((avant + 1)) /dev/shm/pont.log 2>/dev/null)
    if printf '%s' "$tranche" | grep -q "SANS ASSIGNMENT COMPLETE"; then
        verdict=ECHEC
    elif printf '%s' "$tranche" | grep -q "DESARME (CHANNEL RELEASE"; then
        verdict=OK
    else
        verdict=INDETERMINE
    fi
    printf 'appel %d : %s\n' "$n" "$verdict" | tee -a "$OUT/appels.txt"
    printf '%s' "$tranche" > "$OUT/appel-$n-pont.log"
    sleep "$GAP_S"
done

# ── 4. Collecte ────────────────────────────────────────────────────
say "collecte des journaux"
cp /dev/shm/pont.log            "$OUT/pont.log"            2>/dev/null
tail -n +$((PONT_DEBUT + 1)) /dev/shm/pont.log > "$OUT/pont-experience.log" 2>/dev/null
cp "$LOGDIR"/qemu.log           "$OUT/qemu.log"            2>/dev/null
cp "$LOGDIR"/qemu-tete.log      "$OUT/qemu-tete.log"       2>/dev/null
cp "$LOGDIR"/mobile.log         "$OUT/mobile.log"          2>/dev/null
journalctl -u osmo-bsc --since "-30min" --no-pager > "$OUT/osmo-bsc.log" 2>/dev/null
journalctl -u osmo-msc --since "-30min" --no-pager > "$OUT/osmo-msc.log" 2>/dev/null
for f in calypso_dcch_cfg calypso_kc_l1 calypso_tch_cfg; do
    [ -e "/dev/shm/$f" ] && od -A d -t x1 "/dev/shm/$f" > "$OUT/$f.hex" 2>/dev/null
done

# ── 5. Resume ──────────────────────────────────────────────────────
# Les compteurs de pont.py (STATS, DETAIL blocs) sont de VRAIS cumuls. Ceux du
# shunt sont des SHUNT_LOG limites en debit (ex. `if (++n <= 3 || n % 2000 == 0)`)
# : les compter donnerait un chiffre qui ne mesure rien. On ne met donc ici que
# ce qui est fiable, et on le dit.
{
    echo "=== experience $STAMP : $CALLS appels vers $DEST ==="
    cat "$OUT/appels.txt"
    echo
    echo "--- canaux dedies appris par le shunt (compteur reel, non limite) ---"
```

```
    grep -ac "DCCH #" "$OUT/qemu.log" 2>/dev/null
    grep -ah "DCCH #" "$OUT/qemu.log" 2>/dev/null | tail -5
    echo
    echo "--- pont : compteurs cumulatifs (FIABLES) ---"
    grep -a "STATS" "$OUT/pont.log" 2>/dev/null | tail -1
    grep -a "DETAIL blocs" "$OUT/pont.log" 2>/dev/null | tail -1
    grep -a "SOUS-VOIE ACTIVE" "$OUT/pont.log" 2>/dev/null | tail -1
    echo
    echo "--- reseau : echecs d'assignation ---"
    grep -ac "Assignment failed" "$OUT/osmo-bsc.log" 2>/dev/null
    echo
    echo "--- mobile : SAPI invalides ---"
    grep -ac "unsupported SAPI" "$OUT/mobile.log" 2>/dev/null
    echo
    echo "ATTENTION : les SHUNT_LOG du shunt sont limites en debit."
    echo "Ne pas utiliser 'grep -c' sur eux comme un compteur."
} > "$OUT/resume.txt" 2>&1

cat "$OUT/resume.txt"
say "tout est dans $OUT"
```

3.333 /opt/GSM/osmo-operator/scripts/call-diag.sh

7409 octets, 183 lignes → 183 lignes

```
#!/bin/bash
# call-diag.sh - Diagnostic chemin d'appel MS → Asterisk echo test
# Usage: sudo docker exec -ti osmo-operator-1 bash /scripts/call-diag.sh
#
# Verifie chaque maillon : MSC → MNCC → osmo-sip-connector → Asterisk

GREEN='\033[0;32m'; RED='\033[0;31m'; YELLOW='\033[1;33m'
CYAN='\033[0;36m'; NC='\033[0m'; BOLD='\033[1m'

ok()   { echo -e "  ${GREEN}✓${NC} $*"; }
fail() { echo -e "  ${RED}✗${NC} $*"; }
warn() { echo -e "  ${YELLOW}⚠${NC} $*"; }

echo -e "${CYAN}${BOLD}═══ Diagnostic appel MS → Echo Test ═══${NC}"
echo ""

# ── 1. Services core ──────────────────────────────────────────
echo -e "${BOLD}[1/8] Services Osmocom${NC}"
for svc in osmo-msc osmo-bsc osmo-mgw osmo-stp osmo-hlr; do
    if systemctl is-active --quiet "$svc" 2>/dev/null; then
        ok "$svc actif"
    elif pgrep -f "$svc" >/dev/null 2>&1; then
        ok "$svc (process)"
    else
        fail "$svc INACTIF"
    fi
done
echo ""

# ── 2. MNCC socket ────────────────────────────────────────────
echo -e "${BOLD}[2/8] MNCC socket (/tmp/msc_mncc)${NC}"
if [ -S /tmp/msc_mncc ]; then
    ok "Socket MNCC existe"
    ls -la /tmp/msc_mncc | sed 's/^/       /'
else
    fail "Socket MNCC ABSENT - MSC ne cree pas le socket MNCC external"
    echo "       Verifier que osmo-msc.cfg contient : mncc external /tmp/msc_mncc"
fi
echo ""
```

```
# ── 3. osmo-sip-connector ──────────────────────────────────────────
echo -e "${BOLD}[3/8] osmo-sip-connector${NC}"
if pgrep -f "osmo-sip-connector" >/dev/null 2>&1; then
    ok "Processus actif"
    pgrep -fa "osmo-sip-connector" | head -1 | sed 's/^/       /'

    # Verifier qu'il ecoute bien sur 5061
    if ss -tlnp 2>/dev/null | grep -q ":5061 "; then
        ok "Ecoute sur :5061 (cote SIP)"
    else
        warn "Pas en ecoute sur :5061"
    fi
else
    fail "osmo-sip-connector NON LANCE"
    echo ""
    echo "       C'est le pont MNCC ↔ SIP. Sans lui, aucun appel ne passe."
    echo "       Fix: systemctl start osmo-sip-connector"
    echo "       Ou:  osmo-sip-connector -c /etc/osmocom/osmo-sip-connector.cfg &"
fi
echo ""

# ── 4. Asterisk ────────────────────────────────────────────────────
echo -e "${BOLD}[4/8] Asterisk${NC}"
if pgrep -f "asterisk" >/dev/null 2>&1; then
    ok "Processus actif"

    if ss -ulnp 2>/dev/null | grep -q ":5060 "; then
        ok "Ecoute SIP sur :5060/udp"
    else
        fail "Pas en ecoute sur :5060/udp"
    fi

    # Verifier les endpoints PJSIP
    echo "       Endpoints PJSIP:"
    asterisk -rx "pjsip show endpoints" 2>/dev/null | grep -E "Endpoint:|Contact:" | head -10 | sed 's/^/       /'
     ↪  || warn "Impossible de lister les endpoints"

    # Verifier que le dialplan a le 600
    echo ""
    echo "       Dialplan 600:"
    asterisk -rx "dialplan show 600@gsm_in" 2>/dev/null | head -10 | sed 's/^/       /' || warn "Extension 600
     ↪  introuvable dans gsm_in"
else
    fail "Asterisk NON LANCE"
    echo "       Fix: asterisk -cvvv &"
fi
echo ""

# ── 5. osmo-sip-connector config ──────────────────────────────────
echo -e "${BOLD}[5/8] Config osmo-sip-connector${NC}"
SIP_CFG="/etc/osmocom/osmo-sip-connector.cfg"
if [ -f "$SIP_CFG" ]; then
    echo "       Contenu:"
    cat "$SIP_CFG" | sed 's/^/       /'

    remote_line=$(grep "remote" "$SIP_CFG" | head -1)
    echo ""
    if echo "$remote_line" | grep -q "5060"; then
        ok "Remote SIP pointe vers :5060"
    else
        warn "Remote SIP: $remote_line"
    fi
else
    fail "Config osmo-sip-connector introuvable"
fi
```

```
echo ""

# ── 6. MGW ──────────────────────────────────────────────────────────
echo -e "${BOLD}[6/8] OsmoMGW (RTP)${NC}"
if bash -c "echo >/dev/tcp/127.0.0.1/4243" 2>/dev/null; then
    ok "VTY 4243 accessible"
    echo "       Endpoints:"
    (echo "show mgcp"; sleep 1) | telnet 127.0.0.1 4243 2>/dev/null \
        | grep -vE "^(Trying|Connected|Escape|Welcome|OsmoMGW)" \
        | head -15 | sed 's/^/       /' || true
else
    fail "VTY MGW 4243 inaccessible"
fi
echo ""

# ── 7. BTS-TRX ──────────────────────────────────────────────────────
echo -e "${BOLD}[7/8] osmo-bts-trx${NC}"
if pgrep -f "osmo-bts-trx" >/dev/null 2>&1; then
    ok "Processus actif"
else
    if systemctl is-active --quiet osmo-bts-trx 2>/dev/null; then
        ok "Service actif"
    else
        fail "osmo-bts-trx INACTIF"
        echo "       Sans BTS, pas de canal radio → pas d'appel"
        echo "       Fix: systemctl start osmo-bts-trx"
    fi
fi
echo ""

# ── 8. Mobile registration ──────────────────────────────────────────
echo -e "${BOLD}[8/8] Enregistrement mobile (MSC VTY)${NC}"
MSC_OUT=$( (echo "enable"; echo "show subscriber all"; sleep 1; echo "exit") \
    | telnet 127.0.0.1 4254 2>/dev/null \
    | grep -vE "^(Trying|Connected|Escape|Welcome|OsmoMSC)" || true)

if echo "$MSC_OUT" | grep -qiE "IMSI|subscriber"; then
    ok "Abonnes enregistres sur le MSC:"
    echo "$MSC_OUT" | grep -iE "IMSI|MSISDN|LAC" | head -10 | sed 's/^/       /'
else
    warn "Aucun abonne enregistre sur le MSC"
    echo "       Les MS ne sont pas encore attaches au reseau"
    echo "       Verifier : dans le VTY mobile → show ms"
    echo "       L'etat doit etre 'NORMAL service'"
fi
echo ""

# ── Resume ──────────────────────────────────────────────────────────
echo -e "${BOLD}═══ Chaine d'appel echo test ═══${NC}"
echo ""
echo "  mobile (call 1 600)"
echo "    ↓ TCH via fake_trx"
echo "  osmo-bts-trx → osmo-bsc → osmo-msc"
echo "    ↓ MNCC socket /tmp/msc_mncc"

if [ -S /tmp/msc_mncc ]; then
    echo -e "    ${GREEN}✓${NC} MNCC socket"
else
    echo -e "    ${RED}✗ MNCC socket MANQUANT${NC}"
fi

echo "  osmo-sip-connector (5061 → 5060)"
if pgrep -f "osmo-sip-connector" >/dev/null 2>&1; then
    echo -e "    ${GREEN}✓${NC} sip-connector"
else
    echo -e "    ${RED}✗ sip-connector MANQUANT ← PROBLEME PROBABLE${NC}"
```

```
fi

echo "  Asterisk (gsm_in context, exten 600 → Echo())"
if pgrep -f "asterisk" >/dev/null 2>&1; then
    echo -e "    ${GREEN}✓${NC} Asterisk"
else
    echo -e "    ${RED}✗ Asterisk MANQUANT${NC}"
fi

echo "    ↓ RTP via OsmoMGW"
echo "  osmo-bts-trx → fake_trx → trxcon → mobile"
echo "    ↓ l1phy decode TCH-FR"
echo "  ALSA → PulseAudio → HP"
echo ""
```

3.334 /opt/GSM/osmo-operator/scripts/calypso-mode.sh

7337 octets, 149 lignes → 149 lignes

```
#!/bin/bash
# ============================================================================
# scripts/calypso-mode.sh - a qui est le Calypso (QEMU + ROM TI) : voix ou data.
# ============================================================================
#
# La couche 1 du Calypso (osmocon, /tmp/osmocom_l2) n accepte qu UN client de
# couche 2. Deux clients se la volent : chacun la remet a zero (RESET FULL),
# le mobile perd le reseau (« No service ») et le modem ne s attache pas.
#
#   voix   le `mobile` osmocom-bb (MS#1, VTY 4247) : appels, SMS, USSD.
#   data   le `modem` osmocom-bb GPRS (IMSI du mobile, 001010001000001) : attach GPRS,
#          contexte PDP, tun_modem dans le netns msgprs. Le mobile est arrete.
#
# Le passage data -> voix relance le mobile EXACTEMENT comme c54x_exe/run.sh
# l avait lance (ligne de commande, repertoire et journal releves dans /proc
# avant de l arreter). Appele par tools/osmo-phonesim-banc.py : ATD*99 du
# telephone -> data ; appel ou SMS -> voix (une 2G de classe B : pas de data
# pendant la voix). Un appel ENTRANT ne peut pas arriver en mode data : le
# mobile n est pas la pour etre pagine.
#
# Usage : calypso-mode.sh data|voix|etat [--wait]
#   --wait   data : attendre l adresse de tun_modem (PDP actif, 240 s max)
#            voix : attendre le mobile en service normal (120 s max)
# Etat : /run/osmo-banc/calypso-mode (voix|data), ligne du mobile :
#        /run/osmo-banc/calypso-mobile.{cmd,cwd,log}
# ----------------------------------------------------------------------------
set -uo pipefail
HERE="$(cd "$(dirname "${BASH_SOURCE[0]}")/.." && pwd)"
ETC="${OSMOCOM_CFG:-/etc/osmocom}"
SOCK="${CALYPSO_L2_SOCK:-/tmp/osmocom_l2}"
NETNS="${EGPRS_NETNS:-msgprs}"
ETAT=/run/osmo-banc
MODE_F="$ETAT/calypso-mode"
U_MODEM=osmo-modem-gprs
MOB_VTY="${OSMO_MOB_VTY_PORT:-4247}"
MODEM_BIN="$(command -v modem 2>/dev/null || echo /opt/GSM/osmocom-bb/src/host/layer23/src/modem/modem)"
mkdir -p "$ETAT"

log() { printf '[calypso-mode] %s\n' "$*"; }
mode_courant() { cat "$MODE_F" 2>/dev/null || echo voix; }

# Le fichier de config d un client de couche 2 (mobile ou modem) vise-t-il SOCK ?
# Sans « layer2-socket », osmocom-bb prend /tmp/osmocom_l2.
vise_sock() {
    local cfg="$1" s
```

```
    s="$(awk '$1=="layer2-socket"{print $2; exit}' "$cfg" 2>/dev/null)"
    [ "${s:-/tmp/osmocom_l2}" = "$SOCK" ]
}
# Les PID des `mobile` branches sur SOCK
mobiles_sur_sock() {
    local p c
    for p in $(pgrep -x mobile); do
        c="$(tr '\0' '\n' < "/proc/$p/cmdline" 2>/dev/null | awk 'p{print; exit} $0=="-c"{p=1}')"
        [ -n "$c" ] || continue
        case "$c" in /*) ;; *) c="$(readlink "/proc/$p/cwd")/$c" ;; esac
        vise_sock "$c" && echo "$p"
    done
}
port_open() { timeout 1 bash -c "exec 3<>/dev/tcp/127.0.0.1/$1" 2>/dev/null; }
vty() {
    { sleep 0.3; printf 'enable\n'; local c; for c in "$@"; do sleep 0.3; printf '%s\n' "$c"; done; sleep 0.6; } \
        | timeout 6 nc -q1 127.0.0.1 "$MOB_VTY" 2>/dev/null | tr -d '\r'
}

vers_data() {
    local p n=0
    for p in $(mobiles_sur_sock); do
        # La ligne exacte du mobile, pour le relancer a l identique
        tr '\0' '\n' < "/proc/$p/cmdline" > "$ETAT/calypso-mobile.cmd"
        readlink "/proc/$p/cwd" > "$ETAT/calypso-mobile.cwd"
        readlink "/proc/$p/fd/1" > "$ETAT/calypso-mobile.log" 2>/dev/null || true
        # [2026-10-15] SIGTERM ne suffit pas : le mobile tente un detach IMSI
        # propre, qu il n acheve pas sur une couche 1 disputee (vu : mobile
        # vivant 2 min apres le SIGTERM, a cote du modem). 3 s, puis SIGKILL.
        kill "$p" 2>/dev/null
        local w; for w in 1 2 3 4 5 6; do kill -0 "$p" 2>/dev/null || break; sleep 0.5; done
        kill -0 "$p" 2>/dev/null && kill -9 "$p" 2>/dev/null && log "mobile $p : SIGTERM ignore, SIGKILL"
        n=$((n+1)); log "mobile $p arrete (ligne gardee dans $ETAT/calypso-mobile.cmd)"
    done
    [ "$n" -gt 0 ] && sleep 1
    ip netns list 2>/dev/null | grep -qw "$NETNS" || ip netns add "$NETNS"
    local cfg="$ETC/modem-gprs-dsp.cfg"
    sed "s|layer2-socket .*|layer2-socket $SOCK|" "$HERE/configs/gprs/modem-gprs.cfg" > "$cfg"
    if ! systemctl is-active --quiet "$U_MODEM"; then
        systemctl reset-failed "$U_MODEM" >/dev/null 2>&1 || true
        systemd-run --quiet --unit="$U_MODEM" --collect -p KillMode=control-group -p
         ↪  CPUAffinity="${RT_CPUS:-0-15}" -p Nice=-10 \
            -p SyslogIdentifier="$U_MODEM" "$MODEM_BIN" -c "$cfg" \
            || { log "systemd-run $U_MODEM a echoue"; return 1; }
        log "modem GPRS lance sur $SOCK ($cfg)"
    fi
    echo data > "$MODE_F"
    if [ "$ATTENDRE" = 1 ]; then
        local t=0
        until ip netns exec "$NETNS" ip -4 -o addr show dev tun_modem 2>/dev/null | grep -q 'inet '; do
            [ $t -ge 240 ] && { log "pas de PDP apres 240 s (journalctl -u $U_MODEM)"; return 1; }
            sleep 2; t=$((t+2))
        done
        log "PDP actif en ${t}s : $(ip netns exec "$NETNS" ip -4 -o addr show dev tun_modem | awk '{print $4}')"
    fi
}

vers_voix() {
    if systemctl is-active --quiet "$U_MODEM"; then
        systemctl stop "$U_MODEM"; log "modem GPRS arrete"
    fi
    systemctl reset-failed "$U_MODEM" >/dev/null 2>&1 || true
    if [ -z "$(mobiles_sur_sock)" ]; then
        [ -s "$ETAT/calypso-mobile.cmd" ] || { log "aucune ligne de mobile gardee : relancer le banc (systemctl
         ↪  restart osmo-banc)"; echo voix > "$MODE_F"; return 1; }
        local -a cmd; mapfile -t cmd < "$ETAT/calypso-mobile.cmd"
```

```
        local cwd jrn
        cwd="$(cat "$ETAT/calypso-mobile.cwd" 2>/dev/null || echo /)"
        jrn="$(cat "$ETAT/calypso-mobile.log" 2>/dev/null)"
        case "$jrn" in /*) ;; *) jrn=/tmp/c54x-pont/mobile.log ;; esac
        # Comme c54x_exe/run.sh (etape 4) : priorite de la chaine, sortie ligne a ligne
        ( cd "$cwd" && exec setsid taskset -c "${RT_CPUS:-0-15}" nice -n -10 stdbuf -oL "${cmd[@]}" >> "$jrn" 2>&1
          ↪  < /dev/null ) &
        log "mobile relance : ${cmd[*]} (journal $jrn)"
    fi
    echo voix > "$MODE_F"
    if [ "$ATTENDRE" = 1 ]; then
        local t=0
        until port_open "$MOB_VTY" && vty "show ms" | grep -q 'service is normal'; do
            [ $t -ge 120 ] && { log "mobile pas en service normal apres 120 s"; return 1; }
            sleep 2; t=$((t+2))
        done
        log "mobile en service normal en ${t}s (VTY $MOB_VTY)"
    fi
}

ACTION="${1:-etat}"; ATTENDRE=0
[ "${2:-}" = "--wait" ] && ATTENDRE=1
# [2026-10-15] Pas de couche 1, pas de bascule : banc arrete (ou pas en DSP),
# le mobile ou le modem relance mourait aussitot (« Failed to create unix
# domain socket /tmp/osmocom_l2 »). On le dit au lieu de lancer pour rien.
if [ "$ACTION" = data ] || [ "$ACTION" = voix ]; then
    if [ ! -S "$SOCK" ]; then
        log "pas de couche 1 ($SOCK absent) : le banc DSP ne tourne pas - systemctl start osmo-banc, puis relancer"
        exit 3
    fi
fi
case "$ACTION" in
    data) vers_data ;;
    voix) vers_voix ;;
    etat)
        echo "mode    : $(mode_courant)"
        echo "mobile  : $(mobiles_sur_sock | tr '\n' ' ')"
        echo "modem   : $(systemctl is-active "$U_MODEM" 2>/dev/null)"
        echo "PDP     : $(ip netns exec "$NETNS" ip -4 -o addr show dev tun_modem 2>/dev/null | awk '{print $4}')"
          ↪  ;;
    *) echo "usage : $0 data|voix|etat [--wait]" >&2; exit 2 ;;
esac
```

3.335 /opt/GSM/osmo-operator/scripts/check.sh

2863 octets, 53 lignes → 52 lignes (1 groupes compactés)

```
# ── INTER-STP ──────────────────────────────────────────────────────
sudo docker exec -it osmo-inter-stp telnet 127.0.0.1 4239
# show cs7 instance 0 asp
# show cs7 instance 0 as
# show cs7 instance 0 route

# ── OPERATEUR 1 - STP intra ────────────────────────────────────────
sudo docker exec -it osmo-operator-1 telnet 127.0.0.1 4239
# show cs7 instance 0 asp
# show cs7 instance 0 as
# show cs7 instance 0 sccp users
# show cs7 instance 0 sccp connections

# ── OPERATEUR 1 - MSC ──────────────────────────────────────────────
sudo docker exec -it osmo-operator-1 telnet 127.0.0.1 4254
# en
# show cs7 instance 0 asp
```

```
# show cs7 instance 0 sccp users
# show cs7 instance 0 sccp connections
# show subscriber all
# show connection all

# ── OPERATEUR 1 - BSC ──────────────────────────────────────────────────
sudo docker exec -it osmo-operator-1 telnet 127.0.0.1 4242
# en
# show cs7 instance 0 asp
# show cs7 instance 0 sccp users
# show cs7 instance 0 sccp connections
# show bts 0
# show trx 0 0

# ── OPERATEUR 2 - meme chose ───────────────────────────────────────────
sudo docker exec -it osmo-operator-2 telnet 127.0.0.1 ⟨1⟩  ×3
    ⟨⟩ = (4239) (4254) (4242)

# ── ONE-LINER : dump tout sans interaction ─────────────────────────────
for container in osmo-operator-1 osmo-operator-2; do
    echo "=== $container STP ==="
    sudo docker exec $container sh -c \
        'echo -e "show cs7 instance 0 asp\nshow cs7 instance 0 sccp users\nshow cs7 instance 0 sccp connections\n"
        ↪  | telnet 127.0.0.1 4239 2>/dev/null' \
        | grep -v "^Trying\|^Connected\|^Escape"

    echo "=== $container MSC ==="
    sudo docker exec $container sh -c \
        'echo -e "enable\nshow cs7 instance 0 sccp users\nshow cs7 instance 0 sccp connections\nshow subscriber
        ↪  all\n" | telnet 127.0.0.1 4254 2>/dev/null' \
        | grep -v "^Trying\|^Connected\|^Escape"
done

echo "=== inter-STP ASP ==="
sudo docker exec osmo-inter-stp sh -c \
    'echo -e "show cs7 instance 0 asp\nshow cs7 instance 0 as\n" | telnet 127.0.0.1 4239 2>/dev/null' \
    | grep -v "^Trying\|^Connected\|^Escape"
```

3.336 /opt/GSM/osmo-operator/scripts/dsp-rom-max.sh

14589 octets, 227 lignes → 227 lignes

```
#!/bin/bash
# ============================================================================
# scripts/dsp-rom-max.sh - workflow « le DSP en fait un maximum par la ROM ».
# ============================================================================
#
# Le banc DSP (c54x_exe, mask-ROM TI) peut remplacer certains traitements de
# la ROM par des reconstructions sur l hote (pont.py, montant.c, qosmo) :
# SB/TOA « en conserve », RACH code par le pont, codage xCCH montant par le
# pont, dechiffrement A5 par le pont... Chaque ecart est un commutateur
# d environnement (hacks_actifs() dans c54x_exe/src/pont.c les affiche au
# demarrage). Ce workflow :
#
#   1. profil     pose dans /etc/default/osmo-banc le profil « ROM max » : tous
#                 les commutateurs qui ont un chemin ROM sont sur ROM, aucun
#                 SB/TOA en conserve. Persistant : chaque boot du banc le lit.
#   2. banc       redemarre osmo-banc (--dsp) avec ce profil et attend le camp
#                 (VTY du mobile 4247).
#   3. echelle    joue l echelle de /opt/GSM/tests/banc-max.sh --dsp sur la
#                 pile en cours : LU, SMS MO/MT, appel voix, puis le barreau
#                 « couverture » (tests/modules/99-couverture.sh) qui tranche,
#                 traitement par traitement, ROM ou HOTE d apres les journaux.
#   4. bilan      relit couverture.txt : ce que la ROM a fait, ce qui est reste
```

```
#                 a l hote MALGRE le profil (a corriger), et ce qui n a pas de
#                 chemin ROM branche (le reste a faire : FACCH/SACCH-TCH/parole
#                 montants, A5 montant au fn de l air, vocodage).
#
# Usage : dsp-rom-max.sh [run|profile|report|status] [--no-restart] [--only=camp,attache]
#   run (defaut)   1 -> 4            profile  pose le profil seulement (1)
#   report         relit le dernier /root/banc-max-*/couverture.txt (4)
#   --no-restart   ne redemarre pas osmo-banc (la pile en cours a deja le profil ?)
#   --only=...     passe a banc-max.sh (ex. --only=camp,attache,couverture)
#   --parole-rom   [2026-10-15] TCH montant (parole, FACCH, SACCH/T) emis par la ROM :
#                  CALYPSO_NORM_TC=0 + CALYPSO_C54X_BCTC_NORM=1 (emulateur : le marqueur
#                  de trame de parole de PROM0 0x90a0 n'etait jamais pose) + PONT_UL_RAW_TCH=1.
#                  DEFAUT depuis le 18/10 ; --parole-hote revient en arriere (et coupe --vocodeur-rom).
#   --vocodeur-rom [2026-10-15] le codec FR de la ROM vocode (port voix SP0 modelise,
#                  calypso_vsp.c) : BANC_VOCODEUR=rom (mobile en l1phy, sans GAPK) +
#                  CALYPSO_VSP=1 (pont/vsp_audio.py <-> PulseAudio). Implique --parole-rom.
#                  + emulateur : CALYPSO_C54X_BCD_CARRY=1 (BCD sur C/NC, depaquetage FR PDROM 0xe458)
#                  et CALYPSO_C54X_OVM_LOGIQUE=1 (SFTL/AND/OR/XOR/ROL ne saturent pas, en-tete PDROM 0xe4c4)
#                  et CALYPSO_MAC=1 (MACR/MASR/MACA/MASA/MPYA conformes SPRU172C) : sans lui le decodeur
#                  FR de la ROM sort un spectre retourne (1 kHz rendu a ~3 kHz, 74 % de l'energie
#                  au-dessus de 3 kHz) ; avec, sa sortie au rejeu a le spectre de libgsm (16/10).
#                  + CALYPSO_C54X_BCD_TC=1 (vrai TC sur les 12 BCD tc/ntc : signe des coefficients
#                  de reflexion du codeur, Schur PDROM 0xe763/0xedc5, faux 47 % du temps sans lui)
#                  et CALYPSO_C54X_BCTC_COMBO=1 (BC tc,nc,bio : conditions ET-ees, isa 28). Avec ces
#                  drapeaux et le decalage ASM de ADD/SUB src,ASM (corrige par defaut le 16/10), le
#                  codeur FR de la ROM est bit-exact avec libgsm sur ~99 % des trames au rejeu.
#                  DEFAUT depuis le 18/10 (valide banc 16/10) ; --vocodeur-hote revient a GAPK.
#   --fw-gprs      firmware layer1 avec PDCH (layer1.highram.gprs.elf) : le DEFAUT depuis le 15/10 ;
#   --fw-stock     le firmware d origine (sans GPRS)
#
# Unite : services/osmo-dsp-rom-max.service (oneshot, lancee a la demande :
# une campagne de tests n a rien a faire au boot ; le PROFIL, lui, persiste).
# Journal : journalctl -u osmo-dsp-rom-max ; rapports : /root/banc-max-<date>/
# ------------------------------------------------------------------------------
set -uo pipefail
HERE="$(cd "$(dirname "${BASH_SOURCE[0]}")/.." && pwd)"
DEFAULTS=/etc/default/osmo-banc
TESTS=/opt/GSM/tests
FW_GPRS=/opt/GSM/firmware/board/compal_e88/layer1.highram.gprs.elf
# [2026-10-18] --vocodeur-rom (donc --parole-rom) PAR DEFAUT. Valide au banc le
# 16/10 (27/27, voix 1 kHz 100 %, tout ROM). Sans lui, une machine sans
# /etc/default/osmo-banc (acer, 10.99.0.2) tournait le codec FR de la ROM sans
# ses correctifs d emulateur : trames de silence au montant, echo test muet.
RESTART=1; ONLY=""; FWGPRS=1; PAROLE_ROM=1; VOCODEUR_ROM=1; ACTION=run
MOB_VTY=4247

if [ -t 1 ] && [ -z "${NO_COLOR:-}" ]; then C_OK=$'\033[32m'; C_KO=$'\033[31m'; C_WA=$'\033[33m';
 ↪  C_DIM=$'\033[2m'; C_B=$'\033[1m'; C_Z=$'\033[0m'
else C_OK=; C_KO=; C_WA=; C_DIM=; C_B=; C_Z=; fi
say()  { printf '  %s\n' "$*"; }
ok()   { printf '  %s[ OK ]%s %s\n' "$C_OK" "$C_Z" "$*"; }
ko()   { printf '  %s[ KO ]%s %s\n' "$C_KO" "$C_Z" "$*"; }
warn() { printf '  %s[ !! ]%s %s\n' "$C_WA" "$C_Z" "$*"; }
titre(){ printf '\n%s== %s ==%s\n' "$C_B" "$*" "$C_Z"; }
port_open() { timeout 1 bash -c "exec 3<>/dev/tcp/127.0.0.1/$1" 2>/dev/null; }

# ── 1. le profil ROM max ─────────────────────────────────────────────────
# Chaque ligne : VARIABLE=valeur   # ce qu elle met dans la ROM (et l autre valeur)
profil_lignes() { cat <<'P'
MONTANT_EFFACE_TASK_RA=0
MONTANT_ROM_UL=1
MONTANT_ROM_TX=1
UL_RAW=1
PONT_DSP_DECHIFFRE=0
CALYPSO_A5=1
MONTANT_REQREF=0
```

```
CALYPSO_C54X_BCTC_ETENDU=1
PONT_CAN_TOA=
PONT_CAN_SB_TOA=
PONT_CAN_SB=
P
}
profil_doc() { cat <<'P'
# ── profil « la ROM fait le maximum » (scripts/dsp-rom-max.sh) ──────────────
# MONTANT_EFFACE_TASK_RA=0  RACH : la ROM garde la tache et code l access-burst (1 : le pont le code)
# MONTANT_ROM_UL=1          SDCCH/SACCH montants : bursts codes par la ROM (0 : codage hote)
# MONTANT_ROM_TX=1          bursts finaux de la ROM publies au pont (calypso_tx_rom)
# UL_RAW=1                  le pont emet ces bursts tels quels (0 : recodage + A5 cote hote)
# PONT_DSP_DECHIFFRE=0      A5 descendant : la ROM dechiffre, pas le pont
# CALYPSO_A5=1              coprocesseur A5 du silicium, modelise dans c54x_exe (0 : coupe)
# MONTANT_REQREF=0          IMMEDIATE ASSIGNMENT : reference de la BTS laissee telle quelle (1 : qosmo la recale
 ↪  sur last_rach)
# CALYPSO_C54X_BCTC_ETENDU=1  emulateur : BC tc/ntc au vrai TC apres cmpr/bit/bitt et apres cmpm/bitf + ST
 ↪  (0xb2c7, double tampon I/Q) (0 : heuristique ACC)
# PONT_CAN_TOA= / PONT_CAN_SB_TOA= / PONT_CAN_SB=   rien « en conserve » : FB/SB/TOA par la ROM
P
}
etape_profil() {
    titre "1. profil ROM max dans $DEFAULTS"
    # [2026-10-18] acer (10.99.0.2) n a pas de layer1.highram.gprs.elf : le profil
    # l imposait quand meme et le banc refusait de partir. Firmware d origine alors.
    if [ "$FWGPRS" = 1 ] && { [ ! -r "$FW_GPRS" ] || [ ! -r "${FW_GPRS%.elf}.bin" ]; }; then
        warn "$FW_GPRS absent : firmware d origine (compiler le layer1 GPRS pour --fw-gprs)"
        FWGPRS=0
    fi
    touch "$DEFAULTS"
    # retirer une ancienne version du bloc, puis le reposer
    sed -i '/profil « la ROM fait le maximum »/,/^# ── fin profil ROM max/d' "$DEFAULTS"
    while IFS= read -r l; do [ -n "$l" ] && sed -i "/^${l%%=*}=/d" "$DEFAULTS"; done < <(profil_lignes)
    sed -i '/^FIRM-
     ↪  WARE_ELF=/d;/^FIRMWARE_BIN=/d;/^CALYPSO_NORM_TC=/d;/^CALYPSO_C54X_BCTC_NORM=/d;/^PONT_UL_RAW_TCH=/d;/^#
     ↪  parole montante
     ↪  ROM/d;/^BANC_VOCODEUR=/d;/^CALYPSO_VSP=/d;/^CALYPSO_C54X_BCD_CARRY=/d;/^CALYPSO_C54X_OVM_LOGIQUE=/d;/^CALYPSO_MAC=/d;/^
     ↪  vocodeur ROM/d' "$DEFAULTS"
    { profil_doc; profil_lignes
      if [ "$PAROLE_ROM" = 1 ]; then printf '# parole montante ROM (--parole-rom,
       ↪  defaut)\nCALYPSO_NORM_TC=0\nCALYPSO_C54X_BCTC_NORM=1\nPONT_UL_RAW_TCH=1\n'
      else printf '# parole montante hote (--parole-hote)\n'; fi
      if [ "$VOCODEUR_ROM" = 1 ]; then printf '# vocodeur ROM (--vocodeur-rom, de-
       ↪  faut)\nBANC_VOCODEUR=rom\nCALYPSO_VSP=1\nCALYPSO_C54X_BCD_CARRY=1\nCALYPSO_C54X_OVM_LOGIQUE=1\nCALYPSO_MAC=1\nCALYPSO
      else printf '# vocodeur hote, GAPK (--vocodeur-hote)\n'; fi
      if [ "$FWGPRS" = 1 ]; then printf 'FIRMWARE_ELF=%s\nFIRMWARE_BIN=%s\n' "$FW_GPRS" "${FW_GPRS%.elf}.bin"
      else printf '# FIRMWARE_ELF=%s   # --fw-gprs : layer1 avec PDCH (L1CTL_GPRS_*)\n' "$FW_GPRS"; fi
      printf '# ── fin profil ROM max ──\n'; } >> "$DEFAULTS"
    # [2026-10-15] PAS de OSMO_BANC_ARGS="--dsp" ici : dans l unite, EnvironmentFile=
    # l emporte sur `systemctl set-environment` (launch.sh --grgsm) et le banc
    # partait en DSP quoi qu on demande. Le DSP est le defaut de start-native.sh.
    sed -i '/^OSMO_BANC_ARGS=/d' "$DEFAULTS"
    ok "profil pose ($(profil_lignes | grep -c .) commutateurs)"; grep -E '^[A-Z_]+=' "$DEFAULTS" | sed 's/^/
     ↪  /'
}

# ── 2. le banc ──────────────────────────────────────────────────────────────
etape_banc() {
    titre "2. banc DSP (osmo-banc.service)"
    if [ "$RESTART" = 1 ]; then
        say "systemctl restart osmo-banc (plusieurs minutes : QEMU, C54x, camp)"
        systemctl reset-failed osmo-banc 2>/dev/null; systemctl restart osmo-banc || { ko "osmo-banc : journalctl
         ↪  -u osmo-banc -n 50"; return 1; }
    else say "--no-restart : pile en cours"; fi
    local t=0; while ! port_open $MOB_VTY; do [ $t -ge 180 ] && { ko "VTY du mobile DSP :$MOB_VTY absente apres
     ↪  180s"; return 1; }; sleep 3; t=$((t+3)); done
```

```
    ok "mobile DSP (VTY :$MOB_VTY) en ${t}s"
    local dsplog=/tmp/c54x-pont/dsp.log
    if [ -r "$dsplog" ]; then
        local h; h="$(grep -ao 'hacks[^)]*' "$dsplog" | head -1)"
        grep -aq 'laisse a la ROM : elle code l.access-burst' "$dsplog" && ok "RACH : tache laissee a la ROM" ||
         ↪  warn "RACH : pas de trace « laisse a la ROM » dans $dsplog"
        grep -aq 'rom-ul.*actif' "$dsplog" && ok "xCCH montant : bursts de la ROM publies" || warn "rom-ul : non
         ↪  actif dans $dsplog"
        grep -aq 'tx-rom.*actif\|calypso_tx_rom.*actif' "$dsplog" && ok "bursts finaux de la ROM : publies" || say
         ↪  "${C_DIM}(pas de ligne tx-rom actif lue)${C_Z}"
        [ -n "$h" ] && say "${C_DIM}$h${C_Z}"
    fi
    return 0
}

# ── 3. l echelle ──────────────────────────────────────────────────────
etape_echelle() {
    titre "3. echelle banc-max.sh --dsp --continue${ONLY:+ --only=$ONLY}"
    [ -x "$TESTS/banc-max.sh" ] || { ko "$TESTS/banc-max.sh absent"; return 1; }
    local log=/var/log/osmocom/dsp-rom-max-echelle.log
    mkdir -p /var/log/osmocom
    ( cd "$TESTS" && CALYPSO_NO_ATTACH=1 OSMO_NO_ATTACH=1 bash ./banc-max.sh --dsp --continue
     ↪  ${ONLY:+--only=$ONLY} ) > "$log" 2>&1
    local rc=$?
    grep -a 'ELEMENT MAX' "$log" | tail -1 | sed 's/^/  /'
    grep -aE '^\s+[a-z0-9-]+\s+(OK|ECHEC|SAUTE)' "$log" | sed 's/^/  /' | head -20
    [ "$rc" -eq 0 ] && ok "echelle : aucun barreau en echec" || warn "echelle : $rc barreau(x) en echec - $log"
    return 0
}

# ── 4. le bilan ROM / hote ────────────────────────────────────────────
etape_bilan() {
    titre "4. bilan ROM / hote"
    local rap; rap="$(ls -t /root/banc-max-*/couverture.txt 2>/dev/null | head -1)"
    [ -n "$rap" ] || { ko "aucun /root/banc-max-*/couverture.txt (le barreau couverture n a pas joue)"; return 1; }
    say "rapport : $rap"
    python3 -I - "$rap" <<'PY'
import re, sys
txt = open(sys.argv[1], errors='replace').read()
m = re.search(r'chemin ROM / hote du traitement du signal.*?\n(.*?)\nchemin : (\d+) ROM, (\d+) hote, (\d+) mixte,
 ↪  (\d+) non observe', txt, re.S)
if not m:
    print("  [KO] section « chemin ROM / hote » absente du rapport"); sys.exit(1)
rom, hote, mixte, nobs = map(int, m.group(2, 3, 4, 5))
rows = []
for l in m.group(1).splitlines():
    r = re.match(r'^  (.{44}) (ROM|HOTE|MIXTE|NON OBSERVE)\s+(\d+)\s+(\d+)\s+(.*)$', l)
    if r: rows.append((r.group(1).strip(), r.group(2), int(r.group(3)), int(r.group(4)), r.group(5)))
attendu_rom = [r for r in rows if r[4].startswith('ROM')]
hote_constr = [r for r in rows if r[4].startswith('hote')]
print("  %-46s %-12s %s" % ("TRAITEMENT", "VERDICT", "rom/hote"))
for t, v, nr, nh, ch in rows:
    tag = {"ROM": "[ OK ]", "HOTE": "[ KO ]", "MIXTE": "[ !! ]", "NON OBSERVE": "[ -- ]"}[v]
    if v == "HOTE" and ch.startswith('hote'): tag = "[hote]"
    print("  %s %-39s %-12s %d/%d" % (tag, t[:39], v, nr, nh))
print()
print("  ROM %d, hote %d, mixte %d, non observe %d" % (rom, hote, mixte, nobs))
bad = [r for r in attendu_rom if r[1] in ("HOTE", "MIXTE")]
if bad:
    print("  A CORRIGER (chemin ROM attendu, l hote a pris la main) :")
    for t, v, nr, nh, ch in bad: print("    - %s : %s (rom=%d hote=%d) — %s" % (t, v, nr, nh, ch))
else:
    print("  [ OK ] tout ce qui a un chemin ROM est passe par la ROM")
print("  RESTE A BRANCHER SUR LA ROM (hote par construction) :")
for t, v, nr, nh, ch in hote_constr: print("    - %s — %s" % (t, ch))
PY
```

```
}

while [ $# -gt 0 ]; do
    case "$1" in
        run|profile|report|status) ACTION="$1" ;;
        --no-restart) RESTART=0 ;;
        --only=*) ONLY="${1#*=}" ;;
        --fw-gprs) FWGPRS=1 ;;
        --fw-stock) FWGPRS=0 ;;
        --parole-rom) PAROLE_ROM=1 ;;
        --parole-hote) PAROLE_ROM=0; VOCODEUR_ROM=0 ;;
        --vocodeur-rom) VOCODEUR_ROM=1; PAROLE_ROM=1 ;;
        --vocodeur-hote) VOCODEUR_ROM=0 ;;
        -h|--help) sed -n '2,36p' "$0"; exit 0 ;;
        *) echo "option inconnue : $1" >&2; exit 2 ;;
    esac; shift
done
[ "$(id -u)" -eq 0 ] || { echo "a lancer en root" >&2; exit 1; }
case "$ACTION" in
    profile) etape_profil ;;
    report)  etape_bilan ;;
    status)  systemctl is-active osmo-banc; grep -E '^[A-Z_]+=' "$DEFAULTS" 2>/dev/null | sed 's/^/  /'; port_open
     ↪  $MOB_VTY && echo "  mobile DSP : VTY :$MOB_VTY ouverte" ;;
    run)     etape_profil; etape_banc || exit 1; etape_echelle; etape_bilan ;;
esac
```

3.337 /opt/GSM/osmo-operator/scripts/egprs-dsp.sh

23070 octets, 378 lignes → 378 lignes

```
#!/bin/bash
# ============================================================================
# scripts/egprs-dsp.sh - workflow EGPRS du banc, sur la chaine DSP (ROM TI).
# ============================================================================
#
# Ce que ce workflow monte, dans l ordre, et verifie a chaque etape :
#
#   1. preflight    binaires, ROM DSP (7 fichiers), c54x_exe, modem osmocom-bb ;
#                   et la PORTE DSP : le firmware layer1 Calypso sait-il faire
#                   du PDCH (L1CTL_GPRS_*) ? Aujourd hui NON -> le modem GPRS
#                   tourne sur une couche 1 virtuelle (virtphy), dans la MEME
#                   PLMN/LAC/RA que le mobile DSP, qui lui campe sur la BTS 0.
#   2. configs      /etc/osmocom : la BTS 2 virtuelle EGPRS dans osmo-bsc.cfg
#                   (fragment configs/gprs/osmo-bsc-bts2-egprs.frag, re-pose
#                   apres chaque --regen), osmo-bts-virtual-gprs.cfg,
#                   osmo-pcu-gprs.cfg, modem-gprs.cfg.
#   3. banc         osmo-banc.service (start-native.sh --dsp : coeur, BTS 0 sur
#                   le C54x + ROM TI, SGSN/GGSN/PCU de la BTS 0).
#   4. abonne       l IMSI du modem dans le HLR (comp128v1), idempotent.
#   5. cellule      osmo-bts-virtual (BTS 2, Um multicast sur lo) + osmo-pcu
#                   (socket /tmp/pcu_bts3) en unites systemd transitoires ;
#                   attend l OML de la BTS 2 au BSC et le NS-VC du PCU.
#   6. modem        virtphy (-> /tmp/modem_l2) puis osmocom-bb `modem` :
#                   GPRS attach (GMM) + contexte PDP (SM), tun dans le netns
#                   msgprs ; verifie au SGSN (mm-context, pdp-context) et par
#                   un ping vers apn0 depuis le netns.
#
#   EGPRS_L1=dsp : (quand la porte DSP passera) pas de virtphy ; le modem est
#   lance par c54x_exe/run.sh a la place de `mobile` (MOBILE= / MOBILE_CFG= dans
#   /etc/default/osmo-banc) sur /tmp/osmocom_l2. Le script pose ces deux
#   variables et le dit ; il ne force pas un chemin que le firmware n a pas.
#
# Usage : egprs-dsp.sh [start|stop|status|check] [--l1 auto|virtphy|dsp]
#                      [--no-banc] [--regen-gprs] [--all]
```

```
#   start        tout monter (defaut)            stop   demonter la cellule + modem
#   status       etat + SGSN                     check  preflight seul, ne lance rien
#   --no-banc    ne pas toucher a osmo-banc      --all  (stop) arreter aussi osmo-banc
#   --regen-gprs reposer les 4 configs GPRS depuis configs/gprs meme si presentes
#
# Persistant : services/osmo-egprs-dsp.service (Requires=osmo-banc) enclenche ce
# script au boot ; OSMO_BANC_ARGS="--dsp" dans /etc/default/osmo-banc.
# Journal : journalctl -u osmo-egprs-dsp, -u osmo-modem-gprs, -u osmo-pcu-gprs,
#           -u osmo-bts-virtual-gprs, -u osmo-virtphy-gprs.
# ------------------------------------------------------------------------------
set -uo pipefail

HERE="$(cd "$(dirname "${BASH_SOURCE[0]}")/.." && pwd)"
CFG_SRC="$HERE/configs/gprs"
ETC="${OSMOCOM_CFG:-/etc/osmocom}"
BB="${OSMOCOM_BB:-/opt/GSM/osmocom-bb}"
C54X="${C54X_DIR:-/opt/GSM/c54x_exe}"
ROM_DIR="${OSMO_DSP_ROM_DIR:-/opt/GSM}"
L1="${EGPRS_L1:-auto}"              # auto | virtphy | dsp
WITH_BANC="${EGPRS_BANC:-1}"
NETNS="${EGPRS_NETNS:-msgprs}"
REGEN_GPRS=0
STOP_ALL=0

# L abonne du modem : le 3e du plan (01-abonnes-hlr : octet 15 = n, 16 = op).
IMSI="${EGPRS_IMSI:-001010001000001}"
KI="${EGPRS_KI:-00 11 22 33 44 55 66 77 88 99 aa bb cc dd 01 01}"
MSISDN="${EGPRS_MSISDN:-100101}"

SOCK_VIRT=/tmp/modem_l2
SOCK_DSP=/tmp/osmocom_l2
PCU_SOCK=/tmp/pcu_bts3

VTY_BSC=4242; VTY_HLR=4258; VTY_SGSN=4245
PCU_VTY_IP=127.0.0.2; VTY_PCU=4240

U_BTS=osmo-bts-virtual-gprs; U_PCU=osmo-pcu-gprs; U_VPHY=osmo-virtphy-gprs; U_MODEM=osmo-modem-gprs
UNITS=("$U_MODEM" "$U_VPHY" "$U_PCU" "$U_BTS")

# -- affichage ----------------------------------------------------------------
if [ -t 1 ] && [ -z "${NO_COLOR:-}" ]; then
    C_OK=$'\033[32m'; C_KO=$'\033[31m'; C_WA=$'\033[33m'; C_DIM=$'\033[2m'; C_B=$'\033[1m'; C_Z=$'\033[0m'
else C_OK=; C_KO=; C_WA=; C_DIM=; C_B=; C_Z=; fi
say()  { printf '  %s\n' "$*"; }
ok()   { printf '  %s[ OK ]%s %s\n' "$C_OK" "$C_Z" "$*"; }
ko()   { printf '  %s[ KO ]%s %s\n' "$C_KO" "$C_Z" "$*"; }
warn() { printf '  %s[ !! ]%s %s\n' "$C_WA" "$C_Z" "$*"; }
titre(){ printf '\n%s== %s ==%s\n' "$C_B" "$*" "$C_Z"; }
declare -A BILAN=()
bilan() { BILAN["$1"]="$2"; }

# -- outils -------------------------------------------------------------------
# Le VTY Osmocom lit ligne a ligne : une pause entre chaque commande, sinon il
# en saute (meme constat que start-interstp.sh --status).
vty() {   # vty HOST PORT cmd...
    local host="$1" port="$2"; shift 2
    { sleep 0.4; printf 'enable\n'; local c; for c in "$@"; do sleep 0.3; printf '%s\n' "$c"; done
      sleep 0.8; printf 'exit\n'; sleep 0.2; } | timeout 10 nc -q1 "$host" "$port" 2>/dev/null | tr -d '\r'
}
port_open() { timeout 1 bash -c "exec 3<>/dev/tcp/${2:-127.0.0.1}/$1" 2>/dev/null; }
wait_until() {   # wait_until SECONDES "libelle" cmd...
    local to="$1" what="$2"; shift 2; local t=0
    while ! "$@" >/dev/null 2>&1; do
        [ "$t" -ge "$to" ] && { ko "$what : toujours rien apres ${to}s"; return 1; }
        sleep 2; t=$((t+2))
    done
```

```
    ok "$what (${t}s)"
}
unit_active() { systemctl is-active --quiet "$1"; }
unit_run() {   # unit_run NOM cmd... : unite transitoire, vue par systemctl/journalctl
    local u="$1"; shift
    if unit_active "$u"; then say "$u deja actif"; return 0; fi
    systemctl reset-failed "$u" >/dev/null 2>&1 || true
    systemd-run --quiet --unit="$u" --collect -p KillMode=control-group \
        -p SyslogIdentifier="$u" "$@" || { ko "systemd-run $u a echoue"; return 1; }
    say "$u lance : $*"
}
unit_stop() { unit_active "$1" && systemctl stop "$1" 2>/dev/null; systemctl reset-failed "$1" >/dev/null 2>&1 ||
 ↪  true; }

# ── 1. preflight ──────────────────────────────────────────────────────
DSP_GPRS=0
MODEM_BIN=""
etape_preflight() {
    titre "1. preflight"
    local rc=0 b
    for b in osmo-bts-virtual osmo-pcu virtphy nc systemd-run; do
        command -v "$b" >/dev/null 2>&1 && ok "$b : $(command -v "$b")" || { ko "$b introuvable"; rc=1; }
    done
    # modem osmocom-bb : dans le PATH, sinon celui de l arbre ; on l installe
    # (persistant) s il n est que dans l arbre.
    MODEM_BIN="$(command -v modem 2>/dev/null || true)"
    if [ -z "$MODEM_BIN" ] && [ -x "$BB/src/host/layer23/src/modem/modem" ]; then
        install -m755 "$BB/src/host/layer23/src/modem/modem" /usr/local/bin/modem 2>/dev/null \
            && MODEM_BIN=/usr/local/bin/modem && say "modem installe dans /usr/local/bin (depuis $BB)"
    fi
    [ -n "$MODEM_BIN" ] && ok "modem : $MODEM_BIN" || { ko "modem (osmocom-bb layer23) absent : make -C
     ↪  $BB/src/host/layer23"; rc=1; }

    # La chaine DSP : ROM TI + c54x_exe (le banc par defaut de start-native.sh)
    local n=0 r
    for r in PROM0 PROM1 PROM2 PROM3 DROM PDROM Registers; do [ -f "$ROM_DIR/calypso_dsp.$r.bin" ] && n=$((n+1));
     ↪  done
    [ "$n" -eq 7 ] && ok "ROM DSP TI : $n/7 dans $ROM_DIR" || warn "ROM DSP : $n/7 dans $ROM_DIR - --dsp ne
     ↪  demarrera pas (c54x_exe/rom/fetch-rom.sh)"
    [ -x "$C54X/c54x_exe" ] && ok "c54x_exe : $C54X/c54x_exe" || warn "c54x_exe non compile : make -C $C54X"

    # LA PORTE DSP : le firmware layer1 (celui qui tourne sur l ARM emule, devant
    # la ROM) traite-t-il les primitives GPRS de L1CTL ? Sans elles, aucun PDCH
    # ne peut etre ouvert sur cette chaine, quoi que sache la ROM.
    if grep -q 'L1CTL_GPRS' "$BB/src/target/firmware/layer1/l23_api.c" 2>/dev/null; then
        DSP_GPRS=1; ok "porte DSP : le firmware layer1 traite L1CTL_GPRS_* -> PDCH possible sur le C54x"
    else
        DSP_GPRS=0; warn "porte DSP fermee : pas de L1CTL_GPRS_* dans firmware/layer1/l23_api.c -> le modem ira
         ↪  sur virtphy"
    fi
    grep -qi 'gprs' "$C54X"/src/*.c 2>/dev/null && ok "c54x_exe : du code GPRS present" \
        || say "${C_DIM}c54x_exe : aucune tache GPRS cote pont (src/*.c) - a faire quand la porte s ouvrira${C_Z}"
    say "${C_DIM}rappel : la ROM Calypso module en GMSK seulement (GPRS CS-1..4) ; la cellule annonce EGPRS,"
    say "le 8-PSK (MCS-5..9) restera au modem virtphy, jamais au C54x.${C_Z}"

    case "$L1" in
        auto) [ "$DSP_GPRS" = 1 ] && L1=dsp || L1=virtphy ;;
        dsp|virtphy) ;;
        *) ko "--l1 $L1 inconnu (auto|virtphy|dsp)"; rc=1 ;;
    esac
    say "couche 1 du modem : $C_B$L1$C_Z"
    [ "$rc" -eq 0 ] && bilan preflight OK || bilan preflight KO
    return $rc
}

# ── 2. configs ────────────────────────────────────────────────────────
```

```
BSC_CHANGE=0
pose_cfg() {   # pose_cfg SRC DST [sed-expr]
    local src="$CFG_SRC/$1" dst="$ETC/$2"
    [ -r "$src" ] || { ko "gabarit absent : $src"; return 1; }
    if [ -f "$dst" ] && [ "$REGEN_GPRS" = 0 ]; then say "$dst present (--regen-gprs pour le reposer)"; return 0; fi
    sed -e "${3:-}" "$src" > "$dst" && ok "$dst pose"
}
etape_configs() {
    titre "2. configs dans $ETC"
    local rc=0 sgsn_ip gsmtap_ip
    sgsn_ip="$(awk '$1=="listen"{print $2; exit}' "$ETC/osmo-sgsn.cfg" 2>/dev/null)"
    [ -n "$sgsn_ip" ] || { warn "pas de « listen » NS dans osmo-sgsn.cfg - 127.0.0.1"; sgsn_ip=127.0.0.1; }
    gsmtap_ip="$(awk '$1=="log" && $2=="gsmtap"{print $3; exit}' "$ETC/osmo-pcu.cfg" 2>/dev/null)"
    [ -n "$gsmtap_ip" ] || gsmtap_ip=127.0.0.1
    pose_cfg osmo-bts-virtual-gprs.cfg osmo-bts-virtual-gprs.cfg || rc=1
    pose_cfg osmo-pcu-gprs.cfg        osmo-pcu-gprs.cfg "s|__GSMTAP_IP__|$gsmtap_ip|g" || rc=1
    pose_cfg modem-gprs.cfg           modem-gprs.cfg || rc=1
    # Variante DSP du modem : meme abonne, socket L1CTL du C54x. Posee pour le
    # jour ou la porte s ouvre (MOBILE_CFG= dans /etc/default/osmo-banc).
    pose_cfg modem-gprs.cfg modem-gprs-dsp.cfg "s|layer2-socket .*|layer2-socket $SOCK_DSP|" || rc=1

    # La BTS 2 dans osmo-bsc.cfg : un bloc « bts 2 » insere avant « msc 0 ».
    # --regen regenere ce fichier depuis configs/osmo-bsc.cfg (2 BTS) : on
    # verifie a chaque start, on re-insere si besoin.
    local bsc="$ETC/osmo-bsc.cfg"
    if [ ! -f "$bsc" ]; then ko "$bsc absent (start-native.sh --regen)"; rc=1
    elif grep -q '^ bts 2$' "$bsc"; then ok "BTS 2 EGPRS deja dans osmo-bsc.cfg"
    elif ! grep -q '^msc 0$' "$bsc"; then ko "pas de « msc 0 » dans $bsc : je ne sais pas ou inserer la BTS 2"; rc=1
    else
        cp -a "$bsc" "$bsc.avant-bts2-egprs"
        sed "s|__SGSN_IP__|$sgsn_ip|g" "$CFG_SRC/osmo-bsc-bts2-egprs.frag" > "$bsc.frag.$$"
        # Insertion avant « msc 0 », sans toucher au reste du fichier.
        awk -v frag="$bsc.frag.$$" '
            /^msc 0$/ && !done { while ((getline l < frag) > 0) print l; done=1 }
            { print }' "$bsc.avant-bts2-egprs" > "$bsc"
        rm -f "$bsc.frag.$$"
        grep -q '^ bts 2$' "$bsc" && { ok "BTS 2 EGPRS inseree dans osmo-bsc.cfg (sauvegarde :
         ↪  $bsc.avant-bts2-egprs)"; BSC_CHANGE=1; } \
            || { ko "insertion de la BTS 2 ratee"; cp -a "$bsc.avant-bts2-egprs" "$bsc"; rc=1; }
    fi
    [ "$rc" -eq 0 ] && bilan configs OK || bilan configs KO
    return $rc
}

# ── 3. banc (osmo-banc.service, chaine DSP) ──────────────────────────
etape_banc() {
    titre "3. banc DSP (osmo-banc.service)"
    if [ "$WITH_BANC" != 1 ]; then say "--no-banc : on suppose le banc deja monte"
    elif unit_active osmo-banc; then
        ok "osmo-banc actif"
        if [ "$BSC_CHANGE" = 1 ] && unit_active osmo-bsc; then
            warn "la BTS 2 vient d etre ajoutee : osmo-bsc relit sa config -> restart (le mobile DSP recampera)"
            systemctl restart osmo-bsc
        fi
    else
        say "systemctl start osmo-banc (start-native.sh --no-attach, DSP par defaut) - plusieurs minutes"
        systemctl start osmo-banc || { ko "osmo-banc : journalctl -u osmo-banc -n 50"; bilan banc KO; return 1; }
    fi
    wait_until 120 "VTY OsmoBSC :$VTY_BSC"  port_open "$VTY_BSC"  || { bilan banc KO; return 1; }
    wait_until 60  "VTY OsmoHLR :$VTY_HLR"  port_open "$VTY_HLR"  || { bilan banc KO; return 1; }
    wait_until 60  "VTY OsmoSGSN :$VTY_SGSN" port_open "$VTY_SGSN" || { bilan banc KO; return 1; }
    local mode; mode="$(vty 127.0.0.1 $VTY_BSC "show running-config" | awk '/^ bts 0$/{b=1} b && /gprs mode/{print
     ↪  $3; exit}')"
    [ "$mode" = egprs ] && ok "BTS 0 (C54x) : gprs mode egprs" || warn "BTS 0 : gprs mode ${mode:-?}"
    bilan banc OK
}
```

```
# ── 4. l abonne du modem dans le HLR ──────────────────────────────────────
etape_abonne() {
    titre "4. abonne $IMSI (HLR :$VTY_HLR)"
    if vty 127.0.0.1 $VTY_HLR "subscriber imsi $IMSI show" | grep -q "IMSI: *$IMSI"; then
        ok "deja present"
    else
        vty 127.0.0.1 $VTY_HLR "subscriber imsi $IMSI create" \
            "subscriber imsi $IMSI update msisdn $MSISDN" \
            "subscriber imsi $IMSI update aud2g comp128v1 ki $(echo "$KI" | tr -d ' ')" >/dev/null
        vty 127.0.0.1 $VTY_HLR "subscriber imsi $IMSI show" | grep -q "IMSI: *$IMSI" \
            && ok "cree (MSISDN $MSISDN, comp128v1)" || { ko "creation ratee : telnet 127.0.0.1 $VTY_HLR"; bilan
             ↪  abonne KO; return 1; }
    fi
    bilan abonne OK
}

# ── 5. la cellule paquet : BTS 2 virtuelle + son PCU ──────────────────────
etape_cellule() {
    titre "5. cellule EGPRS virtuelle (BTS 2, PCU $PCU_SOCK)"
    rm -f "$PCU_SOCK" 2>/dev/null
    unit_run "$U_BTS" osmo-bts-virtual -c "$ETC/osmo-bts-virtual-gprs.cfg" || { bilan cellule KO; return 1; }
    bts2_oml() { vty 127.0.0.1 $VTY_BSC "show bts 2" | grep -i 'OML' | grep -viE 'disconnected|not connected' |
     ↪  grep -qi 'connected'; }
    wait_until 60 "OML de la BTS 2 au BSC" bts2_oml || { warn "journalctl -u $U_BTS -n 30 ; show bts 2 sur
     ↪  :$VTY_BSC"; }
    unit_run "$U_PCU" osmo-pcu -c "$ETC/osmo-pcu-gprs.cfg" || { bilan cellule KO; return 1; }
    wait_until 30 "socket PCU $PCU_SOCK" test -S "$PCU_SOCK" || warn "la BTS 2 n a pas ouvert $PCU_SOCK"
    pcu_ns() { vty $PCU_VTY_IP $VTY_PCU "show ns" | grep -qiE 'ALIVE|UNBLOCKED'; }
    wait_until 60 "NS-VC PCU <-> SGSN (nsei 11)" pcu_ns || warn "NS : show ns sur $PCU_VTY_IP:$VTY_PCU ;
     ↪  journalctl -u $U_PCU"
    bilan cellule OK
}

# ── 6. le modem GPRS ──────────────────────────────────────────────────────
etape_modem() {
    titre "6. modem GPRS (couche 1 : $L1)"
    ip netns list 2>/dev/null | grep -qw "$NETNS" || { ip netns add "$NETNS" && say "netns $NETNS cree"; }
    if [ "$L1" = dsp ]; then
        # QEMU + ROM TI : le modem prend la place de `mobile` SUR LA MEME socket
        # L1CTL (/tmp/osmocom_l2, osmocon du banc DSP). On ne passe pas par
        # MOBILE=/MOBILE_CFG= dans /etc/default/osmo-banc : start-native.sh
        # reecrit le fichier designe par MOBILE_CFG avec son gabarit `mobile`
        # (gps device...), que `modem` refuse. On arrete donc le mobile du banc
        # DSP et on lance modem nous-memes, en unite transitoire.
        local cfg="$ETC/modem-gprs-dsp.cfg"
        sed "s|layer2-socket .*|layer2-socket $SOCK_DSP|" "$CFG_SRC/modem-gprs.cfg" > "$cfg"
        # Le firmware layer1 avec PDCH (L1CTL_GPRS_*) et les bursts finaux de la ROM
        # (UL_RAW=1, pont/dsp/uplink.py) : poses pour le prochain demarrage du banc.
        local fw=/opt/GSM/firmware/board/compal_e88/layer1.highram.gprs.elf df=/etc/default/osmo-banc
        if [ -r "$fw" ]; then
            touch "$df"
            grep -q "^FIRMWARE_ELF=$fw" "$df" || { sed -i '/^FIRMWARE_ELF=/d;/^FIRMWARE_BIN=/d' "$df"; printf
             ↪  'FIRMWARE_ELF=%s\nFIRMWARE_BIN=%s\n' "$fw" "${fw%.elf}.bin" >> "$df"; warn "firmware GPRS pose
             ↪  dans $df : « systemctl restart osmo-banc » pour le charger"; }
            grep -q '^UL_RAW=1' "$df" || { sed -i '/^UL_RAW=/d' "$df"; printf 'UL_RAW=1\n' >> "$df"; }
            tr -d '\0' < "/proc/$(pgrep -f 'qemu-system-arm.*calypso' | head -1)/cmdline" 2>/dev/null | grep -q
             ↪  "$fw" \
               && ok "QEMU tourne avec le firmware GPRS" || warn "QEMU ne tourne pas (encore) avec le firmware
                ↪  GPRS : restart osmo-banc"
        else
            warn "firmware GPRS absent : $fw (make board/compal_e88/layer1.highram.elf dans
             ↪  osmocom-bb/src/target/firmware)"
        fi
        [ -S "$SOCK_DSP" ] || { ko "socket L1CTL $SOCK_DSP absente : le banc DSP (osmocon) ne tourne pas"; bilan
         ↪  modem KO; return 1; }
```

```
        # [2026-10-15] Le banc doit etre EN DSP : sans c54x_exe, la couche 1 est gr-gsm
        # (OSMO_BANC_ARGS=--grgsm pose par le lanceur du bureau jusqu au reboot) et
        # le PDCH n y existe pas.
        if ! pgrep -f 'c54x_exe --arm' >/dev/null; then
            ko "le banc tourne en gr-gsm (pas de c54x_exe) : $(systemctl show-environment 2>/dev/null | grep
             ↪  OSMO_BANC_ARGS || echo 'OSMO_BANC_ARGS non pose')"
            say "pour le DSP : systemctl set-environment OSMO_BANC_ARGS=--dsp && systemctl restart osmo-banc"
            bilan modem KO; return 1
        fi
        # Tout `mobile` branche sur la meme socket L1CTL (mobile_pont du banc DSP,
        # mobile_group1 du fork...) se battrait avec le modem : chacun remet la
        # couche 1 a zero (RESET FULL + PM_REQ contre RESET FULL + FBSB_REQ).
        local p c n=0
        for p in $(pgrep -x mobile); do
            c="$(tr '\0' ' ' < /proc/$p/cmdline 2>/dev/null | sed -n 's/.* -c \([^ ]*\).*/\1/p')"
            [ -n "$c" ] && grep -q "layer2-socket $SOCK_DSP\$" "$c" 2>/dev/null && kill "$p" 2>/dev/null &&
             ↪  n=$((n+1)) && say "mobile $p ($c) arrete : il occupait $SOCK_DSP"
        done
        [ "$n" -gt 0 ] && sleep 1
        unit_run "$U_MODEM" "$MODEM_BIN" -c "$cfg" || { bilan modem KO; return 1; }
        [ "$DSP_GPRS" = 1 ] || warn "porte DSP fermee : le modem campera, mais l attach echouera tant que le
         ↪  firmware layer1 ne traite pas L1CTL_GPRS_* (voir preflight)"
    else
        rm -f "$SOCK_VIRT"
        unit_run "$U_VPHY" virtphy -s "$SOCK_VIRT" -D lo || { bilan modem KO; return 1; }
        wait_until 20 "socket L1CTL $SOCK_VIRT" test -S "$SOCK_VIRT" || { bilan modem KO; return 1; }
        unit_run "$U_MODEM" "$MODEM_BIN" -c "$ETC/modem-gprs.cfg" || { bilan modem KO; return 1; }
    fi
    # [2026-10-15] L attach et le PDP se lisent chez le modem : « show mm-context »
    # du SGSN n affiche pas « GMM_REGISTERED » (le test echouait sur un attach
    # reussi). Attach = « Attach success » du modem ; PDP = tun_modem adresse
    # dans le netns, posee par le modem a l Activate PDP Context Accept.
    local t_debut; t_debut="$(date '+%Y-%m-%d %H:%M:%S')"
    attache() { journalctl -u "$U_MODEM" --since "$t_debut" -o cat 2>/dev/null | grep -q 'Attach success'; }
    wait_until 180 "GPRS attach de $IMSI (Attach success, P-TMSI attribue)" attache || {
       warn "journalctl -u $U_MODEM -n 40 ; show mm-context imsi $IMSI sur :$VTY_SGSN"; bilan modem KO; return 1; }
    pdp() { ip netns exec "$NETNS" ip -4 -o addr show dev tun_modem 2>/dev/null | grep -q 'inet '; }
    wait_until 90 "contexte PDP : adresse sur tun_modem (netns $NETNS)" pdp || { warn "pas de PDP : show
     ↪  pdp-context all ; journalctl -u $U_MODEM"; bilan modem KO; return 1; }
    ok "adresse du modem : $(ip netns exec "$NETNS" ip -4 -o addr show dev tun_modem | awk '{print $4}')"
    local apn_ip; apn_ip="$(ip -4 -o addr show apn0 2>/dev/null | awk '{print $4}' | cut -d/ -f1 | head -1)"
    if [ -n "$apn_ip" ]; then
        ip netns exec "$NETNS" ip route replace default dev tun_modem 2>/dev/null
        if ip netns exec "$NETNS" ping -c4 -W8 "$apn_ip" >/dev/null 2>&1; then ok "ping $apn_ip (apn0) depuis le
         ↪  netns $NETNS : la donnee passe"
        else warn "ping $apn_ip depuis le netns $NETNS rate (ip netns exec $NETNS ip route)"; fi
    fi
    bilan modem OK
}

# ── actions ──────────────────────────────────────────────────────────
do_stop() {
    titre "arret"
    local u; for u in "${UNITS[@]}"; do unit_stop "$u" && say "$u arrete"; done
    rm -f "$SOCK_VIRT" "$PCU_SOCK"
    [ "$STOP_ALL" = 1 ] && { systemctl stop osmo-banc; say "osmo-banc arrete"; }
    return 0
}
do_status() {
    titre "etat"
    local u; for u in osmo-banc "${UNITS[@]}"; do printf '  %-24s %s\n' "$u" "$(systemctl is-active "$u"
     ↪  2>/dev/null)"; done
    port_open "$VTY_SGSN" && { say "SGSN mm-context :"; vty 127.0.0.1 $VTY_SGSN "show mm-context all" | grep -E
     ↪  'IMSI|MM state|Routing|PDP' | sed 's/^/    /'
        say "SGSN pdp-context :"; vty 127.0.0.1 $VTY_SGSN "show pdp-context all" | grep -E 'IMSI|APN|Address|PDP'
         ↪  | sed 's/^/    /'; }
```

```
    port_open "$VTY_PCU" "$PCU_VTY_IP" && { say "PCU gprs (tbf) :"; vty $PCU_VTY_IP $VTY_PCU "show tbf all" | grep
     ↪  -vE '^(OsmoPCU|enable|exit|$)' | sed 's/^/    /' | head -12; }
    return 0
}
do_start() {
    etape_preflight || return 1
    etape_configs   || return 1
    etape_banc      || return 1
    etape_abonne    || true
    # La cellule virtuelle n a de sens que pour un modem sur virtphy : en mode
    # dsp (QEMU + ROM TI) le modem campe sur la BTS 0 du C54x, comme `mobile`.
    if [ "$L1" = virtphy ] || [ "${EGPRS_VIRTUAL_CELL:-0}" = 1 ]; then etape_cellule || true
    else say "cellule virtuelle (BTS 2) non montee : mode $L1 (EGPRS_VIRTUAL_CELL=1 pour l avoir quand meme)";
     ↪  bilan cellule "--"; fi
    etape_modem     || true
    titre "bilan"
    local k; for k in preflight configs banc abonne cellule modem; do printf '  %-10s %s\n' "$k"
     ↪  "${BILAN[$k]:-??}"; done
    say "${C_DIM}status : $0 status - arret : $0 stop - journaux : journalctl -u $U_MODEM -u $U_PCU -u $U_BTS -u
     ↪  $U_VPHY${C_Z}"
    return 0
}

ACTION=start
while [ $# -gt 0 ]; do
    case "$1" in
        start|stop|status|check) ACTION="$1" ;;
        --l1)        L1="${2:-auto}"; shift ;;
        --l1=*)      L1="${1#*=}" ;;
        --no-banc)   WITH_BANC=0 ;;
        --regen-gprs) REGEN_GPRS=1 ;;
        --all)       STOP_ALL=1 ;;
        -h|--help)   sed -n '2,40p' "$0"; exit 0 ;;
        *) echo "option inconnue : $1" >&2; exit 2 ;;
    esac; shift
done
[ "$(id -u)" -eq 0 ] || { echo "a lancer en root (systemd-run, ip netns)" >&2; exit 1; }
case "$ACTION" in
    start)  do_start ;;
    stop)   do_stop ;;
    status) do_status ;;
    check)  etape_preflight; etape_configs ;;
esac
```

3.338 /opt/GSM/osmo-operator/scripts/entrypoint.sh

1969 octets, 64 lignes → 64 lignes

```
#!/bin/bash
set -e

# --- AJOUT DE LA PARTIE TUN ---
echo "[*] Initialisation du peripherique TUN pour osmo-ggsn..."
mkdir -p /dev/net
if [ ! -c /dev/net/tun ]; then
    # Creation du noeud de peripherique (c = caractere, 10 = majeur, 200 = mineur)
    mknod /dev/net/tun c 10 200
    chmod 666 /dev/net/tun
fi

# --- CONFIGURATION ENVIRONNEMENT ---
container=docker
export container
```

```
# Verification de la presence d'une commande
if [ $# -eq 0 ]; then
  echo >&2 'ERROR: No command specified. You probably want to run bash or a script.'
  exit 1
fi

# Export des variables pour les sessions systemd
env > /etc/docker-entrypoint-env

# Creation du service de demarrage
quoted_args="$(printf " %q" "${@}")"
echo "${quoted_args}" > /etc/docker-entrypoint-cmd

cat >/etc/systemd/system/docker-entrypoint.service <<EOT
[Unit]
Description=Lancement de la stack Osmocom Simulee
After=network.target

[Service]
ExecStart=/bin/bash -exc "source /etc/docker-entrypoint-cmd"
StandardInput=tty-force
StandardOutput=inherit
StandardError=inherit
WorkingDirectory=/etc/osmocom
EnvironmentFile=/etc/docker-entrypoint-env

[Install]
WantedBy=multi-user.target
EOT

# Desactivation des services systemd conflictuels
systemctl mask systemd-firstboot.service systemd-udevd.service systemd-modules-load.service \
              systemd-udevd-kernel systemd-udevd-control 2>/dev/null || true
systemctl enable docker-entrypoint.service

# Localisation et execution de systemd
if [ -x /lib/systemd/systemd ]; then
  exec /lib/systemd/systemd --show-status=false --unit=multi-user.target
elif [ -x /usr/lib/systemd/systemd ]; then
  exec /usr/lib/systemd/systemd --show-status=false --unit=multi-user.target
else
  echo >&2 'ERROR: systemd is not installed'
  exit 1
fi

# NOTE : Les directives RUN/COPY/EXPOSE qui etaient ici sont des instructions
# Dockerfile et doivent etre dans le Dockerfile, pas dans ce script.
# exec systemd remplace le process - rien apres cette ligne n'est execute.
```

3.339 /opt/GSM/osmo-operator/scripts/gapk-start.sh

38590 octets, 732 lignes → 732 lignes

```
#!/bin/bash
# gapk-start.sh - Gestionnaire audio GSM (osmo-gapk ↔ ALSA/RTP)
#
# ═══════════════════════════════════════════════════════════════════════
# ARCHITECTURE AUDIO DOUBLE CHEMIN (osmo-nitb-for-calypso v2)
# ═══════════════════════════════════════════════════════════════════════
#
# CHEMIN 1 - Baseband (mobile l1phy) :
#   mobile process ← TCH frames → trxcon → fake_trx → osmo-bts-trx
#                  ↓ io-handler l1phy
#                  ALSA device (decodage GSM-FR direct, zero latence reseau)
```

```
#
#   → Active par tch-voice { io-handler l1phy } dans mobile.cfg
#   → Pas besoin de gapk pour ce chemin
#   → Simule un combine GSM physique
#
# CHEMIN 2 - Reseau (MGW RTP) :
#   BTS → BSC → MSC → MNCC → Asterisk → SIP → Linphone
#                          → MGCP → OsmoMGW → RTP
#                                             ↓
#                                        osmo-gapk (monitor/record/bridge)
#
#   → osmo-gapk se greffe sur les ports RTP du MGW
#   → Utile pour : monitoring, enregistrement, injection audio, tests
#   → Linphone recoit l'audio directement via SIP (pas besoin de gapk)
#
# RESUME :
#   • Appel MS → MS (meme operateur) : audio via l1phy ALSA (chemin 1)
#   • Appel MS → Linphone            : audio via Asterisk SIP (chemin 2)
#   • Monitoring reseau              : gapk mode auto/monitor (chemin 2)
#   • Enregistrement appel           : gapk mode record (chemin 2)
#
# ═══════════════════════════════════════════════════════════════════════

# ═══════════════════════════════════════════════════════════════════════
# CONTRAT CLI D'osmo-gapk - RELIRE AVANT DE TOUCHER UNE INVOCATION
# ═══════════════════════════════════════════════════════════════════════
# [2026-08-10] Tout ce fichier parlait a un osmo-gapk QUI N'EXISTE PAS : un
# format nomme "gsmfr" et une syntaxe d'URL "-i rtp://host:port/20".
# Consequence mesuree : gapk-rx.log et gapk-tx.log ne contenaient QUE
# "Unsupported format: gsmfr", en boucle depuis le demarrage - les deux
# process mouraient a l'analyse des arguments, le superviseur les relancait,
# et le pont RTP<->ALSA n'a JAMAIS transporte un seul echantillon. Le
# superviseur, lui, restait vivant : la sonde `_audio_gapk_vivant` le voyait
# et declarait le pont "en place". Sonde mensongere de bout en bout.
#
# Le contrat REEL (osmo-gapk --help, et src/app_osmo_gapk.c l.206-295) :
#
#   -i FICHIER        entree FICHIER          -o FICHIER   sortie FICHIER
#   -I HOTE/PORT      entree RTP              -O HOTE/PORT sortie RTP
#   -a PERIPH         entree ALSA             -A PERIPH    sortie ALSA
#   -f FORMAT         format d'ENTREE         -g FORMAT    format de SORTIE
#   -p PT             payload type RTP in     -P PT        payload type RTP out
#   -t                throttle (une trame codec / 20 ms)
#
# Pieges :
#   • `-i` n'est PAS `-I`. Une URL passee a `-i` devient un NOM DE FICHIER.
#   • `-f` seul ne suffit pas : sans `-g`, aucun format de sortie n'est defini.
#   • Le nom du format FR est "gsm" (fichier .gsm ET payload RTP RFC3551),
#     PAS "gsmfr". `osmo-gapk --help` fait foi ; la liste complete est
#     gsm | rawpcm-s16le | rtp-efr | rtp-hr-etsi | rtp-hr-ietf | amr-* | ti-* |
#     racal-* | hr-ref-*.
#   • Un seul codec est enc+dec dans ce build : `fr`. hr/efr/amr/pcm sont
#     format-only (colonne "enc dec" vide dans --help). Le reseau est en
#     payload type 3 = GSM FR (verifie : osmo-mgw "payload-types:3=GSM").
#   • `-t` UNIQUEMENT sur une entree FICHIER. Sur RTP c'est le reseau qui
#     cadence, sur ALSA c'est la carte : throttler en plus desynchronise.
# ═══════════════════════════════════════════════════════════════════════

set -euo pipefail

GAPK_DEFAULT_FORMAT="${GAPK_FORMAT:-gsm}"
GAPK_DEFAULT_DEVICE="${GAPK_ALSA_DEV:-${ALSA_CARD:-default}}"
# [2026-08-10] Peripherique de CAPTURE distinct de celui de LECTURE. Les deux
# pattes utilisaient `$dev` (gsm_out) : cote TX cela demandait a ALSA de
# CAPTURER sur un peripherique de lecture. Et surtout, capturer la ou les
# mobiles ecrivent, c'est exactement la boucle fermee que le commentaire
```

```
# d'/etc/asound.conf documente pour le 08/08 (RMS x11 en 10 s). La capture doit
# viser gsm_in (= moniteur du null-sink gsm_mic, que personne n'alimente).
# Ne JAMAIS faire retomber ce defaut sur $GAPK_DEFAULT_DEVICE.
GAPK_DEFAULT_DEVICE_IN="${GAPK_ALSA_DEV_IN:-gsm_in}"
# Payload type RTP du GSM FR (RFC3551). OsmoMGW negocie "3=GSM".
GAPK_RTP_PT="${GAPK_RTP_PT:-3}"
GAPK_LOG_DIR="${GAPK_LOG_DIR:-/var/log/osmocom}"
GAPK_REC_DIR="${GAPK_REC_DIR:-/var/lib/gapk}"
# Periode de trame GSM. osmo-gapk la porte en dur (option -t) : cette variable
# ne sert plus qu'a documenter la cadence dans les messages.
GAPK_FRAME_MS=20
MGW_VTY_HOST="127.0.0.1"
MGW_VTY_PORT="4243"
AUTO_POLL_INTERVAL="${GAPK_POLL_INTERVAL:-2}"

GAPK_PID_RX="/var/run/gapk-rx.pid"
GAPK_PID_TX="/var/run/gapk-tx.pid"
GAPK_AUTO_LOCK="/var/run/gapk-auto.lock"

RED='\033[0;31m'; GREEN='\033[0;32m'; YELLOW='\033[1;33m'
CYAN='\033[0;36m'; NC='\033[0m'; BOLD='\033[1m'

mkdir -p "$GAPK_LOG_DIR" "$GAPK_REC_DIR"

log_info()  { echo -e "${GREEN}[gapk]${NC} $*"; }
log_warn()  { echo -e "${YELLOW}[gapk]${NC} $*"; }
log_error() { echo -e "${RED}[gapk ERROR]${NC} $*" >&2; }
log_auto()  { echo -e "${CYAN}[gapk-auto]${NC} $(date '+%H:%M:%S') $*"; }
# [2026-09-23] La chaine audio lourde de l hote (annuleur d echo, boucles,
# enregistrement) ne tourne que pendant un appel : posee sur « Endpoint actif »,
# retiree sur « Endpoint disparu ». En tache de fond, pour ne pas retarder la
# scrutation. Voir scripts/audio-chain.sh --appel.
_audio_appel() {
    [ "${AUDIO_PENDANT_APPEL:-0}" = "1" ] || return 0
    local chaine="${OSMO_OPERATOR_DIR:-/opt/GSM/osmo-operator}/scripts/audio-chain.sh"
    [ -f "$chaine" ] || return 0
    bash "$chaine" --appel "$1" 3 >>"${GAPK_LOG_DIR:-/tmp}/audio-appel.log" 2>&1 &
}

# ── Verifications ──────────────────────────────────────────────────────
check_gapk() {
    command -v osmo-gapk >/dev/null 2>&1 || {
        log_error "osmo-gapk introuvable."
        log_error "Verifier que le Dockerfile inclut l'etape osmo-gapk --enable-alsa."
        exit 1
    }
}

# [2026-08-12] Cette fonction jugeait sur la PRESENCE DE /dev/snd. Or aucun PCM
# de cette chaine n'est une carte : gsm_out et gsm_in sont `type pulse`
# (/etc/asound.conf) et sortent dans des null-sinks. Sur une machine sans carte
# son - un serveur, exactement la cible du projet - start-docker.sh ne passe pas
# `--device /dev/snd` (start-docker.sh:191), check_alsa echouait, et mode_loopback
# faisait `exit 1 : ALSA requis` alors que toute la chaine pulse etait
# fonctionnelle. On teste donc CE QU'ON VA UTILISER : le PCM s'ouvre-t-il ?
# Meme geste que assert_audio_devices() de lib/audio.sh - une seconde de
# silence, et un verdict qui porte sur le peripherique reel.
check_alsa() {
    local dev="${1:-default}"
    command -v aplay >/dev/null 2>&1 || {
        log_warn "aplay absent - impossible de verifier le PCM '${dev}'"
        return 1
    }
    if timeout 5 aplay -D "$dev" -f S16_LE -r 8000 -c 1 -d 1 /dev/zero >/dev/null 2>&1; then
        return 0
    fi
```

```
    log_warn "PCM ALSA '${dev}' ne s'ouvre pas."
    [ -d /dev/snd ] || log_warn "  (/dev/snd absent - sans importance si le PCM est 'type pulse')"
    pactl info >/dev/null 2>&1 \
        || log_warn "  PulseAudio injoignable (PULSE_SERVER=${PULSE_SERVER:-non defini})"
    log_warn "  verifier /etc/asound.conf et 'pactl list short sinks' (gsm_audio + gsm_mic)"
    return 1
}

# ── Detection du mode audio mobile ──────────────────────────────────
detect_audio_mode() {
    # [2026-09-09] Le mobile porte SA PROPRE chaine audio des qu'un io-handler
    # (gapk ou l1phy) est declare dans un de ses cfg : c'est lui qui lit gsm_in
    # et ecrit gsm_out. Avant, seul /root/.osmocom/bb/mobile.cfg etait regarde -
    # un fichier qui n'existe plus (mobile_group1.cfg, mobile_faketrx_bts1.cfg) -
    # d'ou un "gapk" par defaut qui ne voulait rien dire.
    #   mobile  : le process mobile fait l'audio (io-handler gapk|l1phy)
    #   l1phy   : idem, conserve pour les appelants historiques
    #   gapk    : aucun mobile n'a d'audio -> chemin 2 (bridge RTP<->ALSA)
    local cfg
    for cfg in "${OSMOCOM_HOME:-/root/.osmocom}"/bb/mobile*.cfg /root/.osmocom/bb/mobile*.cfg; do
        [ -f "$cfg" ] || continue
        grep -q "io-handler l1phy" "$cfg" 2>/dev/null && { echo "l1phy"; return 0; }
        grep -q "io-handler gapk"  "$cfg" 2>/dev/null && { echo "mobile"; return 0; }
    done
    echo "gapk"
}

# ── Adressage RTP ──────────────────────────────────────────────────
# osmo-gapk decoupe HOTE/PORT au SLASH (app_osmo_gapk.c:190, strtok(dup, "/")).
# Un "127.0.0.1:16386" lui donne host="127.0.0.1:16386" puis strtok NULL ->
# -EINVAL -> "Invalid port". Or tout ce script, sa CLI publique et la sortie
# de mode_mgw_ports parlent en IP:PORT. On normalise ICI, une seule fois, pour
# ne pas avoir a changer le contrat visible du script.
rtp_hostport() {
    local hp="$1"
    # deja en HOTE/PORT : on ne touche pas. Sinon dernier ':' -> '/'.
    case "$hp" in
        */*) echo "$hp" ;;
        *)   echo "${hp%:*}/${hp##*:}" ;;
    esac
}

# ── Cycle de vie background ─────────────────────────────────────────
run_bg() {
    local pid_file="$1"; shift
    local log_file="$1"; shift
    osmo-gapk "$@" >> "$log_file" 2>&1 &
    echo $! > "$pid_file"
    log_info "PID $(cat "$pid_file") → $(basename "$log_file")"
}

stop_pid() {
    local pid_file="$1"
    [ -f "$pid_file" ] || return 0
    local pid; pid=$(cat "$pid_file")
    if kill -0 "$pid" 2>/dev/null; then
        kill "$pid" 2>/dev/null && log_info "Arrete PID $pid" || true
    fi
    rm -f "$pid_file"
}

is_running() {
    local pid_file="$1"
    [ -f "$pid_file" ] && kill -0 "$(cat "$pid_file")" 2>/dev/null
}
```

```
# ═══════════════════════════════════════════════════════════════════════
# info - affiche l'architecture audio actuelle
# ═══════════════════════════════════════════════════════════════════════
mode_info() {
    local audio_mode
    audio_mode=$(detect_audio_mode)

    echo -e "${CYAN}${BOLD}"
    echo "╔═══════════════════════════════════════════════════════════╗"
    echo "║        Architecture audio osmo-nitb-for-calypso           ║"
    echo "╚═══════════════════════════════════════════════════════════╝"
    echo -e "${NC}"

    echo -e "  Mode mobile.cfg : ${BOLD}${audio_mode}${NC}"
    echo ""

    if [ "$audio_mode" = "l1phy" ]; then
        echo -e "  ${YELLOW}● Chemin 1 - Baseband (l1phy)${NC}"
        echo -e "    mobile.cfg a tch-voice l1phy mais fake_trx ne transporte"
        echo -e "    PAS le payload audio TCH (bursts downlink = zeros)."
        echo -e "    → Pas de son direct sur les MS virtuels."
        echo ""
        echo -e "  ${GREEN}● Chemin 2 - Reseau (SIP/RTP) - SEUL CHEMIN AUDIO${NC}"
        echo -e "    MS appelle → Asterisk → SIP → Linphone (audio natif)"
        echo -e "    Linphone appelle → Asterisk → MNCC → MSC → MS (signalisation)"
        echo -e "    → L'audio du lab passe par Linphone."
    else
        echo -e "  ${GREEN}● Mode gapk classique${NC}"
        echo -e "    osmo-gapk bridge RTP ↔ ALSA"
        echo -e "    → Requis pour l'audio des appels MS"
    fi

    echo ""
    echo -e "  ${BOLD}Linphone :${NC}"
    local inter_ip
    inter_ip=$(ip -4 addr show | grep '172\.20\.0\.' | awk '{print $2}' | cut -d/ -f1 | head -1 || echo
    ↪  "172.20.0.X")
    echo -e "    SIP server : ${CYAN}${inter_ip}:5060${NC}"
    echo -e "    linphone_A : user=${CYAN}linphone_A${NC}  pass=${CYAN}tester${NC}   → 100"
    echo -e "    linphone_B : user=${CYAN}linphone_B${NC}  pass=${CYAN}testerB${NC}  → 200"
    echo ""

    # MGW ports
    mode_mgw_ports "" 2>/dev/null || true
}

# ═══════════════════════════════════════════════════════════════════════
# mgw-ports
# ═══════════════════════════════════════════════════════════════════════
mode_mgw_ports() {
    local quiet="${1:-}"
    local mgw_out

    mgw_out=$( (echo "show mgcp"; sleep 1; echo "exit") | telnet "$MGW_VTY_HOST" "$MGW_VTY_PORT" 2>/dev/null \
        | grep -v "^Trying\|^Connected\|^Escape\|Welcome\|OsmoMGW[>#]\|Copyright\|License\|Contributions\|free
        ↪  software\|NO WARRANTY\|Based on") || true

    if [ -z "$mgw_out" ]; then
        [ -z "$quiet" ] && echo "(MGW non disponible ou aucun endpoint)"
        return 1
    fi

    # Format reel OsmoMGW :
    #   Virtual trunk 0 endpoint rtpbridge/1@mgw:
    #      CONN: (9/rtp C:51E1AF8F r=127.0.0.1:4072<->l=127.0.0.1:4068)
    #      CONN: (9/rtp C:96FA1465 r=127.0.0.1:30004<->l=127.0.0.1:4066)
```

```
    if [ -z "$quiet" ]; then
        echo -e "${CYAN}── Endpoints OsmoMGW actifs ──${NC}"
        local has_conn=false
        echo "$mgw_out" | while IFS= read -r line; do
            if echo "$line" | grep -q "endpoint.*@mgw"; then
                local ep; ep=$(echo "$line" | grep -oP 'rtpbridge/\S+')
                # Peek if next lines have CONN
                continue
            fi
            if echo "$line" | grep -q "CONN:.*r="; then
                has_conn=true
                local remote local_p
                remote=$(echo "$line" | grep -oP 'r=\K[0-9.]+:[0-9]+')
                local_p=$(echo "$line" | grep -oP 'l=\K[0-9.]+:[0-9]+')
                echo -e "  ${GREEN}●${NC} local=${CYAN}${local_p}${NC}  remote=${CYAN}${remote}${NC}"
            fi
        done
        # [2026-08-10] has_conn etait affecte DANS le sous-shell du pipeline
            # (`echo | while`) : perdu a la sortie -> "aucune connexion active"
            # s'imprimait en listant 4 connexions. Sonde mensongere.
            if ! echo "$mgw_out" | grep -q "CONN:.*r="; then
            echo "  (aucune connexion active)"
        fi
    else
        # Mode machine : "LOCAL_PORT REMOTE_IP:REMOTE_PORT" par connexion active
        echo "$mgw_out" | grep "CONN:.*r=" | while IFS= read -r line; do
            local remote local_p
            remote=$(echo "$line" | grep -oP 'r=\K[0-9.]+:[0-9]+')
            # [2026-08-10] LE MOTIF EXIGEAIT 'l=PORT', OsmoMGW ecrit 'l=IP:PORT'. Sans le
            # point dans la classe, grep bute sur le premier '.' et rend VIDE -> le test
            # ci-dessous echoue -> le mode machine ne sort AUCUNE ligne -> mode_auto ne
            # lance JAMAIS gapk, en silence, endpoints actifs ou non. La branche
            # d'AFFICHAGE portait deja [0-9.]+ : seule celle qui pilote l'automate etait
            # restee en arriere. ATTENTION : la SOURCE est ici (scripts/), /etc/osmocom en
            # est une COPIE regeneree au demarrage - patcher la copie seule ne survit pas.
            local_p=$(echo "$line" | grep -oP 'l=\K[0-9.]+:[0-9]+' | cut -d: -f2)
            [ -n "$local_p" ] && [ -n "$remote" ] && echo "$local_p $remote"
        done
    fi
}

# ═══════════════════════════════════════════════════════════════════
# call - appel bidirectionnel ALSA ↔ RTP (chemin 2 uniquement)
# ═══════════════════════════════════════════════════════════════════
mode_call() {
    local rx_port="${1:?RTP_RX_PORT requis (port MGW downlink)}"
    local tx_dest="${2:?RTP_TX_DEST:PORT requis (dest MGW uplink)}"
    local fmt="${3:-$GAPK_DEFAULT_FORMAT}"
    local dev="${4:-$GAPK_DEFAULT_DEVICE}"
    local dev_in="${5:-$GAPK_DEFAULT_DEVICE_IN}"

    check_gapk
    check_alsa "$dev" || true

    stop_pid "$GAPK_PID_RX"
    stop_pid "$GAPK_PID_TX"

    local audio_mode; audio_mode=$(detect_audio_mode)
    if [ "$audio_mode" = "l1phy" ]; then
        echo -e "${YELLOW}${BOLD}NOTE: mobile.cfg utilise io-handler l1phy${NC}"
        echo -e "${YELLOW}L'audio MS passe directement via ALSA (pas via gapk).${NC}"
        echo -e "${YELLOW}Ce mode gapk bridge le RTP reseau (MGW) ↔ ALSA.${NC}"
        echo -e "${YELLOW}Pour les appels MS↔MS, l'audio est deja gere par mobile.${NC}"
        echo ""
    fi
```

```
    echo -e "${CYAN}${BOLD}"
    echo "╔═══════════════════════════════════════════════╗"
    echo "║   gapk - Bridge RTP reseau ↔ ALSA             ║"
    echo "╚═══════════════════════════════════════════════╝"
    echo -e "${NC}"
    printf "  Codec     : ${CYAN}%s${NC}\n" "$fmt"
    printf "  Periph HP : ${CYAN}%s${NC}\n" "$dev"
    printf "  Periph mic: ${CYAN}%s${NC}\n" "$dev_in"
    printf "  RX ecoute : ${CYAN}0.0.0.0:%s${NC}  (MGW downlink → HP)\n" "$rx_port"
    printf "  TX envoi  : ${CYAN}%s${NC}  (micro → MGW uplink)\n" "$tx_dest"
    echo ""

    local log_rx="${GAPK_LOG_DIR}/gapk-rx.log"
    local log_tx="${GAPK_LOG_DIR}/gapk-tx.log"

    # RX : RTP (FR) → decodage codec fr → PCM → ALSA lecture
    log_info "RX : rtp/${rx_port} → alsa:${dev}  [${fmt} → rawpcm-s16le]"
    run_bg "$GAPK_PID_RX" "$log_rx" \
        -I "0.0.0.0/${rx_port}" -f "$fmt" -p "$GAPK_RTP_PT" \
        -A "$dev"               -g rawpcm-s16le

    sleep 0.3

    # TX : ALSA capture → PCM → encodage codec fr → RTP (FR)
    log_info "TX : alsa:${dev_in} → rtp/${tx_dest}  [rawpcm-s16le → ${fmt}]"
    run_bg "$GAPK_PID_TX" "$log_tx" \
        -a "$dev_in"                    -f rawpcm-s16le \
        -O "$(rtp_hostport "$tx_dest")"  -g "$fmt" -P "$GAPK_RTP_PT"

    echo ""
    log_info "Bridge actif. Arreter : gapk-start.sh stop"
}

# ═══════════════════════════════════════════════════════════════════════
# auto - surveille MGW, bridge gapk auto
# ═══════════════════════════════════════════════════════════════════════
mode_auto() {
    local fmt="${1:-$GAPK_DEFAULT_FORMAT}"
    local dev="${2:-$GAPK_DEFAULT_DEVICE}"
    local dev_in="${3:-$GAPK_DEFAULT_DEVICE_IN}"

    check_gapk
    check_alsa "$dev" || log_warn "ALSA absent - mode RTP-only"

    local audio_mode; audio_mode=$(detect_audio_mode)

    echo -e "${CYAN}${BOLD}"
    echo "╔═══════════════════════════════════════════════════╗"
    echo "║   gapk-auto - Integration OsmoMGW (chemin reseau)   ║"
    echo "╚═══════════════════════════════════════════════════╝"
    echo -e "${NC}"

    if [ "$audio_mode" = "l1phy" ]; then
        log_auto "Mode l1phy detecte → audio MS via mobile process"
        log_auto "gapk surveille le chemin reseau (MGW RTP) uniquement"
        log_auto "Pour les appels MS↔MS, l'audio est gere nativement"
        echo ""
    fi

    log_auto "Codec=${fmt}  HP=${dev}  Mic=${dev_in}  Poll=${AUTO_POLL_INTERVAL}s"
    log_auto "Ctrl+C pour arreter"
    echo ""

    touch "$GAPK_AUTO_LOCK"
    local prev_rx_port=""
```

```
cleanup_auto() {
    log_auto "Arret..."
    stop_pid "$GAPK_PID_RX"
    stop_pid "$GAPK_PID_TX"
    rm -f "$GAPK_AUTO_LOCK"
    exit 0
}
trap cleanup_auto SIGINT SIGTERM

while [ -f "$GAPK_AUTO_LOCK" ]; do
    local first_active
    first_active=$(mode_mgw_ports "quiet" 2>/dev/null | head -1) || first_active=""

    if [ -n "$first_active" ]; then
        local rx_port tx_dest
        rx_port=$(echo "$first_active" | awk '{print $1}')
        tx_dest=$(echo "$first_active" | awk '{print $2}')

        if [ "$rx_port" != "$prev_rx_port" ]; then
            if [ -n "$prev_rx_port" ]; then
                log_auto "Changement endpoint (${prev_rx_port} → ${rx_port})"
                stop_pid "$GAPK_PID_RX"
                stop_pid "$GAPK_PID_TX"
            fi

            log_auto "Endpoint actif : RX=${rx_port}  TX=${tx_dest}"
            [ -z "$prev_rx_port" ] && _audio_appel debut

            local log_rx="${GAPK_LOG_DIR}/gapk-rx.log"
            local log_tx="${GAPK_LOG_DIR}/gapk-tx.log"

            # ── PATTE RX : ARCHITECTURALEMENT IMPOSSIBLE, ON NE LA LANCE PLUS ──
            # [2026-08-10] `-I 0.0.0.0/$rx_port` echoue en
            # "unable to bind socket:0.0.0.0:4006: Address already in use".
            # Ce n'est pas un reglage a trouver : $rx_port est le port LOCAL
            # d'OsmoMGW, il le detient deja. Deux process ne peuvent pas
            # ecouter le meme port UDP, et surtout le MGW n'ENVOIE pas vers
            # ce port - il envoie vers son pair distant. Recevoir la copie
            # d'un flux MGW demande que le MGW nous l'adresse, donc une
            # connexion MGCP supplementaire sur l'endpoint : ca se regle
            # cote MGCP, jamais cote gapk.
            # Avant ce correctif, la patte RX mourait a chaque tour et le
            # superviseur la relancait indefiniment - une boucle muette qui
            # donnait l'illusion d'un pont en place.
            # POUR ECOUTER l'audio reseau : Asterisk enregistre deja chaque
            # appel (MixMonitor, cf. sub-record dans extensions.conf), avec
            # depuis le 10/08 un fichier par sens : *-ul.wav / *-dl.wav.
            log_auto "RX non lancee : ${rx_port} appartient a OsmoMGW (voir le commentaire)"
            log_auto "  ecoute reseau -> enregistrements MixMonitor (*-ul.wav / *-dl.wav)"

            # ── PATTE TX : MEME VERDICT QUE LA RX, ON NE LA LANCE PLUS ──
            # [2026-09-09] CAUSE DU « 600 qu'on entend mal ». $tx_dest est le
            # PAIR DISTANT de la premiere connexion MGW - en pratique le port
            # RTP d'osmo-bts (127.0.0.1:16384, cf. osmo-mgw.log « Got media
            # info via SDP: port:16384 ... 3=GSM » sur rtpbridge/2). Cette
            # patte encodait donc le micro (gsm_in) en GSM-FR et l'ENVOYAIT
            # AU BTS, sur le meme port que le flux legitime du MGW (la voix
            # qui revient d'Asterisk). Le BTS recevait DEUX flux RTP entre-
            # meles - deux SSRC, deux horloges - dans un seul jitter-buffer
            # de 100 ms : trames jetees, ordre casse, et en prime le micro
            # renvoye en direct dans l'oreille du mobile. Son hache, double,
            # inintelligible ; aucun compteur d'erreur ne bronchait puisque
            # chaque paquet etait valide.
            # Le mobile (io-handler gapk, alsa gsm_in/gsm_out) fait deja
            # l'aller-retour micro <-> TCH par la radio : cette patte etait
```

```
                # un DOUBLON par construction, pas un reglage a affiner.
                # GAPK_AUTO_TX=1 la reactive pour le seul chemin 2 (aucun mobile
                # avec audio) ; par defaut le superviseur ne fait qu'observer.
                if [ "${GAPK_AUTO_TX:-0}" = "1" ] && [ "$audio_mode" = "gapk" ]; then
                    # TX : ALSA capture → PCM → codec fr (encode) → RTP (FR)
                    run_bg "$GAPK_PID_TX" "$log_tx" \
                        -a "$dev_in"                    -f rawpcm-s16le \
                        -O "$(rtp_hostport "$tx_dest")"  -g "$fmt" -P "$GAPK_RTP_PT"
                    log_auto "Audio actif (PID RX=$(cat $GAPK_PID_RX 2>/dev/null) TX=$(cat $GAPK_PID_TX
                     ↪  2>/dev/null))"
                else
                    log_auto "TX non lancee : ${tx_dest} est le pair du MGW (BTS/Asterisk), pas une entree libre"
                  log_auto "  le montant est porte par le mobile (mode ${audio_mode}) - GAPK_AUTO_TX=1 pour forcer"
                fi

                prev_rx_port="$rx_port"
            fi

            # On ne surveille QUE la patte TX, et seulement si elle a ete
            # lancee : la RX n'est plus lancee (cf. ci-dessus). L'exiger
            # relancait le couple sans fin.
            if [ -f "$GAPK_PID_TX" ] && ! is_running "$GAPK_PID_TX"; then
                log_warn "Processus gapk mort - restart"
                stop_pid "$GAPK_PID_RX"; stop_pid "$GAPK_PID_TX"
                prev_rx_port=""
            fi
        else
            if [ -n "$prev_rx_port" ]; then
                log_auto "Endpoint disparu - arret gapk"
                _audio_appel fin
                stop_pid "$GAPK_PID_RX"; stop_pid "$GAPK_PID_TX"
                prev_rx_port=""
            fi
        fi
        sleep "$AUTO_POLL_INTERVAL"
    done
}

# ═══════════════════════════════════════════════════════════════════════
# monitor - ecoute passive RTP → ALSA
# ═══════════════════════════════════════════════════════════════════════
mode_monitor() {
    local rx_port="${1:?RTP_RX_PORT requis}"
    local fmt="${2:-$GAPK_DEFAULT_FORMAT}"
    local dev="${3:-$GAPK_DEFAULT_DEVICE}"

    check_gapk; check_alsa "$dev" || true
    stop_pid "$GAPK_PID_RX"

    log_info "Monitor : rtp/${rx_port} → alsa:${dev}  [${fmt} → rawpcm-s16le]"
    exec osmo-gapk \
        -I "0.0.0.0/${rx_port}" -f "$fmt" -p "$GAPK_RTP_PT" \
        -A "$dev"               -g rawpcm-s16le
}

# ═══════════════════════════════════════════════════════════════════════
# record - RTP → fichier .gsm
# ═══════════════════════════════════════════════════════════════════════
mode_record() {
    local rx_port="${1:?RTP_RX_PORT requis}"
    local outfile="${2:-${GAPK_REC_DIR}/record_$(date '+%Y%m%d_%H%M%S').gsm}"
    local fmt="${3:-$GAPK_DEFAULT_FORMAT}"

    check_gapk; stop_pid "$GAPK_PID_RX"

    # Sortie fichier au MEME format que l'entree : on archive les trames FR
```

```
    # telles quelles (.gsm lisible par gapk playback et par grgsm_decode -d FR).
    log_info "Record : rtp/${rx_port} → ${outfile}  [${fmt}]"
    exec osmo-gapk \
        -I "0.0.0.0/${rx_port}" -f "$fmt" -p "$GAPK_RTP_PT" \
        -o "$outfile"           -g "$fmt"
}

# ═══════════════════════════════════════════════════════════════════
# playback - fichier .gsm → RTP
# ═══════════════════════════════════════════════════════════════════
mode_playback() {
    local infile="${1:?INPUT_FILE requis}"
    local tx_dest="${2:?RTP_TX_DEST:PORT requis}"
    local fmt="${3:-$GAPK_DEFAULT_FORMAT}"

    check_gapk
    [ -f "$infile" ] || { log_error "Fichier introuvable : $infile"; exit 1; }

    # -t OBLIGATOIRE ici : l'entree est un FICHIER, rien ne cadence la lecture.
    # Sans throttle gapk vide le fichier a pleine vitesse et noie le jitter
    # buffer d'en face (des minutes d'audio expediees en quelques secondes).
    log_info "Playback : ${infile} → rtp/${tx_dest}  [${fmt}]"
    exec osmo-gapk -t \
        -i "$infile"                    -f "$fmt" \
        -O "$(rtp_hostport "$tx_dest")"  -g "$fmt" -P "$GAPK_RTP_PT"
}

# ═══════════════════════════════════════════════════════════════════
# loopback - ALSA micro → codec → ALSA HP
# ═══════════════════════════════════════════════════════════════════
mode_loopback() {
    local fmt="${1:-$GAPK_DEFAULT_FORMAT}"
    local dev="${2:-$GAPK_DEFAULT_DEVICE}"
    local dev_in="${3:-$GAPK_DEFAULT_DEVICE_IN}"

    check_gapk
    check_alsa "$dev" || { log_error "ALSA requis."; exit 1; }

    echo -e "${YELLOW}${BOLD}⚠  UTILISEZ DES ECOUTEURS - risque de larsen !${NC}"
    # CE QUE CE MODE PROUVE, ET CE QU'IL NE PROUVE PAS.
    # osmo-gapk construit UNE chaine "format d'entree -> canon -> format de
    # sortie" : le codec n'apparait que du cote ou le format EST un format
    # codec. PCM -> fr -> PCM n'est donc PAS exprimable en une invocation, et
    # $fmt n'est volontairement pas utilise ici. Ce loopback valide le chemin
    # ALSA (capture, cadence, peripheriques), PAS le vocodeur FR.
    # Pour eprouver le vocodeur : `record` puis `playback` du .gsm obtenu.
    log_info "Loopback ALSA (sans vocodeur) : alsa:${dev_in} → alsa:${dev}"
    exec osmo-gapk \
        -a "$dev_in" -f rawpcm-s16le \
        -A "$dev"    -g rawpcm-s16le
}

# ═══════════════════════════════════════════════════════════════════
# stop / status
# ═══════════════════════════════════════════════════════════════════
mode_stop() {
    stop_pid "$GAPK_PID_RX"
    stop_pid "$GAPK_PID_TX"
    rm -f "$GAPK_AUTO_LOCK"
    pkill -x osmo-gapk 2>/dev/null && log_info "Toutes instances arretees" \
        || log_info "Aucune instance active"
}

mode_status() {
    local audio_mode; audio_mode=$(detect_audio_mode)
    echo -e "${CYAN}── Etat audio ──${NC}"
```

```
    echo -e "  Mode mobile.cfg : ${BOLD}${audio_mode}${NC}"
    echo ""

    local any=0
    for f in "$GAPK_PID_RX" "$GAPK_PID_TX"; do
        [ -f "$f" ] || continue
        local pid; pid=$(cat "$f")
        if kill -0 "$pid" 2>/dev/null; then
            local cmdline; cmdline=$(cat "/proc/${pid}/cmdline" 2>/dev/null \
                | tr '\0' ' ' | cut -c1-80 || echo "?")
            echo -e "  ${GREEN}●${NC} PID ${pid}  [$(basename "$f")]  ${cmdline}"
            any=1
        else
            echo -e "  ${RED}✗${NC} PID ${pid} mort"; rm -f "$f"
        fi
    done
    [ -f "$GAPK_AUTO_LOCK" ] && echo -e "  ${CYAN}●${NC} mode AUTO actif" && any=1
    [ $any -eq 0 ] && echo -e "  ${YELLOW}Aucune instance gapk active${NC}"

    if [ "$audio_mode" = "l1phy" ]; then
        echo ""
        echo -e "  ${GREEN}●${NC} Audio MS via l1phy (natif dans mobile process)"
        if pgrep -f "mobile.*mobile" >/dev/null 2>&1; then
            echo -e "  ${GREEN}●${NC} Processus mobile actif"
        else
            echo -e "  ${YELLOW}○${NC} Processus mobile non detecte"
        fi
    fi

    if pgrep -x osmo-gapk >/dev/null 2>&1; then
        echo ""; echo -e "${CYAN}Processus gapk :${NC}"
        ps -o pid,args --no-headers -C osmo-gapk 2>/dev/null | sed 's/^/  /' || true
    fi
}

mode_list_codecs() {
    check_gapk
    # `--list-codecs` n'existe pas dans osmo-gapk : l'option echouait, et seul
    # le `||` de repli produisait la liste - en imprimant d'abord une erreur
    # d'usage qui donnait l'air casse un mode qui marchait. On interroge
    # directement --help, qui est la source de verite des noms.
    echo -e "${CYAN}Codecs et formats supportes (osmo-gapk --help) :${NC}"
    osmo-gapk --help 2>&1 | sed -n '/Supported codecs:/,$p'
}

mode_list_devices() {
    echo -e "${CYAN}── Peripheriques ALSA ──${NC}"
    echo "Lecture (HP) :"
    aplay -L 2>/dev/null | head -20 || echo "  (aplay indisponible)"
    echo ""; echo "Capture (micro) :"
    arecord -L 2>/dev/null | head -20 || echo "  (arecord indisponible)"
    echo ""; echo "Cartes son :"
    cat /proc/asound/cards 2>/dev/null || echo "  (relancer avec --device /dev/snd)"

    local audio_mode; audio_mode=$(detect_audio_mode)
    if [ "$audio_mode" = "l1phy" ]; then
        echo ""
        echo -e "${CYAN}── Config l1phy (mobile.cfg) ──${NC}"
        grep -A3 "tch-voice" /root/.osmocom/bb/mobile.cfg 2>/dev/null | sed 's/^/  /' || true
    fi
}

# ═══════════════════════════════════════════════════════════════
# Usage
# ═══════════════════════════════════════════════════════════════
usage() {
```

```
    local audio_mode; audio_mode=$(detect_audio_mode)
    cat <<EOF
Usage: gapk-start.sh <mode> [options]

  Mode audio mobile.cfg : ${audio_mode}
  $([ "$audio_mode" = "l1phy" ] && echo "  → Audio MS natif via l1phy. gapk = monitoring reseau uniquement." ||
↪  echo "  → Audio MS via gapk (mode classique).")

  info                    Affiche l'architecture audio et les IPs Linphone

  auto     [FORMAT] [ALSA_HP] [ALSA_MIC]
              Surveille MGW, bridge RTP↔ALSA auto (chemin reseau).

  call     <RX_PORT> <TX_DEST:PORT> [FORMAT] [ALSA_HP] [ALSA_MIC]
              Bridge bidirectionnel RTP reseau ↔ ALSA.

  monitor  <RX_PORT> [FORMAT] [ALSA_HP]
              Ecoute passive RTP → ALSA.

  record   <RX_PORT> [FICHIER] [FORMAT]
              Enregistre flux RTP → fichier .gsm.

  playback <FICHIER> <TX_DEST:PORT> [FORMAT]
              Injecte fichier .gsm → RTP.

  loopback [FORMAT] [ALSA_HP] [ALSA_MIC]
              Loopback ALSA mic→HP (ECOUTEURS OBLIGATOIRES).
              Ne traverse PAS le vocodeur - voir mode_loopback.

  mgw-ports     Endpoints RTP OsmoMGW actifs.
  stop          Arrete les instances gapk.
  status        Etat audio complet (l1phy + gapk).
  list-codecs   Codecs disponibles.
  list-devices  Peripheriques ALSA + config l1phy.

Formats : gsm (defaut, = GSM FR / RFC3551) | rawpcm-s16le | rtp-efr
          | rtp-hr-etsi | rtp-hr-ietf | amr-* | ti-* | racal-*
          "gsmfr", "gsmefr", "gsmhr", "pcm8", "pcm16" N'EXISTENT PAS :
          ces noms ont fait mourir gapk au demarrage jusqu'au 10/08/2026.
          Liste faisant foi : gapk-start.sh list-codecs
Codecs  : seul "fr" est encodeur ET decodeur dans ce build.

Variables : GAPK_FORMAT, GAPK_ALSA_DEV (HP), GAPK_ALSA_DEV_IN (micro),
            GAPK_RTP_PT, ALSA_CARD, GAPK_POLL_INTERVAL
EOF
}

# ── Dispatch ──────────────────────────────────────────────────────────
MODE="${1:-help}"; shift 2>/dev/null || true
case "$MODE" in
    info)          mode_info ;;
    auto)          mode_auto "$@" ;;
    call)          mode_call "$@" ;;
    monitor)       mode_monitor "$@" ;;
    record)        mode_record "$@" ;;
    playback)      mode_playback "$@" ;;
    loopback)      mode_loopback "$@" ;;
    mgw-ports)     mode_mgw_ports "" ;;
    stop)          mode_stop ;;
    status)        mode_status ;;
    list-codecs)   mode_list_codecs ;;
    list-devices)  mode_list_devices ;;
    help|-h|--help) usage ;;
    *) log_error "Mode inconnu : $MODE"; echo ""; usage; exit 1 ;;
esac
```

3.340 /opt/GSM/osmo-operator/scripts/ism_payload.py

15884 octets, 401 lignes → 401 lignes

```
#!/usr/bin/env python3
"""
ism_payload.py — le PAYLOAD des SMS sur une liaison radio ISM, le reste par IP.

Idee (option start-native.sh --payload-ism, cf. wiki/Payload-ISM.md) : entre deux
noeuds osmo-operator, la SIGNALISATION (adresses, MSISDN, IMSI, SC-address, SS7,
SIP) continue de passer par IP comme avec --wan. SEUL le contenu du message (le
texte du SMS, extrait du TPDU) voyage sur une liaison radio en bande ISM (433 /
868 / 915 MHz, 2,4 GHz — sans licence). Les deux relais se coordonnent par TCP ;
le TCP ne porte plus le texte mais un JETON de correlation, et le texte arrive
par la radio, estampille du meme jeton.

Ce module ne connait aucun materiel : il parle a l INTERFACE DE DONNEES de ta
radio (un peripherique serie, une paire de tubes nommes, ou l entree/sortie d un
programme que tu fournis — un montage SDR GNU Radio, rtl_433, un module LoRa/
RFM69/CC1101 en UART...). On choisit le backend par variable d environnement ou
par la section [payload] de sms-routing.conf.

Trame (robuste sur un lien bruite, sans dependance hors stdlib) : une ligne JSON
terminee par \\n, payload en hexa, CRC32 :
    {"t":"<jeton>","p":"<hex>","c":"<crc32hex>"}\\n
Une ligne tronquee ou un CRC faux est ignore (la radio peut perdre des octets).

BACKENDS (ISM_LINK / [payload] link=) :
  loopback  tout en memoire, pour un essai sur une seule machine (pas de radio)
  fifo      deux tubes nommes : ISM_TX (ecriture) et ISM_RX (lecture).
            Branche n importe quel outil radio dessus (socat, un flowgraph SDR...)
  serial    un /dev/tty* en lecture/ecriture brute (module ISM en UART).
            Debit pose par le script de mise en place (stty), ISM_SERIAL=/dev/ttyUSB0
  exec      lance une commande TX (trame sur son stdin) et une commande RX
            (trames sur son stdout) : ISM_TX_CMD / ISM_RX_CMD. Le plus general.

Variables : ISM_LINK, ISM_TX, ISM_RX, ISM_SERIAL, ISM_TX_CMD, ISM_RX_CMD,
ISM_PAYLOAD_TTL (secondes, defaut 120), ISM_RECV_TIMEOUT (defaut 30).

Autotest (sans radio) :
    python3 ism_payload.py --selftest
    python3 ism_payload.py --selftest --link fifo --tx /tmp/ism_a --rx /tmp/ism_b
"""

import os
import sys
import time
import json
import zlib
import queue
import shlex
import select
import threading
import subprocess
from typing import Optional, Dict, Tuple, Callable

DEFAULT_TTL = float(os.environ.get("ISM_PAYLOAD_TTL", "120"))
DEFAULT_RECV_TIMEOUT = float(os.environ.get("ISM_RECV_TIMEOUT", "30"))


def _log(msg: str):
    sys.stderr.write("[ism] %s\n" % msg)
    sys.stderr.flush()


def frame_encode(token: str, payload: bytes) -> bytes:
```

```
    """(token, payload) -> une ligne de trame terminee par \\n."""
    hexp = payload.hex()
    crc = "%08x" % (zlib.crc32(payload) & 0xFFFFFFFF)
    obj = {"t": token, "p": hexp, "c": crc}
    return (json.dumps(obj, separators=(",", ":")) + "\n").encode("ascii")


def frame_decode(line: bytes) -> Optional[Tuple[str, bytes]]:
    """Une ligne de trame -> (token, payload), ou None si invalide / CRC faux."""
    try:
        obj = json.loads(line.decode("ascii", "replace").strip())
        token = obj["t"]
        payload = bytes.fromhex(obj["p"])
        if ("%08x" % (zlib.crc32(payload) & 0xFFFFFFFF)) != obj.get("c"):
            _log("trame rejetee : CRC faux (jeton %s)" % token)
            return None
        return token, payload
    except Exception:
        return None


# ============================================================================
# Backends : chacun expose write_frame(bytes) et un generateur read_frames()
# ============================================================================

class _Backend:
    def write_frame(self, data: bytes) -> None:
        raise NotImplementedError

    def read_line(self, timeout: float) -> Optional[bytes]:
        """Une ligne (sans le \\n), ou None si rien avant timeout."""
        raise NotImplementedError

    def close(self) -> None:
        pass


class LoopbackBackend(_Backend):
    """TX ecrit directement dans la file de RX : un essai sans radio."""
    def __init__(self):
        self._q: "queue.Queue[bytes]" = queue.Queue()

    def write_frame(self, data: bytes) -> None:
        for line in data.splitlines():
            if line:
                self._q.put(line)

    def read_line(self, timeout: float) -> Optional[bytes]:
        try:
            return self._q.get(timeout=timeout)
        except queue.Empty:
            return None


class FifoBackend(_Backend):
    """Deux tubes nommes : tx (ecriture, vers la radio) et rx (lecture)."""
    def __init__(self, tx_path: str, rx_path: str):
        self.tx_path = tx_path
        self.rx_path = rx_path
        for p in (tx_path, rx_path):
            if p and not os.path.exists(p):
                os.mkfifo(p, 0o600)
        # Lecture non bloquante : on ouvre le rx en O_RDONLY|O_NONBLOCK et on
        # garde un ecrivain fictif pour ne pas voir EOF a chaque vidage du tube.
        self._rx_fd = os.open(rx_path, os.O_RDONLY | os.O_NONBLOCK)
        try:
```

```
            self._rx_keepalive = os.open(rx_path, os.O_WRONLY | os.O_NONBLOCK)
        except OSError:
            self._rx_keepalive = -1
        self._buf = b""

    def write_frame(self, data: bytes) -> None:
        # O_WRONLY peut bloquer si personne ne lit : ecriture best-effort.
        fd = os.open(self.tx_path, os.O_WRONLY | os.O_NONBLOCK)
        try:
            os.write(fd, data)
        except BlockingIOError:
            _log("fifo tx pleine, trame perdue")
        finally:
            os.close(fd)

    def read_line(self, timeout: float) -> Optional[bytes]:
        if b"\n" in self._buf:
            line, self._buf = self._buf.split(b"\n", 1)
            return line
        r, _, _ = select.select([self._rx_fd], [], [], timeout)
        if not r:
            return None
        try:
            chunk = os.read(self._rx_fd, 4096)
        except BlockingIOError:
            return None
        if not chunk:
            return None
        self._buf += chunk
        if b"\n" in self._buf:
            line, self._buf = self._buf.split(b"\n", 1)
            return line
        return None

    def close(self) -> None:
        for fd in (getattr(self, "_rx_fd", -1), getattr(self, "_rx_keepalive", -1)):
            if fd and fd >= 0:
                try:
                    os.close(fd)
                except OSError:
                    pass


class SerialBackend(_Backend):
    """Un /dev/tty* en brut. Le debit est pose dehors (stty), on lit/ecrit les octets."""
    def __init__(self, dev: str):
        self.dev = dev
        self._fd = os.open(dev, os.O_RDWR | os.O_NONBLOCK | os.O_NOCTTY)
        self._buf = b""

    def write_frame(self, data: bytes) -> None:
        try:
            os.write(self._fd, data)
        except BlockingIOError:
            time.sleep(0.05)
            try:
                os.write(self._fd, data)
            except OSError:
                _log("serie tx occupee, trame perdue")

    def read_line(self, timeout: float) -> Optional[bytes]:
        if b"\n" in self._buf:
            line, self._buf = self._buf.split(b"\n", 1)
            return line
        r, _, _ = select.select([self._fd], [], [], timeout)
        if not r:
```

```
            return None
        try:
            chunk = os.read(self._fd, 4096)
        except BlockingIOError:
            return None
        if chunk:
            self._buf += chunk
        if b"\n" in self._buf:
            line, self._buf = self._buf.split(b"\n", 1)
            return line
        return None

    def close(self) -> None:
        try:
            os.close(self._fd)
        except OSError:
            pass


class ExecBackend(_Backend):
    """Lance un TX (trame sur stdin) et un RX (trames sur stdout) : ton pipeline SDR."""
    def __init__(self, tx_cmd: str, rx_cmd: str):
        self.tx_cmd = tx_cmd
        self.rx_cmd = rx_cmd
        self._rx = subprocess.Popen(shlex.split(rx_cmd), stdout=subprocess.PIPE,
                                    bufsize=0)

    def write_frame(self, data: bytes) -> None:
        # Un TX par trame : simple et sans etat ; la commande emet puis rend la main.
        try:
            p = subprocess.Popen(shlex.split(self.tx_cmd), stdin=subprocess.PIPE)
            p.stdin.write(data)
            p.stdin.close()
            p.wait(timeout=30)
        except Exception as e:
            _log("exec tx echec : %s" % e)

    def read_line(self, timeout: float) -> Optional[bytes]:
        if self._rx.stdout is None:
            return None
        r, _, _ = select.select([self._rx.stdout], [], [], timeout)
        if not r:
            return None
        line = self._rx.stdout.readline()
        return line.rstrip(b"\n") if line else None

    def close(self) -> None:
        try:
            self._rx.terminate()
        except Exception:
            pass


def make_backend(cfg: Dict[str, str]) -> _Backend:
    link = (cfg.get("link") or os.environ.get("ISM_LINK") or "loopback").lower()
    if link == "loopback":
        return LoopbackBackend()
    if link == "fifo":
        tx = cfg.get("tx") or os.environ.get("ISM_TX") or "/tmp/osmo_ism_tx"
        rx = cfg.get("rx") or os.environ.get("ISM_RX") or "/tmp/osmo_ism_rx"
        return FifoBackend(tx, rx)
    if link == "serial":
        dev = cfg.get("serial") or os.environ.get("ISM_SERIAL") or "/dev/ttyUSB0"
        return SerialBackend(dev)
    if link == "exec":
        tx = cfg.get("tx_cmd") or os.environ.get("ISM_TX_CMD")
```

```
        rx = cfg.get("rx_cmd") or os.environ.get("ISM_RX_CMD")
        if not tx or not rx:
            raise ValueError("backend exec : ISM_TX_CMD et ISM_RX_CMD requis")
        return ExecBackend(tx, rx)
    raise ValueError("backend ISM inconnu : %r (loopback, fifo, serial, exec)" % link)


# ============================================================================
# PayloadLink : send(jeton, payload) / recv(jeton, timeout), RX en tache de fond
# ============================================================================

class PayloadLink:
    """La liaison payload ISM, vue par le relais SMS."""
    def __init__(self, cfg: Optional[Dict[str, str]] = None,
                 ttl: float = DEFAULT_TTL,
                 on_payload: Optional[Callable[[str, bytes], None]] = None):
        self.cfg = cfg or {}
        self.ttl = ttl
        self.backend = make_backend(self.cfg)
        self._store: Dict[str, Tuple[bytes, float]] = {}
        self._lock = threading.Lock()
        self._cv = threading.Condition(self._lock)
        self._on_payload = on_payload
        self._running = True
        self._rx = threading.Thread(target=self._rx_loop, daemon=True)
        self._rx.start()
        _log("liaison payload ISM prete (backend %s)"
             % (self.cfg.get("link") or os.environ.get("ISM_LINK") or "loopback"))

    def _rx_loop(self):
        while self._running:
            line = self.backend.read_line(timeout=1.0)
            if not line:
                self._prune()
                continue
            dec = frame_decode(line)
            if not dec:
                continue
            token, payload = dec
            with self._cv:
                self._store[token] = (payload, time.time())
                self._cv.notify_all()
            if self._on_payload:
                try:
                    self._on_payload(token, payload)
                except Exception as e:
                    _log("on_payload : %s" % e)

    def _prune(self):
        now = time.time()
        with self._lock:
            old = [t for t, (_, ts) in self._store.items() if now - ts > self.ttl]
            for t in old:
                del self._store[t]

    def send(self, token: str, payload: bytes) -> None:
        """Emet le payload sur la radio, estampille du jeton."""
        self.backend.write_frame(frame_encode(token, payload))

    def recv(self, token: str, timeout: float = DEFAULT_RECV_TIMEOUT) -> Optional[bytes]:
        """Attend le payload de ce jeton, recu par la radio (None si absent a temps)."""
        deadline = time.time() + timeout
        with self._cv:
            while token not in self._store:
                remaining = deadline - time.time()
                if remaining <= 0:
```

```
                    return None
                self._cv.wait(timeout=remaining)
            payload, _ = self._store.pop(token)
            return payload

    def close(self):
        self._running = False
        self.backend.close()


def new_token() -> str:
    """Un jeton de correlation court et unique (12 hexa)."""
    return os.urandom(6).hex()


# ============================================================================
# Autotest
# ============================================================================

def _selftest(cfg: Dict[str, str]) -> int:
    link = cfg.get("link") or os.environ.get("ISM_LINK") or "loopback"
    print("autotest liaison payload ISM — backend %s" % link)
    if link == "fifo":
        # deux liens croises : A ecrit sur tx et lit sur rx ; un second lien
        # miroir rejoue ce que A emet vers A lui-meme, pour boucler sans radio.
        a = PayloadLink(dict(cfg))
        mirror_cfg = dict(cfg)
        mirror_cfg["tx"], mirror_cfg["rx"] = cfg.get("rx"), cfg.get("tx")
        b = PayloadLink(mirror_cfg)
        links = [a, b]
        tx, rx = a, b
    else:
        link_obj = PayloadLink(dict(cfg))
        links = [link_obj]
        tx = rx = link_obj

    ok = True
    for text in ["Bonjour banc", "éàü accents 2G", "x" * 140]:
        token = new_token()
        payload = text.encode("utf-8")
        tx.send(token, payload)
        got = rx.recv(token, timeout=5)
        good = got == payload
        ok = ok and good
        print("  %s jeton=%s  %r" % ("OK " if good else "KO ", token,
                                      text[:30] + ("…" if len(text) > 30 else "")))
    # CRC : une trame corrompue doit etre rejetee
    bad = frame_encode("deadbeef", b"hello")
    assert frame_decode(bad.replace(b'"c":"', b'"c":"00')) is None
    print("  OK  trame a CRC faux rejetee")
    for l in links:
        l.close()
    print("resultat :", "OK" if ok else "ECHEC")
    return 0 if ok else 1


def main(argv):
    import argparse
    ap = argparse.ArgumentParser(description="Liaison payload ISM")
    ap.add_argument("--selftest", action="store_true", help="autotest (sans radio par defaut)")
    ap.add_argument("--link", help="loopback|fifo|serial|exec")
    ap.add_argument("--tx"); ap.add_argument("--rx")
    ap.add_argument("--serial"); ap.add_argument("--tx-cmd"); ap.add_argument("--rx-cmd")
    a = ap.parse_args(argv)
    cfg = {k: v for k, v in {
        "link": a.link, "tx": a.tx, "rx": a.rx, "serial": a.serial,
```

```
        "tx_cmd": a.tx_cmd, "rx_cmd": a.rx_cmd}.items() if v}
    if a.selftest:
        return _selftest(cfg)
    ap.print_help()
    return 0


if __name__ == "__main__":
    sys.exit(main(sys.argv[1:]))
```

3.341 /opt/GSM/osmo-operator/scripts/l1ctl_bridge.py

13624 octets, 309 lignes → 309 lignes

```
#!/usr/bin/env python3
"""
l1ctl_bridge.py — Pure sercomm ↔ UDP bridge.

The bridge is a translator/forwarder:
  mobile ←L1CTL→ bridge ←sercomm/UART→ firmware ARM (L1 processing)
  BTS ←TRXD/TRXC/CLK→ bridge (built-in TRX server)

The firmware ARM does ALL L1 processing. The bridge just forwards.
Bursts from BTS arrive via TRXD → bridge sends to firmware via sercomm (new DLCI).
TX bursts from firmware arrive via sercomm → bridge sends to BTS via TRXD.
"""

import errno, fcntl, os, select, signal, socket, struct, sys, termios, threading, time

# ═══════════════════════════════════════════════════════════════════════
# Constants
# ═══════════════════════════════════════════════════════════════════════

GSM_HYPERFRAME = 2715648
GSM_FRAME_US   = 4615.0
L1CTL_SOCK     = "/tmp/osmocom_l2"
SERCOMM_FLAG   = 0x7E; SERCOMM_ESCAPE = 0x7D; SERCOMM_XOR = 0x20
DLCI_L1CTL     = 5
DLCI_BURST     = 6   # New DLCI for burst data (firmware ↔ bridge)

# ═══════════════════════════════════════════════════════════════════════
# Helpers
# ═══════════════════════════════════════════════════════════════════════

def udp_bind_connect(bind_port, dst_addr, dst_port):
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    s.bind(("127.0.0.1", bind_port))
    s.connect((dst_addr, dst_port))
    s.setblocking(False); return s

def sercomm_wrap(dlci, payload):
    out = bytearray([SERCOMM_FLAG])
    for b in bytes([dlci, 0x03]) + payload:
        if b in (SERCOMM_FLAG, SERCOMM_ESCAPE):
            out.append(SERCOMM_ESCAPE); out.append(b ^ SERCOMM_XOR)
        else: out.append(b)
    out.append(SERCOMM_FLAG); return bytes(out)

class SercommParser:
    def __init__(self): self.buf = bytearray()
    def feed(self, data):
        self.buf.extend(data); frames = []
        while True:
```

```
            try: s = self.buf.index(SERCOMM_FLAG)
            except ValueError: self.buf.clear(); break
            if s > 0: del self.buf[:s]
            try: e = self.buf.index(SERCOMM_FLAG, 1)
            except ValueError: break
            raw = bytes(self.buf[1:e]); del self.buf[:e+1]
            if not raw: continue
            out = bytearray(); i = 0
            while i < len(raw):
                if raw[i] == SERCOMM_ESCAPE: i += 1; out.append(raw[i] ^ SERCOMM_XOR) if i < len(raw) else None
                else: out.append(raw[i])
                i += 1
            if len(out) >= 2: frames.append((out[0], bytes(out[2:])))
        return frames

class LPReader:
    def __init__(self): self.buf = bytearray()
    def feed(self, data):
        self.buf.extend(data); msgs = []
        while len(self.buf) >= 2:
            ml = struct.unpack(">H", self.buf[:2])[0]
            if len(self.buf) < 2 + ml: break
            msgs.append(bytes(self.buf[2:2+ml])); del self.buf[:2+ml]
        return msgs


# ═══════════════════════════════════════════════════════════════
# BTS TRX Server — built-in virtual TRX for the BTS
# ═══════════════════════════════════════════════════════════════

class BTSTrx:
    """Virtual TRX for the BTS. Provides CLK, TRXC, TRXD."""

    def __init__(self, bts_base=5700):
        self.bts_base = bts_base
        self.clk_sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        self.clk_sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        self.clk_sock.bind(("127.0.0.1", bts_base)); self.clk_sock.setblocking(False)

        self.trxc_sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        self.trxc_sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        self.trxc_sock.bind(("127.0.0.1", bts_base + 1)); self.trxc_sock.setblocking(False)

        self.trxd_sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        self.trxd_sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        self.trxd_sock.bind(("127.0.0.1", bts_base + 2)); self.trxd_sock.setblocking(False)

        self.bts_trxd_addr = None
        self.clk_fn = 0
        self.running = False
        self.stats = {"dl": 0, "ul": 0}

        self._stop = threading.Event()
        self._clk_thread = threading.Thread(target=self._clock_run, daemon=True)
        self._clk_thread.start()
        print(f"bts-trx: listening on {bts_base}-{bts_base+2}", flush=True)

    def _clock_run(self):
        tick_ns = int(GSM_FRAME_US * 1000)
        t_next = time.monotonic_ns()
        while not self._stop.is_set():
            t_next += tick_ns
            dt = t_next - time.monotonic_ns()
            if dt > 0: time.sleep(dt / 1e9)
            elif dt < -tick_ns * 10: t_next = time.monotonic_ns()
            self.clk_fn = (self.clk_fn + 1) % GSM_HYPERFRAME
```

```
            if self.clk_fn % 102 == 0:
                try: self.clk_sock.sendto(f"IND CLOCK {self.clk_fn}\0".encode(),
                                          ("127.0.0.1", self.bts_base + 100))
                except: pass

    def handle_trxc(self):
        try: data, addr = self.trxc_sock.recvfrom(256)
        except: return
        cmd = data.strip(b'\x00').decode(errors='replace')
        parts = cmd.split()
        if len(parts) < 2 or parts[0] != "CMD": return
        verb = parts[1]
        params = " ".join(parts[2:])
        if verb != "SETPOWER":
            print(f"  [TRXC] {cmd}", flush=True)

        if verb == "POWERON":
            self.running = True; rsp = "RSP POWERON 0"
        elif verb == "POWEROFF":
            self.running = False; rsp = "RSP POWEROFF 0"
        elif verb == "SETFORMAT":
            rsp = "RSP SETFORMAT 0"
        elif params:
            rsp = f"RSP {verb} 0 {params}"
        else:
            rsp = f"RSP {verb} 0"
        self.trxc_sock.sendto(rsp.encode(), addr)

    def recv_dl_burst(self):
        """Receive DL burst from BTS. Returns raw TRXD packet or None."""
        try: data, addr = self.trxd_sock.recvfrom(512)
        except: return None
        self.bts_trxd_addr = addr
        self.stats["dl"] += 1
        return data

    def send_ul_burst(self, data):
        """Send UL burst to BTS. Data is raw TRXD packet."""
        if not self.bts_trxd_addr: return
        self.trxd_sock.sendto(data, self.bts_trxd_addr)
        self.stats["ul"] += 1

    def sockets(self):
        return [self.trxc_sock, self.trxd_sock]

    def close(self):
        self._stop.set()
        for s in [self.clk_sock, self.trxc_sock, self.trxd_sock]:
            try: s.close()
            except: pass


# ═══════════════════════════════════════════════════════════════════
# Main — pure forwarder
# ═══════════════════════════════════════════════════════════════════

def send_to_mobile(client, payload):
    if client:
        try: client.sendall(struct.pack(">H", len(payload)) + payload); return True
        except: return False
    return False

def main():
    if len(sys.argv) < 2:
        print(f"Usage: {sys.argv[0]} <uart-sock> [l1ctl-sock] [bts-base]"); sys.exit(1)
```

```
pty_path = sys.argv[1]
global L1CTL_SOCK
if len(sys.argv) > 2 and not sys.argv[2].startswith("-"):
    L1CTL_SOCK = sys.argv[2]
nums = [a for a in sys.argv[2:] if a.isdigit()]
bts_base = int(nums[0]) if len(nums) > 0 else 5700

try: os.unlink(L1CTL_SOCK)
except FileNotFoundError: pass

# QEMU UART
import stat as stat_mod
qemu_sock = None
if os.path.exists(pty_path) and stat_mod.S_ISSOCK(os.stat(pty_path).st_mode):
    qemu_sock = socket.socket(socket.AF_UNIX, socket.SOCK_STREAM)
    qemu_sock.connect(pty_path); qemu_sock.setblocking(False)
    pty_fd = qemu_sock.fileno()
else:
    pty_fd = os.open(pty_path, os.O_RDWR | os.O_NOCTTY)
    attrs = termios.tcgetattr(pty_fd)
    attrs[0] = attrs[1] = attrs[3] = 0
    attrs[2] = termios.CS8 | termios.CLOCAL | termios.CREAD
    attrs[4] = attrs[5] = termios.B115200
    attrs[6][termios.VMIN] = 1; attrs[6][termios.VTIME] = 0
    termios.tcsetattr(pty_fd, termios.TCSANOW, attrs)
    fl = fcntl.fcntl(pty_fd, fcntl.F_GETFL)
    fcntl.fcntl(pty_fd, fcntl.F_SETFL, fl | os.O_NONBLOCK)

# L1CTL server
srv = socket.socket(socket.AF_UNIX, socket.SOCK_STREAM)
srv.bind(L1CTL_SOCK); srv.listen(1); srv.setblocking(False)

# BTS TRX server
trx = BTSTrx(bts_base)

print(f"bridge: uart={pty_path} l1ctl={L1CTL_SOCK} bts={bts_base}", flush=True)
print(f"  sercomm DLCI {DLCI_L1CTL}=L1CTL, DLCI {DLCI_BURST}=bursts", flush=True)

parser = SercommParser(); lp = LPReader()
client = None; running = True
stats = {"l1ctl_tx": 0, "l1ctl_rx": 0, "burst_dl": 0, "burst_ul": 0}

def shutdown(s, f): nonlocal running; running = False
signal.signal(signal.SIGINT, shutdown); signal.signal(signal.SIGTERM, shutdown)

try:
    while running:
        rlist = [srv, pty_fd] + trx.sockets()
        if client: rlist.append(client)
        try: readable, _, _ = select.select(rlist, [], [], 0.5)
        except (OSError, ValueError): break

        # ── Accept mobile ──
        if srv in readable:
            conn, _ = srv.accept(); conn.setblocking(False)
            if client:
                try: client.close()
                except: pass
            client = conn; lp = LPReader()
            print("mobile connected", flush=True)

        # ── L1CTL: mobile → firmware (pure forward via sercomm) ──
        if client and client in readable:
            try: raw = client.recv(4096)
            except BlockingIOError: raw = b""
            if not raw:
```

```
                    print("mobile disconnected", flush=True)
                    try: client.close()
                    except: pass
                    client = None
                else:
                    for payload in lp.feed(raw):
                        stats["l1ctl_tx"] += 1
                        frame = sercomm_wrap(DLCI_L1CTL, payload)
                        try: os.write(pty_fd, frame)
                        except OSError: pass
                        if stats["l1ctl_tx"] <= 10:
                            mt = payload[0] if payload else 0
                            print(f"  [L1CTL→fw] type=0x{mt:02x} len={len(payload)}", flush=True)

            # ── UART: firmware → mobile/bridge ──
            if pty_fd in readable:
                try: raw = os.read(pty_fd, 4096)
                except OSError as e:
                    if e.errno in (errno.EAGAIN, errno.EWOULDBLOCK): raw = b""
                    else: raise
                if raw:
                    for dlci, payload in parser.feed(raw):
                        if dlci == DLCI_L1CTL and client:
                            # L1CTL from firmware → mobile
                            stats["l1ctl_rx"] += 1
                            send_to_mobile(client, payload)
                        elif dlci == DLCI_BURST:
                            # TX burst from firmware → BTS
                            trx.send_ul_burst(payload)
                            stats["burst_ul"] += 1

            # ── BTS TRXC ──
            if trx.trxc_sock in readable:
                trx.handle_trxc()

            # ── BTS TRXD: DL burst → firmware via sercomm (TS0, 1 per multiframe) ──
            if trx.trxd_sock in readable:
                data = trx.recv_dl_burst()
                if data and len(data) >= 6 and client:
                    tn = data[0] & 0x07
                    fn = struct.unpack(">I", data[1:5])[0]
                    mf = fn % 51
                    # Forward FCCH(0,10,20,30,40), SCH(1,11,21,31,41), BCCH(2-5), CCCH(6-9)
                    if tn == 0 and mf <= 9:
                        frame = sercomm_wrap(DLCI_BURST, data)
                        try: os.write(pty_fd, frame)
                        except OSError: pass
                        stats["burst_dl"] += 1

    finally:
        if client: client.close()
        srv.close(); trx.close()
        if qemu_sock: qemu_sock.close()
        else: os.close(pty_fd)
        try: os.unlink(L1CTL_SOCK)
        except: pass
        print(f"done l1ctl={stats['l1ctl_tx']}/{stats['l1ctl_rx']} "
              f"burst={stats['burst_dl']}/{stats['burst_ul']}", flush=True)

if __name__ == "__main__":
    main()
```

3.342 /opt/GSM/osmo-operator/scripts/magnetite-c139.sh

5721 octets, 101 lignes → 101 lignes

```
#!/bin/bash
# magnetite-c139.sh - FreeCalypso Magnetite (cible c139, config hybrid-ui-vo) dans qosmo.
#
# [2026-10-21] Le vrai firmware TI/TCS211 (recompile par fc-magnetite avec le
# compilateur TMS470 sous Wine) a la place de la layer1 osmocom-bb :
#   - image flash 4 MiB = bootloader Compal FreeCalypso en secteur 0 +
#     fwimage.bin a 0x10000 (/opt/GSM/firmware/magnetite/c139-flash.img) ;
#   - qosmo demarre a 0 dans la flash (pas de -kernel), CALYPSO_LCD=c139 met
#     l'ecran couleur 96x64 du C139 sur uWire CS0 (calypso_uwire.c) ;
#   - traces RVTMUX du firmware sur l'UART modem (-serial 0 -> pty), a lire
#     avec rvinterf (/opt/GSM/fc-host-tools) ; UART IrDA sur le 2e pty.
#
#   scripts/magnetite-c139.sh [start|stop|status|log|traces|fsio|shell|gtk]
#     QEMU=/opt/GSM/qosmo/build/qemu-system-arm  FLASH=.../c139-flash.img
#     FLASH_RESET=1 : repartir de l'image neuve (FFS efface)
#     RVINTERF=0 : ne pas lancer rvinterf (traces RVTMUX decodees dans $RUNDIR/rvinterf.log,
#       socket $RUNDIR/rvinterf.sock pour fc-shell / fc-fsio -s)
#     QEMU_DEBUG=1 : -d ${QEMU_DEBUG_FLAGS:-guest_errors,unimp} dans qemu-debug.log
#     QEMU_CPU=ti925t (defaut) : ARMv4T comme l ARM7TDMI du Calypso. Avec arm946
#       (ARMv5) les trampolines Thumb de TI (pop {pc} sans bit 0) partent en ARM
#       -> prefetch abort a la premiere, vu le 21/10.
#     CALYPSO_LCD_DEBUG=1 : commandes LCD sur stderr
set -u
RUNDIR="${RUNDIR:-/dev/shm/magnetite}"
QEMU="${QEMU:-/opt/GSM/qosmo/build/qemu-system-arm}"
FLASH_SRC="${FLASH:-/opt/GSM/firmware/magnetite/c139-flash.img}"
HERE="$(cd "$(dirname "$0")/.." && pwd)"
cmd="${1:-start}"

vivant() { [ -f "$RUNDIR/qemu.pid" ] && kill -0 "$(cat "$RUNDIR/qemu.pid")" 2>/dev/null; }

case "$cmd" in
start)
    if vivant; then echo "deja lance (pid $(cat "$RUNDIR/qemu.pid"))"; exit 0; fi
    [ -x "$QEMU" ] || { echo "QEMU absent : $QEMU" >&2; exit 1; }
    [ -f "$FLASH_SRC" ] || { echo "image flash absente : $FLASH_SRC (voir doc/magnetite.md)" >&2; exit 1; }
    mkdir -p "$RUNDIR"
    # copie de travail : le firmware ecrit son FFS dans la flash (0x3C0000, 3 x 64 KiB),
    # gardee d'un lancement a l'autre (IMEI, rfcap, config) ; FLASH_RESET=1 ou une
    # image source plus recente (nouveau build) la remplacent.
    if [ "${FLASH_RESET:-0}" != 0 ] || [ ! -f "$RUNDIR/flash.img" ] || [ "$FLASH_SRC" -nt "$RUNDIR/flash.img" ];
     ↪  then
        cp "$FLASH_SRC" "$RUNDIR/flash.img" && echo "flash de travail (re)creee depuis $FLASH_SRC"
    fi
    dbg=()
    [ "${QEMU_DEBUG:-0}" != 0 ] && dbg=(-d "${QEMU_DEBUG_FLAGS:-guest_errors,unimp}" -D "$RUNDIR/qemu-debug.log")
    # QEMU_GDB=1 : stub gdb sur tcp::${QEMU_GDB_PORT:-1240} (gdb-multiarch, symboles : adresses de fwimage.map)
    [ "${QEMU_GDB:-0}" != 0 ] && dbg+=(-gdb "tcp::${QEMU_GDB_PORT:-1240}")
    [ "${QEMU_GDB_WAIT:-0}" != 0 ] && dbg+=(-S)     # CPU gele au depart, gdb fait « continue »
    [ -n "${QEMU_TRACE:-}" ] && dbg+=(-trace "$QEMU_TRACE")   # ex. QEMU_TRACE='pflash*' (dans qemu.log)
    CALYPSO_PACER_RATTRAPAGE="${CALYPSO_PACER_RATTRAPAGE:-0}" \
    CALYPSO_LCD=c139 CALYPSO_LCD_SHM="${CALYPSO_LCD_SHM:-/dev/shm/calypso_lcd}" \
    CALYPSO_KEYS_SOCK="${CALYPSO_KEYS_SOCK:-/tmp/calypso_keys.sock}" \
    "$QEMU" -M calypso -cpu "${QEMU_CPU:-ti925t}" -display none -parallel none \
        -serial pty -serial pty -monitor "unix:$RUNDIR/monitor.sock,server,nowait" \
        -drive if=pflash,file="$RUNDIR/flash.img",format=raw "${dbg[@]}" \
        > "$RUNDIR/qemu.log" 2>&1 &
    echo $! > "$RUNDIR/qemu.pid"
    for i in 1 2 3 4 5 6 7 8 9 10; do
        grep -q 'char device redirected' "$RUNDIR/qemu.log" 2>/dev/null && break; sleep 0.3
    done
    grep -o '/dev/pts/[0-9]*' "$RUNDIR/qemu.log" | head -1 > "$RUNDIR/modem.pty"
```

```
    grep -o '/dev/pts/[0-9]*' "$RUNDIR/qemu.log" | sed -n 2p > "$RUNDIR/irda.pty"
    RVINTERF_BIN="${RVINTERF_BIN:-/opt/GSM/fc-host-tools/rvinterf/lowlevel/rvinterf}"
    if [ "${RVINTERF:-1}" != 0 ] && [ -x "$RVINTERF_BIN" ] && [ -s "$RUNDIR/modem.pty" ]; then
        rm -f "$RUNDIR/rvinterf.sock"
        "$RVINTERF_BIN" -n -l "$RUNDIR/rvinterf.log" -s "$RUNDIR/rvinterf.sock" "$(cat "$RUNDIR/modem.pty")" \
            > "$RUNDIR/rvinterf.out" 2>&1 &
        echo $! > "$RUNDIR/rvinterf.pid"
    fi
    echo "qemu pid $(cat "$RUNDIR/qemu.pid")  UART modem (RVTMUX) $(cat "$RUNDIR/modem.pty")  IrDA $(cat
     ↪  "$RUNDIR/irda.pty")"
    echo "journal : $RUNDIR/qemu.log ; ecran : /usr/bin/python3 $HERE/tools/osmo-c123-gtk.py --modele c139"
    ;;
stop)
    vivant && kill "$(cat "$RUNDIR/qemu.pid")" && echo "arrete"
    [ -f "$RUNDIR/rvinterf.pid" ] && kill "$(cat "$RUNDIR/rvinterf.pid")" 2>/dev/null
    rm -f "$RUNDIR/qemu.pid" "$RUNDIR/rvinterf.pid"
    ;;
traces)
    tail -n "${2:-40}" "$RUNDIR/rvinterf.log"
    ;;
fsio|shell)
    # fc-fsio / fc-shell de FreeCalypso sur le socket de rvinterf
    bin="/opt/GSM/fc-host-tools/rvinterf/$([ "$cmd" = fsio ] && echo etmsync/fc-fsio || echo asyncshell/fc-shell)"
    exec "$bin" -s "$RUNDIR/rvinterf.sock" "${@:2}"
    ;;
status)
    if vivant; then echo "en marche (pid $(cat "$RUNDIR/qemu.pid")), modem $(cat "$RUNDIR/modem.pty"
     ↪  2>/dev/null)"; else echo "arrete"; fi
    ;;
log)
    tail -n "${2:-40}" "$RUNDIR/qemu.log"
    ;;
gtk)
    # [2026-10-21] --lcd/--keys explicites : sans eux la fenetre prend la premiere instance vivante non-TRX
    # (rssi, instance 4) et affiche son ecran au lieu de Magnetite.
    exec /usr/bin/python3 "$HERE/tools/osmo-c123-gtk.py" --modele c139 \
        --lcd "${CALYPSO_LCD_SHM:-/dev/shm/calypso_lcd}" --keys "${CALYPSO_KEYS_SOCK:-/tmp/calypso_keys.sock}"
         ↪  "${@:2}"
    ;;
*)
    echo "usage : $0 start|stop|status|log|traces|fsio [cmd]|shell [cmd]|gtk" >&2; exit 2
    ;;
esac
```

3.343 /opt/GSM/osmo-operator/scripts/osmo-banc-aliases.sh

4786 octets, 56 lignes → 55 lignes (1 groupes compactés)

```
# osmo-banc-aliases.sh - les alias du banc Calypso (a sourcer : /etc/profile.d/98-osmo-banc-aliases.sh)
# [2026-10-19] Deux chaines, quatre processus chacune, par c54x_exe/run.sh (MODE=dsp) :
#   banc-trx     c54x_exe + qosmo (trx.highram.elf) + osmocon + transceiver  : le Calypso est le
#                TRX de la BTS#0 (TRXD 5700-5702) ; pas de pont, il tiendrait les memes ports
#   banc-mobile  INSTANCE 2 : c54x_exe + qosmo (layer1) + osmocon + mobile, et l'AIR vers l'instance 1
#                (banc-mobile-stop/-status/-logs pour cette instance ; c123-1 pour son ecran)
#   banc-mobile-seul  l'ancien montage : mobile + pont vers osmo-bts-trx, instance 1
#   banc-rssi    c54x_exe + qosmo (rssi.highram.elf) + osmocon : l'appli RSSI seule, ecran et
#                clavier emules ; la fenetre : c123
#   banc-stop / banc-status / banc-logs   le meme run.sh, --stop / --status / --logs
# Le reseau (coeur, BSC, BTS, side-car) reste a ./start-native.sh ; ces alias ne montent que
# la chaine Calypso, le reseau etant deja debout.
# [2026-10-20] Tout d'un coup, reseau compris : ./start-native.sh --bts (la radio seule, = --osmo-trx)
#   ou ./start-native.sh --bts --ms (la radio puis le mobile par l'air, = banc-trx puis banc-mobile).
alias banc-trx='MODE=dsp OSMO_TRX=1 bash /opt/GSM/c54x_exe/run.sh'
# [2026-10-19] banc-mobile = INSTANCE 2 (run.sh INSTANCE=2) : deuxieme c54x_exe, QEMU, osmocon, mobile,
```

```
# et l'air (pont/air.py) qui relie ses bursts a ceux de l'instance 1 (banc-trx). Journaux : /tmp/c54x-pont2.
alias banc-mobile='INSTANCE=2 MODE=dsp AIR=1 bash /opt/GSM/c54x_exe/run.sh'
alias banc-mobile-stop='INSTANCE=2 MODE=dsp bash /opt/GSM/c54x_exe/run.sh --stop'
alias banc-mobile-status='INSTANCE=2 MODE=dsp bash /opt/GSM/c54x_exe/run.sh --status'
alias banc-mobile-logs='INSTANCE=2 MODE=dsp bash /opt/GSM/c54x_exe/run.sh --logs'
# l'ancien montage a un seul Calypso (mobile + pont vers osmo-bts-trx), instance 1
alias banc-mobile-seul='MODE=dsp PONT=1 bash /opt/GSM/c54x_exe/run.sh'
# [2026-10-21] rssi sur l'INSTANCE 4, SEPAREE du reste du banc : ni environnement copie de l'instance 1
# (le TRX), ni arret par banc-stop / start-native --stop (banc_dsp_arreter_instances l'epargne).
# banc-rssi-stop pour l'arreter seul ; c123-3 pour son ecran.
unalias banc-rssi 2>/dev/null   # l'ancien alias (meme nom) s'expanserait a la definition de la fonction
# [2026-10-23] En BTS CLASSIQUE (--bts-classic : pont_dsp.py vivant), pas d'air : il relaierait l'anneau TX de
# l'instance 1, qui est alors le MOBILE (rien en veille, d'ou « Sync error ») et renverrait les bursts de rssi
# vers son BSP. Le descendant arrive par la copie du pont (PONT_DSP_COPIES, udp 6762). Sinon (--bts), l'air.
# Air realiste (calypso_bsp.c, CALYPSO_AIR_NIVEAUX) : la cellule n'est entendue que sur son ARFCN (plan
# radio), le descendant a CALYPSO_AIR_DL_DBM, le montant du mobile a CALYPSO_AIR_UL_DBM, ailleurs du bruit.
# CALYPSO_AIR_NIVEAUX=0 banc-rssi : l'ancien comportement (tout au maximum, sur toutes les frequences).
function banc-rssi {
    local air=1 arfcn
    pgrep -f 'pont/pont_dsp\.py' >/dev/null && air=0
    arfcn="$( . /etc/osmocom/radio-plan.env 2>/dev/null; echo "${PLAN_ARFCN:-}")"
    INSTANCE=4 MODE=dsp FIRMWARE_APP=rssi AIR="${AIR:-$air}" \
        CALYPSO_AIR_NIVEAUX="${CALYPSO_AIR_NIVEAUX:-1}" CALYPSO_FAUX_AIR="${CALYPSO_FAUX_AIR:-$arfcn}" \
        bash /opt/GSM/c54x_exe/run.sh "$@"
}
# [2026-10-23] banc-rssi --stop (ou banc-rssi-stop) arrete TOUTE l'instance 4, restes compris (run.sh arreter).
 ↪  Une
# fonction et plus un alias : un alias n'existe pas dans un shell non interactif (bash -lc), l'arret n'y faisait
 ↪  rien.
unalias banc-rssi-stop 2>/dev/null
function banc-rssi-stop { banc-rssi --stop; }
alias banc-stop='MODE=dsp bash /opt/GSM/c54x_exe/run.sh --stop'
alias banc-status='MODE=dsp OSMO_TRX=1 bash /opt/GSM/c54x_exe/run.sh --status'
alias banc-logs='MODE=dsp bash /opt/GSM/c54x_exe/run.sh --logs'
# [2026-10-21] numerotation demandee par le banc : c123 = TRX (instance 1, TS0), c123-1 = layer1 (le mobile,
# instance 2), c123-2 = trx2 (telephone 2 du TRX, TS1, instance 3), c123-3 = rssi (instance 4, banc-rssi).
alias c123='/usr/bin/python3 /opt/GSM/osmo-operator/tools/osmo-c123-gtk.py --instance 1'
alias c123-⟨1⟩='/usr/bin/python3 /opt/GSM/osmo-operator/tools/osmo-c123-gtk.py --instance ⟨2⟩'  ×3
    ⟨⟩ = (1,2) (2,3) (3,4)
alias c123-mobile='/usr/bin/python3 /opt/GSM/osmo-operator/tools/osmo-c123-gtk.py'   # la premiere instance
 ↪  vivante qui n'est pas un TRX
alias c123-tous='/usr/bin/python3 /opt/GSM/osmo-operator/tools/osmo-c123-gtk.py --toutes 1,2,3,4'   # + celles
 ↪  qui apparaissent ensuite
```

3.344 /opt/GSM/osmo-operator/scripts/osmo-disclaimer.sh

1258 octets, 18 lignes → 18 lignes

```
# osmo-disclaimer.sh — rappel affiche a l'ouverture d'un shell dans le conteneur.
#
# SOURCE, pas execute : /root/.bashrc fait `. /etc/osmo-disclaimer.sh` (symlink
# pose par Dockerfile.run vers ce fichier). Le pendant ISO est
# /etc/profile.d/01-osmo-disclaimer.sh, ecrit par build-iso.sh.
#
# Ce texte vivait dans un heredoc de Dockerfile.run. Le builder docker classique
# (Step N/M, sans BuildKit) ne connait pas les heredocs : il joignait les lignes
# de continuation et lancait le RUN sans le corps, d'ou
#   « here-document at line 1 delimited by end-of-file (wanted `DISC') »
# et un /etc/osmo-disclaimer.sh vide. Un fichier du depot copie par
# `COPY scripts/* /etc/osmocom/` marche avec les deux builders.
export OSMO_DISCLAIMER_SHOWN=1
printf "\n  \033[1;33mDisclaimer\033[0m - banc d'essai GSM/SS7 Osmocom (conteneur %s).\n" "$(hostname)"
printf "  A n'utiliser que sur un reseau radio \033[1mISOLE\033[0m (cage/attenuateur) ou\n"
printf "  sur une bande sous licence : emettre sur le spectre public est illegal.\n"
```

```
printf "  \033[1;33mDemarrer la pile radio :\033[0m cd /opt/GSM/osmo-operator && ./start-native.sh --force\n"
printf "  \033[0;36mSuivre :\033[0m tmux attach -t calypso   \033[0;36m(Ctrl-b puis d pour se detacher)\033[0m\n\n"
```

3.345 /opt/GSM/osmo-operator/scripts/osmo-gnuarm-profile.sh

832 octets, 14 lignes → 14 lignes

```
# osmo-gnuarm-profile.sh - la chaine gnuarm d osmocom-bb dans le PATH (a sourcer :
↪  /etc/profile.d/97-osmo-gnuarm.sh)
# [2026-10-24] arm-elf-gcc 4.5.2 (binutils, newlib) vit dans /root/gnuarm/install/bin, pose par le
# supplement --gnuarm (tools/osmo-firmwares-install.sh, copie de l image osmocom-bb-trx). Les
# compilations du supplement passaient deja ce PATH a make ; ici il vaut pour tout shell de
# connexion : « cd osmocom-bb/src && make firmware » trouve arm-elf-gcc sans rien exporter.
# GNUARM_DIR deplace la chaine. Sans le repertoire (supplement non pose), rien ne change.
_osmo_gnuarm_bin="${GNUARM_DIR:-/root/gnuarm}/install/bin"
if [ -d "$_osmo_gnuarm_bin" ]; then
    case ":$PATH:" in
        *":$_osmo_gnuarm_bin:"*) ;;
        *) PATH="$PATH:$_osmo_gnuarm_bin"; export PATH ;;
    esac
fi
unset _osmo_gnuarm_bin
```

3.346 /opt/GSM/osmo-operator/scripts/osmo-start.sh

7533 octets, 159 lignes → 159 lignes

```
#!/bin/bash
# osmo-start.sh - Demarrage sequence du Core Osmocom
#
# Ordre de dependance :
#   STP (4239) → HLR (4258) → MGW (4243)
#                    ↓
#               MSC (4254) → BSC (4242)
#                    ↓
#          GGSN (4260) → SGSN (4245) → PCU (4239 bts)
#
# Chaque service attend le VTY du precedent avant de demarrer.
# BTS-TRX et SIP-connector sont geres par run.sh (dependent de fake_trx / Asterisk).
#
# Appele par : run.sh (etape 3)

set -e

GREEN='\033[0;32m'; RED='\033[0;31m'; YELLOW='\033[1;33m'
CYAN='\033[0;36m'; NC='\033[0m'

OPERATOR_ID="${OPERATOR_ID:-1}"

# ── Helper : attendre qu'un port TCP soit ouvert ──────────────────────
wait_port() {
    local host="$1" port="$2" label="$3" timeout="${4:-30}"
    local elapsed=0
    echo -ne "  Attente ${label} (${host}:${port})"
    while ! bash -c "echo >/dev/tcp/${host}/${port}" 2>/dev/null; do
        sleep 1; elapsed=$((elapsed + 1))
        echo -n "."
        if [ "$elapsed" -ge "$timeout" ]; then
            echo -e " ${RED}TIMEOUT${NC}"
            return 1
        fi
    done
```

```
    echo -e " ${GREEN}OK${NC} (${elapsed}s)"
}

# ── Helper : demarrer un service et verifier ─────────────────────────
start_svc() {
    local svc="$1" vty_port="$2" label="$3" timeout="${4:-30}"

    echo -e "  ${CYAN}${label}${NC}"
    # ── reset-failed AVANT start ─────────────────────────────────────
    # systemd abandonne un service apres StartLimitBurst redemarrages (compteur
    # a 10 sur osmo-bsc). Une config refusee suffit a l'y amener en dix
    # secondes. Ensuite - et c'est le piege - REPARER LA CONFIG NE SUFFIT PLUS :
    # "systemctl start" rend "start request repeated too quickly" et le service
    # reste mort, sans que rien ne le relie a l'erreur d'origine. C'est ce qui
    # a laisse le BSC de l'operateur 1 eteint alors que son fichier etait
    # redevenu valide. On efface donc le compteur a chaque lancement : ce script
    # EST le geste de demarrage, il ne doit pas heriter des echecs d'hier.
    systemctl reset-failed "$svc" 2>/dev/null || true
    systemctl start "$svc" || {
        echo -e "    ${RED}✗ systemctl start ${svc} echoue${NC}"
        journalctl -u "$svc" -n 20 --no-pager >&2
        return 1
    }

    if [ -n "$vty_port" ]; then
        wait_port 127.0.0.1 "$vty_port" "$label VTY" "$timeout" || {
            echo -e "    ${YELLOW}[WARN] VTY :${vty_port} non accessible apres ${timeout}s${NC}" >&2
            journalctl -u "$svc" -n 10 --no-pager >&2
            return 1
        }
    fi
}

# ═══════════════════════════════════════════════════════════════════════

echo -e "${GREEN}=== Core Osmocom - Op${OPERATOR_ID} ===${NC}"
echo ""

# ── 1. Reseau TUN (APN0 pour GGSN) ──────────────────────────────────
echo -e "${CYAN}[1/4] Interface TUN${NC}"
if ip link show apn0 > /dev/null 2>&1; then
    ip link del dev apn0
fi
ip tuntap add dev apn0 mode tun
ip addr add 176.16.32.0/24 dev apn0
ip link set dev apn0 up
echo -e "  ${GREEN}✓${NC} apn0 up"

# ── 2. Signalisation (STP → HLR → MGW) ──────────────────────────────
#
# STP doit etre pret en premier : tout le SS7 passe par lui.
# HLR doit etre pret avant MSC : MSC se connecte au HLR au demarrage.
# MGW doit etre pret avant MSC : MSC ouvre un MGCP vers MGW.
# ═══════════════════════════════════════════════════════════════════════
echo ""
echo -e "${CYAN}[2/4] Signalisation (STP → HLR → MGW)${NC}"

start_svc osmo-stp  4239 "OsmoSTP"  30 || true
start_svc osmo-hlr  4258 "OsmoHLR"  30 || {
    echo -e "  ${RED}[ERR] HLR indispensable pour MSC - abandon${NC}"
    exit 1
}
start_svc osmo-mgw  4243 "OsmoMGW"  20 || true

# ── 3. Core Network (MSC → BSC) ─────────────────────────────────────
#
```

```
# MSC doit etre pret avant BSC : BSC se connecte au MSC via A-interface.
# ═══════════════════════════════════════════════════════════════════════
echo ""
echo -e "${CYAN}[3/4] Core Network (MSC → BSC)${NC}"

start_svc osmo-msc  4254 "OsmoMSC"  30 || true
start_svc osmo-bsc  4242 "OsmoBSC"  30 || true

# ── 4. Data (GGSN → SGSN → PCU) ──────────────────────────────────────
echo ""
echo -e "${CYAN}[4/4] Data (GGSN → SGSN → PCU)${NC}"

start_svc osmo-ggsn 4260 "OsmoGGSN" 20 || true
start_svc osmo-sgsn 4245 "OsmoSGSN" 20 || true
start_svc osmo-pcu  ""   "OsmoPCU"  0  || true
start_svc osmo-sip-connector  ""   "OsmoSIPConnector"  0  || true
sleep 0.5
chmod 777 /tmp/pcu_bts 2>/dev/null || true

# NOTE : osmo-bts-trx et osmo-sip-connector sont intentionnellement
# absents ici. run.sh les demarre dans l'ordre correct :
#   fake_trx → wait_udp 5700 → osmo-bts-trx  (evite la race condition TRX)
#   Asterisk → osmo-sip-connector             (evite le MNCC connect avant SIP UP)

# ── Resume ────────────────────────────────────────────────────────────
echo ""
echo -e "${GREEN}=== Verification ===${NC}"
SERVICES="osmo-stp osmo-hlr osmo-mgw osmo-msc osmo-bsc osmo-ggsn osmo-sgsn osmo-pcu"
MORTS=""
for svc in $SERVICES; do
    if systemctl is-active --quiet "$svc"; then
        echo -e "  ${GREEN}✓${NC} ${svc}"
    else
        echo -e "  ${RED}✗${NC} ${svc}"
        MORTS="${MORTS} ${svc}"
    fi
done

echo ""
# Chaque start_svc porte un "|| true" : c'est voulu, un GGSN absent ne doit pas
# emporter le reste. Mais la ligne finale disait "Core Osmocom pret" meme avec
# osmo-msc et osmo-bsc en croix juste au-dessus - et c'est cette ligne-la qu'on
# lit. Un coeur sans MSC ne fait pas d'attachement : autant le dire ici plutot
# que de le decouvrir une heure plus tard dans "show bts".
if [ -n "$MORTS" ]; then
    echo -e "${RED}Coeur INCOMPLET -${NC}${MORTS}"
    for svc in $MORTS; do
        echo -e "  ${CYAN}journalctl -u ${svc} -n 30${NC}   ou   ${CYAN}tail -30 /var/log/osmocom/${svc}.log${NC}"
    done
    case "$MORTS" in
        *osmo-msc*|*osmo-bsc*)
            echo -e "  ${YELLOW}Sans MSC ni BSC, aucun mobile ne peut s'attacher.${NC}"
            echo -e "  ${YELLOW}Cause frequente : la config est refusee a la lecture. La ligne fautive${NC}"
            echo -e "  ${YELLOW}est nommee par :${NC} ${CYAN}osmo-msc -c /etc/osmocom/osmo-msc.cfg 2>&1 | grep -A2
             ↪  'Error occurred'${NC}" ;;
    esac
else
    echo -e "${GREEN}Core Osmocom pret. BTS et SIP connector geres par run.sh.${NC}"
fi
```

3.347 /opt/GSM/osmo-operator/scripts/osmo-wg-profile.sh

971 octets, 16 lignes → 16 lignes

```
# osmo-wg-profile.sh - « wg 10.99.0.N » pour monter le tunnel WAN (a sourcer : /etc/profile.d/97-osmo-wg.sh)
# [2026-10-24] Le plan WireGuard du banc (wiki/WAN.md) : 10.99.0.1 le hub, 10.99.0.N les noeuds.
#   wg 1                                ce noeud devient le hub (10.99.0.1)
#   wg N HUB_IP[:PORT] HUB_PUB          ce noeud devient 10.99.0.N et rejoint le hub
#   wg peer CLE_PUBLIQUE N              (sur le hub) accepte un noeud
#   wg up / wg down / wg status / wg pub le tunnel deja pose
# N ou 10.99.0.N, au choix.
# Ces formes vont a tools/osmo-wg.sh (en root, par sudo) ; tout le reste (wg show, wg genkey,
# wg set...) va au vrai wg, inchange. Une fonction, pas un lien dans le PATH : un « wg » dans
# /usr/local/bin masquerait /usr/bin/wg pour les scripts.
wg() {
    case "${1:-}" in
        10.99.0.*|peer|up|down|status|pub|[1-9]|[1-9][0-9]|[12][0-9][0-9])
         ↪  /opt/GSM/osmo-operator/tools/osmo-wg.sh "$@" ;;
        *) command wg "$@" ;;
    esac
}
```

3.348 /opt/GSM/osmo-operator/scripts/pulse-gsm-setup.sh

4131 octets, 106 lignes → 106 lignes

```
#!/bin/bash
# pulse-gsm-setup.sh - Configure PulseAudio for osmo-gapk audio
#
# Chaine audio :
#   osmo-gapk → ALSA "gsm_out" → asound.conf(type pulse, device gsm_audio)
#             → PulseAudio null-sink "gsm_audio"
#             → gsm_audio.monitor → parec → WebSocket → navigateur
#
# Appele par run.sh avant le lancement d'OsmocomBB/gapk.
# Requiert : PULSE_SERVER defini + socket monte par start-docker.sh

set -u

TAG="[pulse-gsm]"
TIMEOUT="${PULSE_TIMEOUT:-30}"
SINK_NAME="gsm_audio"
# [2026-08-08] Sink silencieux servant de MICRO aux mobiles. Sans lui, asound.conf
# fait retomber la capture sur la source par defaut (= gsm_audio.monitor), donc
# sur la sortie que les mobiles viennent d'ecrire : boucle audio fermee.
MIC_NAME="gsm_mic"
SINK_RATE=8000
SINK_CHANNELS=1
SINK_FORMAT="s16le"

log()  { echo "$TAG $*"; }
err()  { echo "$TAG ERROR: $*" >&2; }

# ── 1. Attente PulseAudio ──────────────────────────────────────────
wait_pulse() {
    local i=0
    while [ $i -lt "$TIMEOUT" ]; do
        if pactl info >/dev/null 2>&1; then
            return 0
        fi
        sleep 1
        i=$((i + 1))
    done
    return 1
}

log "Attente connexion PulseAudio (${PULSE_SERVER:-non defini})..."
```

```
if ! wait_pulse; then
    err "PulseAudio injoignable apres ${TIMEOUT}s"
    err "Verifier que PULSE_SERVER est defini et le socket monte"
    exit 1
fi

log "PulseAudio connecte"

# ── 2. Creation du null-sink gsm_audio ──────────────────────────────
if pactl list short sinks 2>/dev/null | grep -q "$SINK_NAME"; then
    log "Sink ${SINK_NAME} deja present"
else
    if pactl load-module module-null-sink \
        sink_name="$SINK_NAME" \
        format="$SINK_FORMAT" \
        rate="$SINK_RATE" \
        channels="$SINK_CHANNELS" \
        sink_properties=device.description=GSM_Audio >/dev/null 2>&1; then
        log "Sink ${SINK_NAME} cree (${SINK_FORMAT}/${SINK_RATE}Hz/${SINK_CHANNELS}ch)"
    else
        err "Echec creation sink ${SINK_NAME}"
        exit 1
    fi
fi

# ── 2bis. Sink silencieux "gsm_mic" (micro des mobiles, cf. asound.conf) ────
if pactl list short sinks 2>/dev/null | grep -q "$MIC_NAME"; then
    log "Sink ${MIC_NAME} deja present"
else
    pactl load-module module-null-sink sink_name="$MIC_NAME" \
        format="$SINK_FORMAT" rate="$SINK_RATE" channels="$SINK_CHANNELS" \
        sink_properties=device.description=GSM_Mic >/dev/null \
        && log "Sink ${MIC_NAME} cree (micro silencieux)" \
        || err "Echec creation sink ${MIC_NAME}"
fi

# ── 3. Verification monitor ─────────────────────────────────────────
if pactl list short sources 2>/dev/null | grep -q "${SINK_NAME}.monitor"; then
    log "Source ${SINK_NAME}.monitor disponible"
else
    err "Source ${SINK_NAME}.monitor introuvable"
    exit 1
fi

# ── 4. Empecher la suspension automatique ───────────────────────────
# PulseAudio suspend les sinks inactifs → osmo-gapk ecrit dans le vide
# et il faut toggle les haut-parleurs sur l'hote pour reveiller le flux

pactl suspend-sink "$SINK_NAME" 0 2>/dev/null || true
pactl suspend-source "${SINK_NAME}.monitor" 0 2>/dev/null || true

# Desactiver module-suspend-on-idle s'il est charge
SUSPEND_IDX=$(pactl list short modules 2>/dev/null | awk '/module-suspend-on-idle/ {print $1; exit}')
if [ -n "$SUSPEND_IDX" ]; then
    pactl unload-module "$SUSPEND_IDX" 2>/dev/null && \
        log "module-suspend-on-idle desactive" || true
fi

log "Sink ${SINK_NAME} force actif (pas de suspend)"

log "Configuration audio prete"
# NOTE : pas de loopback gsm_audio → haut-parleurs.
# Ecoute via parec → WebSocket → navigateur uniquement.
# Sink maintenu actif par suspend-sink 0 + suppression module-suspend-on-idle (etape 4).
```

3.349 /opt/GSM/osmo-operator/scripts/run.sh

29845 octets, 548 lignes → 548 lignes

```
#!/bin/bash
# run.sh - Orchestrateur intra-container Osmocom
#
# PHY_MODE=faketrx (defaut) : fake_trx → trxcon → mobile
# PHY_MODE=virtphy          : osmo-bts-virtual → virtphy → mobile
#
set -euo pipefail

LOG_FILE="/var/log/osmocom/run.sh.log"
mkdir -p "$(dirname "$LOG_FILE")"
exec > >(tee -a "$LOG_FILE") 2>&1

[[ -n "${DEBUG:-}" ]] && set -x

GREEN='\033[0;32m'; CYAN='\033[0;36m'; YELLOW='\033[1;33m'; RED='\033[0;31m'; NC='\033[0m'

# Ce fichier est DEPLOYE HORS DEPOT (/etc/osmocom/run.sh, /root/run.sh) et lance
# par des scripts qui n'exportent pas forcement le manifeste : sous `set -u`, un
# ${GSM_ROOT} nu tuerait le script ici. D'ou les valeurs de repli explicites.
FAKETRX_PY="${FAKETRX_PY:-${FAKE_TRX:-${OSMOCOM_BB:-${GSM_ROOT:-/opt/GSM}/osmocom-bb}
↪  /src/target/trx_toolkit/fake_trx.py}}"
OPERATOR_ID="${OPERATOR_ID:-1}"; N_MS="${N_MS:-1}"; MOBILE_MODE="${MOBILE_MODE:-combined}"
PHY_MODE="${PHY_MODE:-faketrx}"   # faketrx | virtphy
# RUN_NO_PROCESS=1 : prepare/genere uniquement les configs (MS, TRX, handover)
# et sort SANS lancer aucun process (ni osmo-start, ni fake_trx/trxcon, ni
# mobile/asterisk/smsc). Utilise par le mode QEMU de start-docker.sh qui veut juste
# les configs en place avant de lancer ${OQC_ROOT}/run.sh.
RUN_NO_PROCESS="${RUN_NO_PROCESS:-0}"
MAX_MS=64; MAX_MS_PER_MOBILE=8; MS_PER_TRX=16
BB_PORT_BASE=6700; BB_PORT_STEP=3; BTS_PORT_BASE=5700
L2_SOCK_BASE="/tmp/osmocom_l2"; SAP_SOCK_BASE="/tmp/osmocom_sap"

[[ ! "$N_MS" =~ ^[0-9]+$ ]] || [ "$N_MS" -lt 1 ] || [ "$N_MS" -gt $MAX_MS ] && N_MS=1
[[ ! "$OPERATOR_ID" =~ ^[0-9]+$ ]] || [ "$OPERATOR_ID" -lt 1 ] || [ "$OPERATOR_ID" -gt 24 ] && exit 1
[[ "$MOBILE_MODE" != "combined" && "$MOBILE_MODE" != "split" ]] && MOBILE_MODE="combined"
[[ "$PHY_MODE" != "faketrx" && "$PHY_MODE" != "virtphy" ]] && PHY_MODE="faketrx"

N_TRX=$(( (N_MS + MS_PER_TRX - 1) / MS_PER_TRX ))
[ "$MOBILE_MODE" = "combined" ] && N_GROUPS=$(( (N_MS + MAX_MS_PER_MOBILE - 1) / MAX_MS_PER_MOBILE )) ||
↪  N_GROUPS=$N_MS

echo -e "  ${CYAN}Op${OPERATOR_ID}${NC}  N_MS=${N_MS}  mode=${MOBILE_MODE}  phy=${PHY_MODE}  TRX=${N_TRX}
↪  groups=${N_GROUPS}"

# ── Helpers ──────────────────────────────────────────────────────────
bb_port()  { echo $(( BB_PORT_BASE + ($1 - 1) * BB_PORT_STEP )); }
l2_sock()  { echo "${L2_SOCK_BASE}_${1}"; }
sap_sock() { echo "${SAP_SOCK_BASE}_${1}"; }
ms_imsi()  { printf '%s%s%04d%06d' "$_MCC" "$_MNC" "$OPERATOR_ID" "$1"; }
ms_imei()  { printf '3589250059%02d%02d0' "$OPERATOR_ID" "$1"; }
ms_ki()    { printf '00 11 22 33 44 55 66 77 88 99 aa bb cc dd %02x %02x' "$1" "$OPERATOR_ID"; }

wait_port() {
    local host=$1 port=$2 label=$3 timeout=${4:-60} elapsed=0
    echo -ne "  Attente ${label} (${host}:${port})"
    while ! bash -c "echo >/dev/tcp/${host}/${port}" 2>/dev/null; do
        sleep 1; elapsed=$((elapsed+1)); echo -n "."
        [ "$elapsed" -ge "$timeout" ] && echo -e " ${RED}TIMEOUT${NC}" && return 1
    done
    echo -e " ${GREEN}OK${NC} (${elapsed}s)"
}
```

```
wait_udp() {
    local port=$1 label=$2 timeout=${3:-30} elapsed=0
    echo -ne "  Attente ${label} (UDP ${port})"
    while ! ss -unlp | grep -q ":${port} "; do
        sleep 1; elapsed=$((elapsed+1)); echo -n "."
        [ "$elapsed" -ge "$timeout" ] && echo -e " ${RED}TIMEOUT${NC}" && return 1
    done
    echo -e " ${GREEN}OK${NC} (${elapsed}s)"
}

# ── TMUX ──────────────────────────────────────────────────────────────
SESSION="osmocom"; TMUX_SOCKET="/tmp/osmocom_tmux"; TMUX_OK=false

init_tmux() {
    echo -e "  ${CYAN}Init tmux...${NC}"
    rm -f "$TMUX_SOCKET"

    tmux -S "$TMUX_SOCKET" start-server 2>/dev/null || true
    sleep 0.5
    tmux -S "$TMUX_SOCKET" new-session -d -s "$SESSION" -n main 2>/dev/null || true
    sleep 0.5

    local i=0
    while [ ! -S "$TMUX_SOCKET" ] && [ $i -lt 30 ]; do sleep 0.3; i=$((i+1)); done

    if [ -S "$TMUX_SOCKET" ] && tmux -S "$TMUX_SOCKET" has-session -t "$SESSION" 2>/dev/null; then
        TMUX_OK=true
        echo -e "  ${GREEN}✓ tmux OK${NC}"
    else
        TMUX_OK=false
        echo -e "  ${RED}✗ tmux echec - mode background${NC}"
    fi
}

run_in_tmux() {
    local win=$1; shift; local cmd="$*"
    if [ "$TMUX_OK" = true ]; then
        tmux -S "$TMUX_SOCKET" new-window -t "$SESSION" -n "$win" 2>/dev/null || true
        sleep 0.2
        tmux -S "$TMUX_SOCKET" send-keys -t "${SESSION}:${win}" "$cmd" C-m 2>/dev/null || true
    else
        echo -e "  ${YELLOW}[bg]${NC} ${win}"
        bash -c "$cmd" >"/var/log/osmocom/${win}.log" 2>&1 &
    fi
}

# ── Config MS ─────────────────────────────────────────────────────────
generate_ms_configs() {
    local base_cfg="/root/.osmocom/bb/mobile.cfg"
    [ -f "$base_cfg" ] || { echo -e "${RED}mobile.cfg introuvable${NC}"; return 1; }

    local ref_imsi ref_imei ref_ki ref_l2 ref_sap
    ref_imsi=$(grep -oP '^\s+imsi\s+\K[0-9]+' "$base_cfg" 2>/dev/null | head -1 || true)
    ref_imei=$(grep -oP '^\s+imei \K[0-9]+' "$base_cfg" 2>/dev/null | head -1 || true)
    ref_ki=$(grep -oP '^\s+ki comp128 \K[0-9a-f ]+' "$base_cfg" 2>/dev/null | head -1 | sed 's/[[:space:]]*$//' ||
     ↪  true)
    ref_l2=$(grep -oP 'layer2-socket \K\S+' "$base_cfg" 2>/dev/null | head -1 || echo "/tmp/osmocom_l2")
    ref_sap=$(grep -oP 'sap-socket \K\S+' "$base_cfg" 2>/dev/null | head -1 || true)

    if [ -z "$ref_imsi" ]; then
        ref_imsi=$(grep -E 'imsi [0-9]+' "$base_cfg" | head -1 | awk '{print $NF}')
        ref_imei=$(grep -E 'imei [0-9]+' "$base_cfg" | head -1 | awk '{print $2}')
        ref_ki=$(grep -E 'ki comp128' "$base_cfg" | head -1 | sed 's/.*ki comp128 //' | sed 's/[[:space:]]*$//')
    fi
    [ -z "$ref_imsi" ] || [ -z "$ref_imei" ] || [ -z "$ref_ki" ] && return 1
```

```
    _MCC=$(grep -E '^\s+rplmn ' "$base_cfg" | awk '{print $2}' | head -1); _MCC=${_MCC:-001}
    _MNC=$(grep -E '^\s+rplmn ' "$base_cfg" | awk '{print $3}' | head -1); _MNC=${_MNC:-01}
    echo -e "  ${CYAN}[ref]${NC} IMSI=${ref_imsi} MCC/MNC=${_MCC}/${_MNC}"

    local global_line; global_line=$(grep -n '^ms [0-9]' "$base_cfg" | head -1 | cut -d: -f1)
    [ -z "$global_line" ] && return 1
    local global_header; global_header=$(head -n $((global_line - 1)) "$base_cfg")
    local ms_template; ms_template=$(awk '/^ms [0-9]/{if(f)exit;f=1;p=1} p{print} /^(exit|end)$/&&p&&f{p=0;exit}'
     ↪  "$base_cfg")
    [ -z "$ms_template" ] && return 1

    _write_ms_block() {
        local outfile=$1 lnum=$2 gidx=$3
        local ni=$(ms_imsi "$gidx") ne=$(ms_imei "$gidx") nk=$(ms_ki "$gidx") nl=$(l2_sock "$gidx") ns=$(sap_sock
         ↪  "$gidx")
        local block; block=$(echo "$ms_template" | sed \
            -e "s|^ms [0-9]\+|ms ${lnum}|" \
            -e "s|layer2-socket ${ref_l2}|layer2-socket ${nl}|" \
            -e "s|imsi ${ref_imsi}|imsi ${ni}|" \
            -e "s|imei ${ref_imei}[[:space:]]*[0-9]*|imei ${ne}|" \
            -e "s|ki comp128 ${ref_ki}|ki comp128 ${nk}|")
        [ -n "$ref_sap" ] && block=$(echo "$block" | sed "s|sap-socket ${ref_sap}|sap-socket ${ns}|") \
            || block=$(echo "$block" | sed "/layer2-socket/a\\ sap-socket ${ns}")
        echo "$block" >> "$outfile"; echo "" >> "$outfile"
    }

    # Si le fichier est bind-monte (docker -v file:file), rm renvoie EBUSY.
    # Dans ce cas on ne peut pas le supprimer mais on peut le truncate puis
    # re-ecrire dedans, ce qui est equivalent pour notre besoin.
    for f in /root/.osmocom/bb/mobile_ms*.cfg /root/.osmocom/bb/mobile_group*.cfg; do
        [ -e "$f" ] || continue
        if ! rm -f "$f" 2>/dev/null; then
            : > "$f" 2>/dev/null || \
                echo -e "  ${YELLOW}[warn] $f ni rm ni truncate (bind-mount RO ?)${NC}"
        fi
    done

    if [ "$MOBILE_MODE" = "split" ]; then
        for ms in $(seq 1 "$N_MS"); do
            echo "$global_header" > "/root/.osmocom/bb/mobile_ms${ms}.cfg"
            _write_ms_block "/root/.osmocom/bb/mobile_ms${ms}.cfg" 1 "$ms"
        done
    else
        local grp=1 gs=1
        while [ "$gs" -le "$N_MS" ]; do
            local ge=$((gs + MAX_MS_PER_MOBILE - 1)); [ "$ge" -gt "$N_MS" ] && ge=$N_MS
            local cfg="/root/.osmocom/bb/mobile_group${grp}.cfg"
            echo "$global_header" | sed "s|127\.0\.0\.1|127.0.0.${grp}|g" > "$cfg"
            local li=1; for ms in $(seq "$gs" "$ge"); do _write_ms_block "$cfg" "$li" "$ms"; li=$((li+1)); done
            echo -e "  ${CYAN}[Grp${grp}]${NC} IP=127.0.0.${grp} MS ${gs}-${ge}"
            grp=$((grp+1)); gs=$((ge+1))
        done
    fi
    echo -e "  ${GREEN}✓ Configs MS OK${NC}"
}

# ── TRX injection (fake_trx mode) ──────────────────────────────────────
inject_extra_trx() {
    [ "$1" -le 1 ] && return 0
    local bts="/etc/osmocom/osmo-bts-trx.cfg" bsc="/etc/osmocom/osmo-bsc.cfg"
    [ -f "$bts" ] || return 0
    local arfcn0; arfcn0=$(grep -E '^\s+arfcn [0-9]+' "$bsc" | head -1 | awk '{print $2}')
    local tmp; tmp=$(mktemp)
    awk -v n="$1" '/^ osmotrx ip local/&&!d{for(i=1;i<n;i++)printf" instance %d\n  osmotrx rx-gain 40\n  osmotrx
     ↪  tx-attenuation 50\n",i;d=1}{print}' "$bts" > "$tmp"; cp "$tmp" "$bts"; rm -f "$tmp"
    for t in $(seq 1 $(($1-1))); do
```

```
        local a=$((arfcn0 + t*2))
        printf ' trx %d\n  power-ramp max-initial 23000 mdBm\n  power-ramp step-size 2000 mdB\n  power-ramp
         ↪  step-interval 1\n  ms-power-control osmo\n  phy 0 instance %d\n' "$t" "$t" >> "$bts"
    done
    echo -e "  ${GREEN}✓ ${1} TRX injectes (osmo-bts-trx)${NC}"
}

# ── TRX injection (virtphy mode) ──────────────────────────────────────
inject_extra_trx_virtual() {
    [ "$1" -le 1 ] && return 0
    local bts="/etc/osmocom/osmo-bts-virtual.cfg"
    [ -f "$bts" ] || return 0

    # Ajouter des instances phy supplementaires apres "instance 0"
    local tmp; tmp=$(mktemp)
    local insert_done=0
    while IFS= read -r line; do
        echo "$line" >> "$tmp"
        if [ $insert_done -eq 0 ] && echo "$line" | grep -q '^ instance 0'; then
            for t in $(seq 1 $(($1-1))); do
                echo " instance ${t}" >> "$tmp"
            done
            insert_done=1
        fi
    done < "$bts"
    cp "$tmp" "$bts"; rm -f "$tmp"

    # Ajouter les TRX supplementaires en fin de fichier
    for t in $(seq 1 $(($1-1))); do
        printf ' trx %d\n  power-ramp max-initial 23000 mdBm\n  power-ramp step-size 2000 mdB\n  power-ramp
         ↪  step-interval 1\n  ms-power-control osmo\n  phy 0 instance %d\n' "$t" "$t" >> "$bts"
    done
    echo -e "  ${GREEN}✓ ${1} TRX injectes (osmo-bts-virtual)${NC}"
}

# ── Handover ──────────────────────────────────────────────────────────
inject_handover() {
    local cfg="/etc/osmocom/osmo-bsc.cfg"; [ -f "$cfg" ] || return 0
    local tmp; tmp=$(mktemp)
    sed -e 's/^ handover 0/ handover 1/' -e 's/^ handover algorithm 1/ handover algorithm 2/' "$cfg" > "$tmp"
    awk '/^ handover algorithm 2/&&!d{print;print" handover2 window rxlev averaging 10\n handover2 window rxqual
     ↪  averaging 1\n handover2 window rxlev neighbor averaging 10\n handover2 power budget interval 6\n handover2
     ↪  power budget hysteresis 3\n handover2 maximum distance 9999\n handover2 assignment 1\n handover2
     ↪  tdma-measurement full";d=1;next}/^ handover1 /{next}{print}' "$tmp" > "${tmp}.2"
    cp "${tmp}.2" "$cfg"; rm -f "$tmp" "${tmp}.2"
    echo -e "  ${GREEN}✓ Handover algo 2${NC}"
}

# ── Sinks PulseAudio (gsm_audio + gsm_mic) ────────────────────────────
# Docker : /scripts/  |  natif : /etc/osmocom/  (meme script, chemin different)
# Idempotent et NON FATAL : sans PulseAudio l'audio gapk est indispo, mais on ne
# doit jamais tuer run.sh (set -e) pour ca.
# $1 = timeout d'attente de PulseAudio (defaut 30 s). Le demon n'est pas demarre
# par ce script mais par lib/audio.sh (pipeline qemu) : sur l'appel PRECOCE [2d]
# il peut ne pas etre encore la, d'ou un timeout court pour ne pas staller.
setup_pulse_sinks() {
    local PULSE_SETUP=/scripts/pulse-gsm-setup.sh
    [ -f "$PULSE_SETUP" ] || PULSE_SETUP=/etc/osmocom/pulse-gsm-setup.sh
    [ -f "$PULSE_SETUP" ] || { echo -e "  ${YELLOW}pulse-gsm-setup.sh introuvable - audio indispo${NC}"; return 0;
     ↪  }
    PULSE_TIMEOUT="${1:-30}" "$PULSE_SETUP" \
        || echo -e "  ${YELLOW}[warn] pulse-gsm-setup.sh echoue (audio indispo) - on continue${NC}"
}

# ── Bridge audio UNIVERSEL ────────────────────────────────────────────
# gapk (RTP reseau → sink gsm_audio) + sortie son. Appele en mode faketrx/virtphy
```

```
# ET en mode no-process (qemu) - c'est CE bridge qui manquait en qemu (run.sh
# sortait avant [6c] → gsm_audio muet → "pas d'audio en qemu").
#
# Sortie : si le pulse de l'HOTE est joignable en TCP (relai ouvert par start-docker.sh,
# = WSLg→Windows en WSL, = pulse de session en Linux natif), on fait un pont
# direct parec|paplay (1 seule horloge, pas de derive → pas de son "pete").
# Sinon, fallback loopback local vers le sink par defaut du conteneur (carte).
audio_bridge() {
    pactl info >/dev/null 2>&1 || { echo -e "  ${YELLOW}PulseAudio indisponible - bridge audio non lance${NC}";
     ↪  return 0; }

    # gapk : RTP reseau → sink gsm_audio
    local GAPK=/scripts/gapk-start.sh; [ -f "$GAPK" ] || GAPK=/etc/osmocom/gapk-start.sh
    if [ -f "$GAPK" ]; then
        run_in_tmux "gapk" "GAPK_ALSA_DEV=gsm_out bash '$GAPK' auto gsm gsm_out"
        echo -e "  ${GREEN}✓ gapk auto (RTP reseau → gsm_audio)${NC}"
    else
        echo -e "  ${YELLOW}gapk-start.sh absent - pas de bridge RTP${NC}"
    fi

    # Sortie vers le pulse de l'hote si le relai TCP est ouvert (HOST_AUDIO_RELAY
    # est positionne par start-docker.sh quand le relai a ete active : tcp:<gw>:4713).
    local relay="${HOST_AUDIO_RELAY:-}"
    if [ -n "$relay" ] && pactl --server="$relay" info >/dev/null 2>&1; then
        # ── UN SEUL PONT POUR TOUT LE BANC ──────────────────────────────
        # [2026-08-31] Le pkill ci-dessous ne suffit pas des qu il y a
        # PLUSIEURS conteneurs : il s execute DANS le conteneur, qui a son
        # propre espace de PID, et ne voit donc jamais le pont du voisin.
        # Chaque operateur demarrait le sien, et comme ils partagent tous le
        # meme PulseAudio d hote, ils relisaient le MEME sink gsm_audio pour
        # le rejouer sur la MEME sortie : le son etait entendu deux fois a
        # deux operateurs, trois fois a trois. Constate le 31/08.
        #
        # On interroge donc le SERVEUR partage, pas la table des processus
        # locale : un client deja connecte sous ce nom signifie qu un pont
        # tourne quelque part, peu importe dans quel conteneur.
        local _bridge_name="osmo-gsm-bridge"
        if pactl --server="$relay" list clients 2>/dev/null \
             | grep -q "application.name = \"${_bridge_name}\""; then
            echo -e "  ${GREEN}✓ pont audio deja actif sur l hote (${relay}) - on n en ajoute pas un second${NC}"
            return 0
        fi
        # ── NI DOUBLON AVEC LE LOOPBACK DE L HOTE ───────────────────────
        # L hote charge deja, de son cote, un module-loopback
        # gsm_audio.monitor -> carte son (lib/audio.sh, ensure_local_loopback).
        # C est le MEME travail que ce pont : meme source, meme sortie. Les
        # deux ensemble, on entend tout DEUX FOIS - et le symptome survit a
        # l arret d un conteneur, ce qui fait chercher le doublon du mauvais
        # cote. Constate le 31/08 : sink-input #0 "Loopback from Monitor of
        # GSM_Audio" et #399 "paplay" sur la meme sortie.
        # On garde le loopback : il est cote hote et bien plus court
        # (20 ms contre 250 ms pour le pont TCP).
        if pactl --server="$relay" list short modules 2>/dev/null \
             | grep -q 'source=gsm_audio.monitor'; then
            echo -e "  ${GREEN}✓ loopback gsm_audio deja en place sur l hote - pont TCP inutile${NC}"
            return 0
        fi
        pkill -f "paplay --server=${relay}" 2>/dev/null || true
        [ -f /run/host-audio.pid ] && kill -- "-$(cat /run/host-audio.pid)" 2>/dev/null || true
        setsid sh -c '
          while true; do
            parec -d gsm_audio.monitor --latency-msec=30 --format=s16le --rate=8000 --channels=1 \
              | paplay --server='"${relay}"' --client-name='"${_bridge_name}"' --latency-msec=250 --raw
↪  --format=s16le --rate=8000 --channels=1
            sleep 1
          done' >/var/log/osmocom/host-audio.log 2>&1 &
```

```
        echo $! > /run/host-audio.pid
        echo -e "  ${GREEN}✓ pont audio parec|paplay → hote (${relay})${NC}"
    else
        # Fallback natif sans relai : loopback vers le sink par defaut (carte).
        if ! pactl list short modules 2>/dev/null | grep -q 'source=gsm_audio.monitor'; then
            pactl load-module module-loopback source=gsm_audio.monitor latency_msec=20 >/dev/null 2>&1 \
                && echo -e "  ${GREEN}✓ loopback gsm_audio → sink par defaut (carte)${NC}" \
                || echo -e "  ${YELLOW}loopback non charge (pas de sortie son ?)${NC}"
        else
            echo -e "  ${GREEN}✓ loopback gsm_audio deja actif${NC}"
        fi
    fi
}

# ═══════════════════════════════════════════════════════════════════
# MAIN
# ═══════════════════════════════════════════════════════════════════

# RUN_NO_PROCESS=1 : on prepare tmux + configs + core Osmocom (osmo-start :
# STP/HLR/MGW/MSC/BSC... → le HLR est up et peut etre alimente par start-docker.sh),
# mais on NE lance PAS les process radio/mobile/asterisk/smsc dans le tmux.
[ "$RUN_NO_PROCESS" = "1" ] && echo -e "${YELLOW}=== RUN_NO_PROCESS=1 : core seul, PHY/mobile/asterisk/smsc NON
 ↪  lances ===${NC}"

echo -e "${GREEN}=== [1/10] tmux ===${NC}"
init_tmux
echo ""

echo -e "${GREEN}=== [2/10] Config MS ===${NC}"
generate_ms_configs
echo ""

echo -e "${GREEN}=== [2b] TRX ===${NC}"
if [ "$PHY_MODE" = "virtphy" ]; then
    inject_extra_trx_virtual "$N_TRX"
else
    inject_extra_trx "$N_TRX"
fi

echo -e "${GREEN}=== [2c] Handover ===${NC}"
inject_handover
echo ""

# ── [2d] Audio PulseAudio - AVANT le coeur, volontairement ──────────
# [2026-08-12] Ce bloc n'existait qu'en [6b], donc APRES osmo-start.sh. Or
# osmo-start.sh fait `exit 1` si le HLR ne monte pas, et avec `set -euo pipefail`
# (ligne 7) run.sh meurt la : pulse-gsm-setup.sh - le seul endroit qui cree le
# null-sink gsm_mic - n'est alors JAMAIS execute. La capture gsm_in
# (= gsm_mic.monitor) ne s'ouvre plus, gapk_io abandonne les DEUX sens, et tout
# appel est muet, echo-test 600 compris. C'est exactement le "ca marche sur un
# PC et pas sur l'autre" : ca depend de si le HLR est monte du premier coup.
# L'audio n'a AUCUNE dependance sur le coeur - on le monte donc en premier.
# L'appel en [6b] reste en place (idempotent) pour les relances partielles.
echo -e "${GREEN}=== [2d] Audio PulseAudio (sinks) ===${NC}"
setup_pulse_sinks 3
echo ""

echo -e "${GREEN}=== [3/10] Core Osmocom ===${NC}"
/etc/osmocom/osmo-start.sh

# Mode no-process : le core tourne (HLR alimentable par start-docker.sh), on
# s'arrete ici sans lancer PHY/mobile/asterisk/smsc dans le tmux. On
# configure tout de meme PulseAudio (sink gsm_audio) car le pipeline lance
# ensuite (ex. ${OQC_ROOT}/run.sh) en a besoin pour l'audio gapk.
if [ "$RUN_NO_PROCESS" = "1" ]; then
    echo ""
```

```
    echo -e "${GREEN}=== Audio PulseAudio (no-process) ===${NC}"
    setup_pulse_sinks
    # Bridge audio AUSSI en no-process : sans ca, gapk n'est jamais lance et le
    # pipeline qemu n'a aucun audio (bug "pas de son en qemu").
    echo -e "${GREEN}=== Bridge audio (no-process / qemu) ===${NC}"
    audio_bridge
    echo ""
    echo -e "${GREEN}Core Osmocom pret (no-process). PHY/mobile/asterisk/smsc NON lances.${NC}"
    echo -e "  tmux : ${CYAN}tmux -S ${TMUX_SOCKET} attach -t ${SESSION}${NC}"
    exit 0
fi

# ═══════════════════════════════════════════════════════════════════
# PHY : fake_trx (TRXD) OU virtphy (multicast)
# ═══════════════════════════════════════════════════════════════════

if [ "$PHY_MODE" = "virtphy" ]; then
    # ─────────────────────────────────────────────────────────────────
    # MODE VIRTPHY : osmo-bts-virtual → virtphy → mobile
    #
    #   mobile ↔ virtphy ↔ (multicast UDP) ↔ osmo-bts-virtual ↔ BSC
    #
    # Pas de fake_trx, pas de trxcon.
    # ─────────────────────────────────────────────────────────────────

    echo -e "${GREEN}=== [4/10] BTS Virtual ===${NC}"
    # Arreter osmo-bts-trx s'il a ete demarre par systemd
    systemctl stop osmo-bts-trx 2>/dev/null || true
    systemctl disable osmo-bts-trx 2>/dev/null || true

    run_in_tmux "bts" "osmo-bts-virtual -c /etc/osmocom/osmo-bts-virtual.cfg"
    wait_port 127.0.0.1 4241 "BTS-virtual" 30 || true

    echo -e "${GREEN}=== [5/10] virtphy (${N_MS} instances) ===${NC}"
    for ms in $(seq 1 "$N_MS"); do
        local_l2=$(l2_sock "$ms")
        cmd="virtphy -s ${local_l2}"
        pw=$(( (ms-1)/8 )); pl=$(( (ms-1)%8 ))
        win="virtphy${pw}"
        if [ "$pl" -eq 0 ]; then
            run_in_tmux "$win" "$cmd"
        elif [ "$TMUX_OK" = true ]; then
            tmux -S "$TMUX_SOCKET" split-window -v -t "${SESSION}:${win}" 2>/dev/null || true
            tmux -S "$TMUX_SOCKET" select-layout -t "${SESSION}:${win}" even-vertical 2>/dev/null || true
            tmux -S "$TMUX_SOCKET" send-keys -t "${SESSION}:${win}.${pl}" "$cmd" C-m 2>/dev/null || true
        else
            bash -c "$cmd" >"/var/log/osmocom/virtphy_ms${ms}.log" 2>&1 &
        fi
        [ $((ms % 8)) -eq 1 ] || [ "$ms" -eq "$N_MS" ] && echo -e "  ${CYAN}[ue${ms}]${NC} l2=${local_l2}"
    done
    # Laisser virtphy creer les sockets avant de lancer mobile
    sleep 2

    # Pas de step 6 (trxcon) en mode virtphy

else
    # ─────────────────────────────────────────────────────────────────
    # MODE FAKETRX : fake_trx → trxcon → mobile
    #
    #   mobile ↔ trxcon ↔ (TRXD UDP) ↔ fake_trx ↔ osmo-bts-trx ↔ BSC
    # ─────────────────────────────────────────────────────────────────

    echo -e "${GREEN}=== [3b/10] osmo-bts-trx ===${NC}"
    # BTS lance AVANT fake_trx : il poll le transceiver (POWERON) et se connecte
    # des que fake_trx repond. On stoppe une eventuelle instance systemd pour
    # eviter un double-BTS.
```

```
    systemctl stop osmo-bts-trx 2>/dev/null || true
    run_in_tmux "bts" "osmo-bts-trx -c /etc/osmocom/osmo-bts-trx.cfg"

    echo -e "${GREEN}=== [4/10] FakeTRX ===${NC}"
    # [2026-09-08] -s 20 : ordonnancement temps reel (SCHED_RR) pour le
    # transceiver et son horloge TDMA. En ordonnancement normal, sur un hote
    # charge (srsRAN, QEMU, le bureau), le tic de 4,615 ms arrivait en retard
    # de 0,3 a 2 ms des dizaines de fois par minute ; la BTS compensait
    # (« N FN slower/faster than TRX ») et la parole perdait des trames.
    FAKETRX_CMD="python3 ${FAKETRX_PY} -s ${FAKETRX_RR_PRIO:-20} -b 127.0.0.1 -R 127.0.0.1 -r 127.0.0.1 -P
     ↪  ${BTS_PORT_BASE} -p ${BB_PORT_BASE}"
    for t in $(seq 1 $((N_TRX-1))); do
        FAKETRX_CMD+=" --trx bts${t}@127.0.0.1:${BTS_PORT_BASE}/${t}"
    done
    for ms in $(seq 2 "$N_MS"); do
        local_g=$(( (ms-1)/8 + 1 ))
        FAKETRX_CMD+=" --trx ue${ms}@127.0.0.${local_g}:$(bb_port $ms)"
    done
    run_in_tmux "faketrx" "$FAKETRX_CMD"
    wait_udp "$BTS_PORT_BASE" "fake_trx" 30 || true

    echo -e "${GREEN}=== [6/10] trxcon ===${NC}"
    for ms in $(seq 1 "$N_MS"); do
        local_l2=$(l2_sock "$ms"); local_port=$(bb_port "$ms")
        pw=$(( (ms-1)/8 )); pl=$(( (ms-1)%8 )); ug=$((pw+1)); gip="127.0.0.${ug}"
        cmd="trxcon ue${ms} -C 1 -i 127.0.0.1 -b ${gip} -p ${local_port} -s ${local_l2} -F 100"
        win="trxcon${pw}"
        if [ "$pl" -eq 0 ]; then
            run_in_tmux "$win" "$cmd"
        elif [ "$TMUX_OK" = true ]; then
            tmux -S "$TMUX_SOCKET" split-window -v -t "${SESSION}:${win}" 2>/dev/null || true
            tmux -S "$TMUX_SOCKET" select-layout -t "${SESSION}:${win}" even-vertical 2>/dev/null || true
            tmux -S "$TMUX_SOCKET" send-keys -t "${SESSION}:${win}.${pl}" "$cmd" C-m 2>/dev/null || true
        else
            bash -c "$cmd" >"/var/log/osmocom/trxcon_ue${ms}.log" 2>&1 &
        fi
        [ $((ms % 8)) -eq 1 ] || [ "$ms" -eq "$N_MS" ] && echo -e "  ${CYAN}[ue${ms}]${NC} ${gip}:${local_port}"
    done
fi

echo -e "${GREEN}=== [6b/10] Audio PulseAudio ===${NC}"
setup_pulse_sinks   # deja fait en [2d] - idempotent, filet pour relance partielle
echo ""

# ── [6c] Bridge audio : gapk (RTP → gsm_audio) + sortie vers l'hote ──────────
echo -e "${GREEN}=== [6c/10] Bridge audio (gapk + sortie hote) ===${NC}"
audio_bridge
echo ""

echo -e "${GREEN}=== [7/10] Mobile ===${NC}"
if [ "$MOBILE_MODE" = "split" ]; then
    for ms in $(seq 1 "$N_MS"); do
        run_in_tmux "ue${ms}" "mobile -c /root/.osmocom/bb/mobile_ms${ms}.cfg"
    done
else
    for g in $(seq 1 "$N_GROUPS"); do
        if [ "$g" -gt 1 ]; then
            prev_ip="127.0.0.$((g-1))"
            echo -ne "  ${CYAN}Attente Grp$((g-1)) (${prev_ip}:4247)${NC}"
            r=0
            while ! bash -c "echo >/dev/tcp/${prev_ip}/4247" 2>/dev/null; do
                sleep 2; echo -n "."; r=$((r+1)); [ $r -ge 30 ] && break
            done
            echo -e " ${GREEN}OK${NC}"; sleep 3
        fi
        run_in_tmux "ue_g${g}" "mobile -c /root/.osmocom/bb/mobile_group${g}.cfg"
```

```
        echo -e "  ${GREEN}Groupe ${g} demarre${NC}"
    done
fi

echo -e "${GREEN}=== [9/10] Asterisk ===${NC}"
run_in_tmux "asterisk" "pkill asterisk 2>/dev/null; sleep 2; pkill -9 asterisk 2>/dev/null; sleep 1; rm -f
↪  /var/lib/asterisk/astdb.sqlite3; asterisk -cvvv"

echo -e "${GREEN}=== [10/10] SMSC ===${NC}"
if [ -f /etc/osmocom/smsc-start.sh ]; then
    run_in_tmux "smsc" "/etc/osmocom/smsc-start.sh"
else
    run_in_tmux "smsc" "echo 'SMSC non disponible'"
fi

# ── Nettoyage : supprimer la fenetre main inutile ──────────────────────
if [ "$TMUX_OK" = true ]; then
    sleep 1
    tmux -S "$TMUX_SOCKET" kill-window -t "${SESSION}:main" 2>/dev/null || true
fi

# ── Resume ─────────────────────────────────────────────────────────────
echo ""
echo -e "${GREEN}╔═══════════════════════════════════════════════════════════╗${NC}"
echo -e "${GREEN}║  Op${OPERATOR_ID} - ${N_MS} MS - ${MOBILE_MODE} - ${PHY_MODE} - ${N_TRX} TRX     ║${NC}"
echo -e "${GREEN}╚═══════════════════════════════════════════════════════════╝${NC}"
echo ""
if [ "$PHY_MODE" = "virtphy" ]; then
    echo -e "  ${CYAN}PHY       :${NC} virtphy (osmo-bts-virtual, multicast)"
    echo -e "  ${CYAN}tmux      :${NC} bts  virtphy*  ue*  asterisk  smsc"
else
    echo -e "  ${CYAN}PHY       :${NC} faketrx (osmo-bts-trx, TRXD)"
    echo -e "  ${CYAN}BB ports  :${NC} $(bb_port 1) .. $(bb_port $N_MS)"
    echo -e "  ${CYAN}tmux      :${NC} faketrx  trxcon*  ue*  asterisk  smsc"
fi
if [ "$TMUX_OK" = true ]; then
    echo -e "  ${CYAN}Attach    :${NC} tmux -S ${TMUX_SOCKET} attach -t ${SESSION}"
    tmux -S "$TMUX_SOCKET" select-window -t "${SESSION}:ue_g1" 2>/dev/null \
        || tmux -S "$TMUX_SOCKET" select-window -t "${SESSION}:ue1" 2>/dev/null || true
fi
echo ""
echo -e "${GREEN}run.sh termine${NC}"
```

3.350 /opt/GSM/osmo-operator/scripts/send-mt-sms.sh

1005 octets, 29 lignes → 29 lignes

```
#!/bin/bash
# send-mt-sms.sh - Envoi de SMS MT local via proto-smsc-sendmt
# Usage : ./send-mt-sms.sh <imsi> <message> [from_number]
set -euo pipefail

OPERATOR_ID="${OPERATOR_ID:-1}"
SC_ADDRESS="1999001${OPERATOR_ID}444"
SENDMT_SOCKET="/tmp/sendmt_socket"

if [ $# -lt 2 ]; then
    echo "Usage: $0 <imsi> <message> [from_number]"
    echo "  Ex: $0 001010000000001 'Bonjour!'"
    exit 1
fi

DEST_IMSI="$1"; MESSAGE="$2"; FROM="${3:-${SC_ADDRESS}}"
```

```
[ ! -S "$SENDMT_SOCKET" ] && { echo "ERREUR: $SENDMT_SOCKET absent"; exit 1; }

sms-encode-text "$MESSAGE" \
    | gen-sms-deliver-pdu "$FROM" \
    | proto-smsc-sendmt "$SC_ADDRESS" "$DEST_IMSI" "$SENDMT_SOCKET"

# Ce MT part par GSUP vers la radio : les modems logiciels du banc ne le
# verraient jamais (« sms-over-gsup » court-circuite le SMPP d osmo-msc). On
# les previent, sans quoi un essai a la main n arrive que sur le mobile.
"$(dirname "$0")/sms_notify.py" "$DEST_IMSI" "$FROM" "$MESSAGE" || true

echo "MT SMS envoye → IMSI=$DEST_IMSI"
```

3.351 /opt/GSM/osmo-operator/scripts/sms-interop-relay.py

39003 octets, 934 lignes → 934 lignes

```
#!/usr/bin/env python3
"""
sms-interop-relay.py — Relay SMS inter-opérateur via proto-smsc-proto

Fixes par rapport à la version précédente :
  - Retry HLR VTY lookup avec backoff (race condition au démarrage)
  - Retry TCP relay send (3 tentatives avec délai)
  - Timeout TCP configurable (connect_timeout dans sms-routing.conf)
  - Support N opérateurs distants (pas de limite hardcodée)
  - Meilleure gestion des erreurs TPDU
  - Log structuré avec contexte opérateur
"""

import os
import sys
import json
import time
import socket
import struct
import select
import signal
import logging
import argparse
import threading
import subprocess
from pathlib import Path
from typing import Optional, Dict, Tuple

# ═══════════════════════════════════════════════════════════════════════
# Configuration
# ═══════════════════════════════════════════════════════════════════════

# [2026-09-07] PREVENIR LES MODEMS LOGICIELS. Ce MT part par GSUP vers la
# radio : le mobile osmocom-bb le recoit, mais les modems logiciels du banc
# (Android via oFono, la VM postmarketOS via son port serie PCI) ne le voient
# jamais - « sms-over-gsup » court-circuite le chemin SMPP d osmo-msc, seul
# endroit ou ils auraient pu l apprendre. Voir sms_notify.py.
# sms_notify.py ne vit que dans le depot, et ce fichier-ci peut etre la copie
# deployee dans /etc/osmocom : on cherche donc aux deux endroits.
for _rep in (str(Path(__file__).resolve().parent), "/opt/GSM/osmo-operator/scripts"):
    if _rep not in sys.path:
        sys.path.insert(0, _rep)
try:
    from sms_notify import notifier as notifier_modems
except ImportError:                                  # sms_notify.py absent
    def notifier_modems(*a, **k):
        return 0
```

```
RELAY_TCP_PORT = 7890
MO_LOG_POLL_INTERVAL = 0.5
HLR_VTY_HOST = "127.0.0.1"
HLR_VTY_PORT = 4258
SENDMT_SOCKET = "/tmp/sendmt_socket"

# Retry config
HLR_LOOKUP_RETRIES = 3
HLR_LOOKUP_DELAY = 2.0
RELAY_SEND_RETRIES = 3
RELAY_SEND_DELAY = 3.0
RELAY_CONNECT_TIMEOUT = 10

# ─── PAYLOAD ISM (option start-native.sh --payload-ism) ──────────────────
# Quand c'est actif, le TCP inter-noeud NE PORTE PLUS le texte du SMS : il porte
# un JETON. Le texte part sur la liaison radio ISM (scripts/ism_payload.py),
# estampille du meme jeton, et le noeud d'arrivee le recupere par la radio avant
# d'injecter le MT. Tout le reste (adresses, MSISDN, IMSI, SC-address) reste IP.
# Active par l'env PAYLOAD_ISM=1 ou la section [payload] de sms-routing.conf.
PAYLOAD_ISM_RECV_TIMEOUT = float(os.environ.get("ISM_RECV_TIMEOUT", "30"))
_PAYLOAD_LINK = None
_PAYLOAD_LINK_LOCK = threading.Lock()
_PAYLOAD_CFG: Dict[str, str] = {}


def payload_ism_enabled() -> bool:
    if os.environ.get("PAYLOAD_ISM", "0") == "1":
        return True
    return _PAYLOAD_CFG.get("mode", "").lower() == "ism"


def get_payload_link():
    """La liaison payload ISM, construite une seule fois (ou None si indisponible)."""
    global _PAYLOAD_LINK
    if not payload_ism_enabled():
        return None
    with _PAYLOAD_LINK_LOCK:
        if _PAYLOAD_LINK is None:
            try:
                sys.path.insert(0, str(Path(__file__).resolve().parent))
                import ism_payload
                _PAYLOAD_LINK = ism_payload.PayloadLink(cfg=dict(_PAYLOAD_CFG))
                logging.info("Payload ISM actif (liaison %s)",
                             _PAYLOAD_CFG.get("link")
                             or os.environ.get("ISM_LINK") or "loopback")
            except Exception as e:
                logging.error("Payload ISM indisponible, repli sur IP : %s", e)
                _PAYLOAD_LINK = False  # marqueur « echec », on ne retente pas
    return _PAYLOAD_LINK or None


def payload_new_token() -> str:
    import ism_payload
    return ism_payload.new_token()

logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s [RELAY-OP%(operator_id)s] %(levelname)s %(message)s',
    datefmt='%Y-%m-%dT%H:%M:%S'
)

# ═══════════════════════════════════════════════════════════════════════
# TPDU Parser
# ═══════════════════════════════════════════════════════════════════════
```

```
def decode_bcd_number(hex_bytes: bytes, num_digits: int) -> str:
    number = ""
    for byte in hex_bytes:
        lo = byte & 0x0F
        hi = (byte >> 4) & 0x0F
        if lo <= 9:
            number += str(lo)
        if hi <= 9 and len(number) < num_digits:
            number += str(hi)
    return number[:num_digits]


def parse_sms_submit_tpdu(hex_str: str) -> Optional[Dict]:
    try:
        data = bytes.fromhex(hex_str.strip())
    except ValueError:
        return None
    if len(data) < 4:
        return None

    first_octet = data[0]
    mti = first_octet & 0x03
    if mti != 0x01:
        return None

    vpf = (first_octet >> 3) & 0x03
    mr = data[1]
    da_len = data[2]
    da_toa = data[3]
    da_bytes = (da_len + 1) // 2

    if len(data) < 4 + da_bytes:
        return None

    da_digits = decode_bcd_number(data[4:4 + da_bytes], da_len)
    ton = (da_toa >> 4) & 0x07
    prefix = "+" if ton == 1 else ""

    offset = 4 + da_bytes
    result = {
        "mti": "SMS-SUBMIT",
        "mr": mr,
        "da_number": f"{prefix}{da_digits}",
        "da_ton": ton,
        "da_npi": da_toa & 0x0F,
        "da_raw": da_digits,
    }

    if offset < len(data):
        result["tp_pid"] = data[offset]; offset += 1
    if offset < len(data):
        result["tp_dcs"] = data[offset]; offset += 1

    if vpf == 2 and offset < len(data): offset += 1
    elif vpf == 3 and offset + 7 <= len(data): offset += 7
    elif vpf == 1 and offset + 7 <= len(data): offset += 7

    if offset < len(data):
        udl = data[offset]; result["tp_udl"] = udl; offset += 1
        dcs = result.get("tp_dcs", 0)
        if dcs == 0x00 and offset < len(data):
            result["user_data_hex"] = data[offset:].hex()

    return result
```

```
# ═══════════════════════════════════════════════════════════════════
# MO SMS Log Parser
# ═══════════════════════════════════════════════════════════════════

class MOLogEntry:
    def __init__(self):
        self.timestamp = ""
        self.imsi = ""
        self.sm_rp_mr = ""
        self.sm_rp_da = ""
        self.sm_rp_oa = ""
        self.sm_rp_oa_number = ""
        self.sm_rp_ui_len = 0
        self.tpdu_hex = ""
        self.parsed_tpdu = None

    @property
    def destination_number(self) -> str:
        if self.parsed_tpdu:
            return self.parsed_tpdu.get("da_raw", "")
        return ""

    @property
    def sender_number(self) -> str:
        return self.sm_rp_oa_number

    def __repr__(self):
        return f"MOLogEntry(from={self.sender_number}, to={self.destination_number}, imsi={self.imsi})"


def parse_mo_log_block(lines: list) -> Optional[MOLogEntry]:
    entry = MOLogEntry()
    for line in lines:
        line = line.strip()
        if not line: continue
        if line.endswith("Rx MO SM"):
            entry.timestamp = line.replace(" Rx MO SM", "")
        elif line.startswith("IMSI:"):
            entry.imsi = line.split(":", 1)[1].strip()
        elif line.startswith("SM-RP-MR:"):
            entry.sm_rp_mr = line.split(":", 1)[1].strip()
        elif line.startswith("SM-RP-DA:"):
            entry.sm_rp_da = line.split(":", 1)[1].strip()
        elif line.startswith("SM-RP-OA:"):
            parts = line.split()
            entry.sm_rp_oa = line.split(":", 1)[1].strip()
            if parts:
                entry.sm_rp_oa_number = parts[-1]
        elif line.startswith("SM-RP-UI:"):
            try:
                entry.sm_rp_ui_len = int(line.split(":")[1].strip().split()[0])
            except (ValueError, IndexError):
                pass
        elif all(c in "0123456789abcdefABCDEF" for c in line) and len(line) >= 4:
            entry.tpdu_hex = line
            entry.parsed_tpdu = parse_sms_submit_tpdu(line)

    if entry.imsi and entry.tpdu_hex:
        return entry
    return None


# ═══════════════════════════════════════════════════════════════════
# Routing Table — support N opérateurs
# ═══════════════════════════════════════════════════════════════════
```

```
class RoutingTable:
    def __init__(self, config_path: str):
        self.operators: Dict[str, str] = {}
        self.routes: list = []
        self.local_operator_id = "1"
        self.connect_timeout = RELAY_CONNECT_TIMEOUT
        self.retry_count = RELAY_SEND_RETRIES
        self.retry_delay = RELAY_SEND_DELAY
        self._load(config_path)

    def _load(self, path: str):
        section = None
        try:
            with open(path, 'r') as f:
                for line in f:
                    line = line.strip()
                    if not line or line.startswith('#'): continue
                    if line.startswith('[') and line.endswith(']'):
                        section = line[1:-1].lower(); continue
                    if '=' not in line: continue
                    key, val = line.split('=', 1)
                    key = key.strip()
                    val = val.split('#')[0].strip()

                    if section == 'operators':
                        # « ip » ou « ip:port ». Le port explicite sert au WAN :
                        # sur un noeud distant, un seul port public par
                        # opérateur peut atteindre le relais du bon container.
                        self.operators[key] = val
                    elif section == 'routes':
                        # « op_id » ou « op_id strip=N ». strip = nombre de
                        # chiffres d'INDICATIF a retirer avant de transmettre :
                        # le HLR du noeud d'arrivee ne connait que le numero nu.
                        parts = val.split()
                        op_id = parts[0] if parts else val
                        strip = 0
                        for extra in parts[1:]:
                            if extra.startswith('strip='):
                                try:
                                    strip = int(extra.split('=', 1)[1])
                                except ValueError:
                                    strip = 0
                        self.routes.append((key, op_id, strip))
                    elif section == 'local':
                        if key == 'operator_id': self.local_operator_id = val
                    elif section == 'relay':
                        if key == 'connect_timeout':
                            self.connect_timeout = int(val)
                        elif key == 'retry_count':
                            self.retry_count = int(val)
                        elif key == 'retry_delay':
                            self.retry_delay = int(val)
                    elif section == 'payload':
                        # [payload] mode=ism, link=fifo|serial|exec|loopback,
                        # tx=/chemin, rx=/chemin, serial=/dev/ttyUSB0,
                        # tx_cmd=..., rx_cmd=..., recv_timeout=30
                        _PAYLOAD_CFG[key] = val

            self.routes.sort(key=lambda x: len(x[0]), reverse=True)
        except FileNotFoundError:
            logging.warning(f"Routing config not found: {path}, using defaults")

    def lookup(self, msisdn: str) -> Optional[str]:
        clean = msisdn.lstrip('+')
        for prefix, op_id, _strip in self.routes:
            if clean.startswith(prefix):
```

```
                return op_id
        return None

    def lookup_strip(self, msisdn: str) -> int:
        """Chiffres d'indicatif a retirer pour cette destination (0 = aucun)."""
        clean = msisdn.lstrip('+')
        for prefix, _op_id, strip in self.routes:
            if clean.startswith(prefix):
                return strip
        return 0

    def is_local(self, msisdn: str) -> bool:
        op = self.lookup(msisdn)
        return op == self.local_operator_id or op is None

    def get_operator_ip(self, op_id: str) -> Optional[str]:
        """IP seule (compatibilite : le port se lit avec get_operator_target)."""
        target = self.operators.get(op_id)
        return target.split(':', 1)[0] if target else None

    def get_operator_target(self, op_id: str) -> Optional[Tuple[str, int]]:
        """(ip, port) — port par defaut RELAY_TCP_PORT si non precise."""
        target = self.operators.get(op_id)
        if not target:
            return None
        if ':' in target:
            host, _, port = target.partition(':')
            try:
                return host, int(port)
            except ValueError:
                return host, RELAY_TCP_PORT
        return target, RELAY_TCP_PORT


# ═══════════════════════════════════════════════════════════════════
# HLR VTY — MSISDN → IMSI avec retry
# ═══════════════════════════════════════════════════════════════════

def hlr_msisdn_to_imsi(msisdn: str,
                       hlr_host: str = HLR_VTY_HOST,
                       hlr_port: int = HLR_VTY_PORT,
                       retries: int = HLR_LOOKUP_RETRIES,
                       delay: float = HLR_LOOKUP_DELAY) -> Optional[str]:
    """Lookup MSISDN→IMSI via HLR VTY with retry on failure."""
    clean = msisdn.lstrip('+')

    for attempt in range(retries):
        try:
            s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            s.settimeout(5)
            s.connect((hlr_host, hlr_port))

            def recv_until_prompt(timeout=3):
                data = b""
                deadline = time.time() + timeout
                while time.time() < deadline:
                    try:
                        chunk = s.recv(4096)
                        if not chunk: break
                        data += chunk
                        decoded = data.decode("ascii", errors="replace")
                        lines = decoded.strip().splitlines()
                        if lines:
                            last = lines[-1].strip()
                            if last.endswith(">") or last.endswith("#"):
                                break
```

```
                    except socket.timeout:
                        break
                return data.decode("ascii", errors="replace")

            recv_until_prompt()
            s.sendall(b"enable\r\n")
            recv_until_prompt()

            cmd = f"show subscriber msisdn {clean}\r\n"
            s.sendall(cmd.encode())
            response = recv_until_prompt()
            s.close()

            for line in response.splitlines():
                stripped = line.strip()
                if stripped.startswith("IMSI:"):
                    imsi = stripped.split(":", 1)[1].strip()
                    if imsi.isdigit():
                        logging.info(f"HLR match: MSISDN {clean} → IMSI {imsi}")
                        return imsi

            # IMSI not found (not a connection error) — don't retry
            logging.warning(f"MSISDN {clean} not found in HLR (attempt {attempt+1})")
            if attempt < retries - 1:
                time.sleep(delay)
            continue

        except (ConnectionRefusedError, socket.timeout, OSError) as e:
            logging.warning(f"HLR VTY connection failed (attempt {attempt+1}/{retries}): {e}")
            try: s.close()
            except: pass
            if attempt < retries - 1:
                time.sleep(delay)
            continue
        except Exception as e:
            logging.error(f"HLR VTY unexpected error: {e}")
            try: s.close()
            except: pass
            if attempt < retries - 1:
                time.sleep(delay)

    logging.error(f"HLR lookup failed for MSISDN {clean} after {retries} attempts")
    return None


# ═══════════════════════════════════════════════════════════════════
# MT SMS Injection (local)
# ═══════════════════════════════════════════════════════════════════

def inject_mt_sms(dest_imsi: str, message_text: str, from_number: str,
                  sc_address: str, sendmt_socket: str = SENDMT_SOCKET) -> bool:
    # Wait for socket if not present yet (race at startup)
    for _ in range(5):
        if os.path.exists(sendmt_socket):
            break
        logging.warning(f"sendmt socket {sendmt_socket} not found, waiting 2s...")
        time.sleep(2)

    if not os.path.exists(sendmt_socket):
        logging.error(f"sendmt socket {sendmt_socket} still missing after wait")
        return False

    try:
        # Escape single quotes in message
        safe_msg = message_text.replace("'", "'\\''")
        cmd = (
```

```
            f"sms-encode-text '{safe_msg}' "
            f"| gen-sms-deliver-pdu {from_number} "
            f"| proto-smsc-sendmt {sc_address} {dest_imsi} {sendmt_socket}"
        )
        result = subprocess.run(cmd, shell=True, capture_output=True, text=True, timeout=10)
        if result.returncode == 0:
            logging.info(f"MT SMS injected: IMSI={dest_imsi}")
            n = notifier_modems(dest_imsi, from_number, message_text)
            logging.info(f"Modems notified: {n}")
            return True
        else:
            logging.error(f"MT SMS injection failed (rc={result.returncode}): {result.stderr}")
            return False
    except Exception as e:
        logging.error(f"MT SMS injection error: {e}")
        return False


# ═══════════════════════════════════════════════════════════════════════
# TCP Relay Server
# ═══════════════════════════════════════════════════════════════════════

class RelayServer(threading.Thread):
    def __init__(self, port: int, operator_id: str, sc_address: str):
        super().__init__(daemon=True)
        self.port = port
        self.operator_id = operator_id
        self.sc_address = sc_address
        self.running = True

    def run(self):
        server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        server.bind(('0.0.0.0', self.port))
        server.listen(5)
        server.settimeout(1.0)
        logging.info(f"Relay server listening on port {self.port}")

        while self.running:
            try:
                conn, addr = server.accept()
                threading.Thread(target=self._handle_client, args=(conn, addr), daemon=True).start()
            except socket.timeout:
                continue
            except Exception as e:
                logging.error(f"Server error: {e}")
        server.close()

    def _handle_client(self, conn: socket.socket, addr: tuple):
        try:
            conn.settimeout(10)
            data = b""
            while True:
                chunk = conn.recv(4096)
                if not chunk: break
                data += chunk
                if b'\n' in data: break

            if not data:
                conn.close(); return

            msg = json.loads(data.decode('utf-8').strip())
            if msg.get("type") == "mt_sms":
                self._handle_mt_sms(msg, conn)
            else:
                response = {"status": "error", "reason": "unknown message type"}
```

```
                conn.sendall((json.dumps(response) + '\n').encode())
            except Exception as e:
                logging.error(f"Client handler error from {addr}: {e}")
            finally:
                conn.close()

    def _handle_mt_sms(self, msg: dict, conn: socket.socket):
        dest_msisdn = msg.get("dest_msisdn", "")
        from_number = msg.get("from_number", "")
        message_text = msg.get("message_text", "")

        # Payload ISM : le texte n'etait pas dans le TCP, il arrive par la radio.
        # On attend le jeton sur la liaison avant d'injecter le MT.
        if msg.get("payload_via") == "ism":
            token = msg.get("payload_token", "")
            link = get_payload_link()
            if link is None:
                response = {"status": "error",
                            "reason": "payload annonce ISM mais liaison radio absente ici"}
                conn.sendall((json.dumps(response) + '\n').encode())
                return
            payload = link.recv(token, timeout=PAYLOAD_ISM_RECV_TIMEOUT)
            if payload is None:
                logging.error("Payload ISM : jeton %s jamais recu par radio", token)
                response = {"status": "error",
                            "reason": f"payload ISM (jeton {token}) non recu par radio"}
                conn.sendall((json.dumps(response) + '\n').encode())
                return
            message_text = payload.decode("utf-8", "replace")
            logging.info("Payload ISM : texte recu par radio (jeton %s)", token)

        logging.info(f"Incoming interop MT: {from_number} → {dest_msisdn}")

        # Resolve with retries
        dest_imsi = hlr_msisdn_to_imsi(dest_msisdn)
        if not dest_imsi:
            response = {"status": "error", "reason": f"MSISDN {dest_msisdn} not found in HLR"}
            conn.sendall((json.dumps(response) + '\n').encode())
            return

        success = inject_mt_sms(
            dest_imsi=dest_imsi,
            message_text=message_text,
            from_number=from_number,
            sc_address=self.sc_address
        )

        response = {
            "status": "ok" if success else "error",
            "dest_imsi": dest_imsi,
            "reason": "" if success else "injection failed"
        }
        conn.sendall((json.dumps(response) + '\n').encode())


# ═══════════════════════════════════════════════════════════════
# TCP Relay Client — avec retry
# ═══════════════════════════════════════════════════════════════

def send_interop_mt(target_ip: str, target_port: int, dest_msisdn: str,
                    from_number: str, message_text: str, sc_address: str,
                    sender_imsi: str = "",
                    retries: int = RELAY_SEND_RETRIES,
                    delay: float = RELAY_SEND_DELAY,
                    connect_timeout: int = RELAY_CONNECT_TIMEOUT) -> bool:
    """Send interop MT SMS with retry on TCP failure."""
```

```
    msg = {
        "type": "mt_sms",
        "dest_msisdn": dest_msisdn,
        "from_number": from_number,
        "message_text": message_text,
        "sc_address": sc_address,
        "sender_imsi": sender_imsi,
    }

    # Payload ISM : le texte part par la radio, le TCP ne garde qu'un jeton.
    link = get_payload_link()
    if link is not None:
        try:
            token = payload_new_token()
            link.send(token, message_text.encode("utf-8"))
            msg["payload_via"] = "ism"
            msg["payload_token"] = token
            msg["message_text"] = ""   # le texte ne passe pas par IP
            logging.info("Payload ISM : texte emis par radio (jeton %s)", token)
        except Exception as e:
            # La radio a echoue : on ne laisse pas le SMS muet, on retombe sur IP.
            logging.error("Payload ISM echoue, repli IP pour ce SMS : %s", e)

    for attempt in range(retries):
        try:
            s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            s.settimeout(connect_timeout)
            s.connect((target_ip, target_port))
            s.sendall((json.dumps(msg) + '\n').encode())

            data = b""
            while True:
                chunk = s.recv(4096)
                if not chunk: break
                data += chunk
                if b'\n' in data: break
            s.close()

            if data:
                response = json.loads(data.decode('utf-8').strip())
                if response.get("status") == "ok":
                    logging.info(f"Interop MT delivered: {from_number} → {dest_msisdn}")
                    return True
                else:
                    logging.error(f"Interop MT rejected: {response.get('reason')}")
                    # Don't retry if the remote explicitly rejected
                    return False
            else:
                logging.warning(f"Empty response from relay (attempt {attempt+1})")

        except (ConnectionRefusedError, socket.timeout, OSError) as e:
            logging.warning(f"Relay send to {target_ip}:{target_port} failed "
                            f"(attempt {attempt+1}/{retries}): {e}")
            try: s.close()
            except: pass

        if attempt < retries - 1:
            logging.info(f"Retrying in {delay}s...")
            time.sleep(delay)

    logging.error(f"Interop send failed after {retries} attempts to {target_ip}")
    return False


# ═══════════════════════════════════════════════════════════════════
# GSM 7-bit Decoder
```

```
# ═══════════════════════════════════════════════════════════════

GSM7_BASIC = (
    "@£$¥èéùìòÇ\nØø\rÅå"
    "Δ_ΦΓΛΩΠΨΣΘΞ\x1bÆæßÉ"
    " !\"#¤%&'()*+,-./"
    "0123456789:;<=>?"
    "¡ABCDEFGHIJKLMNO"
    "PQRSTUVWXYZÄÖÑÜ§"
    "¿abcdefghijklmno"
    "pqrstuvwxyzäöñüà"
)

def decode_gsm7(data: bytes, num_chars: int) -> str:
    chars = []
    for i in range(num_chars):
        byte_offset = (i * 7) // 8
        bit_offset = (i * 7) % 8
        if byte_offset >= len(data): break
        val = (data[byte_offset] >> bit_offset) & 0x7F
        if bit_offset > 1 and byte_offset + 1 < len(data):
            val = ((data[byte_offset] >> bit_offset) |
                   (data[byte_offset + 1] << (8 - bit_offset))) & 0x7F
        if val < len(GSM7_BASIC):
            chars.append(GSM7_BASIC[val])
        else:
            chars.append('?')
    return ''.join(chars)


def extract_text_from_tpdu(tpdu_hex: str) -> str:
    parsed = parse_sms_submit_tpdu(tpdu_hex)
    if not parsed: return ""
    dcs = parsed.get("tp_dcs", 0)
    udl = parsed.get("tp_udl", 0)
    ud_hex = parsed.get("user_data_hex", "")
    if dcs == 0x00 and ud_hex:
        try:
            return decode_gsm7(bytes.fromhex(ud_hex), udl)
        except: pass
    return ""


def extract_text_via_sms_decode(tpdu_hex: str) -> str:
    try:
        result = subprocess.run(
            ['sms-pdu-decode', '-n'],
            input=tpdu_hex + '\n',
            capture_output=True, text=True, timeout=5
        )
        if result.returncode == 0:
            lines = result.stdout.strip().split('\n')
            text_lines = []
            found_length = False
            for line in lines:
                if found_length and line.strip():
                    text_lines.append(line)
                if line.strip().startswith('Length:'):
                    found_length = True
            if text_lines:
                return '\n'.join(text_lines).strip()
    except (FileNotFoundError, subprocess.TimeoutExpired):
        pass
    return extract_text_from_tpdu(tpdu_hex)
```

```
# ═══════════════════════════════════════════════════════════════
# MO Log Watcher
# ═══════════════════════════════════════════════════════════════

class MOLogWatcher(threading.Thread):
    def __init__(self, log_path: str, routing: RoutingTable,
                 operator_id: str, sc_address: str):
        super().__init__(daemon=True)
        self.log_path = log_path
        self.routing = routing
        self.operator_id = operator_id
        self.sc_address = sc_address
        self.running = True
        self._position = 0

    def run(self):
        logging.info(f"Watching MO log: {self.log_path}")
        while self.running and not Path(self.log_path).exists():
            time.sleep(2)
        if not self.running: return

        with open(self.log_path, 'r') as f:
            f.seek(0, 2)
            self._position = f.tell()

        logging.info(f"MO log watcher started (pos={self._position})")
        while self.running:
            try:
                self._check_new_entries()
            except Exception as e:
                logging.error(f"MO watcher error: {e}")
            time.sleep(MO_LOG_POLL_INTERVAL)

    def _check_new_entries(self):
        try:
            with open(self.log_path, 'r') as f:
                f.seek(self._position)
                new_data = f.read()
                self._position = f.tell()
        except FileNotFoundError:
            return
        if not new_data: return

        blocks = []
        current_block = []
        for line in new_data.split('\n'):
            if "Rx MO SM" in line and current_block:
                blocks.append(current_block)
                current_block = [line]
            else:
                current_block.append(line)
        if current_block:
            blocks.append(current_block)

        for block in blocks:
            entry = parse_mo_log_block(block)
            if entry:
                self._route_mo_sms(entry)

    def _route_mo_sms(self, entry: MOLogEntry):
        dest = entry.destination_number
        sender = entry.sender_number
        if not dest:
            logging.warning(f"MO SMS without parsable destination: {entry}")
            return
```

```
    logging.info(f"MO SMS: {sender} → {dest} (from IMSI {entry.imsi})")
    target_op = self.routing.lookup(dest)

    if target_op is None:
        logging.warning(f"No route for destination {dest}")
        return

    if target_op == self.operator_id:
        logging.info(f"Local delivery for {dest}")
        self._deliver_local(entry, dest)
        return

    target = self.routing.get_operator_target(target_op)
    if not target:
        logging.error(f"No IP for operator {target_op}")
        return
    target_ip, target_port = target

    # Indicatif WAN : on le retire ICI, cote emetteur. Le noeud d'arrivee
    # recoit le numero nu, celui que son HLR connait ; sans ce retrait il
    # cherche « 3310001 » et repond « MSISDN not found in HLR ».
    strip = self.routing.lookup_strip(dest)
    dest_out = dest.lstrip('+')[strip:] if strip else dest
    if strip:
        logging.info(f"WAN strip {strip}: {dest} → {dest_out}")

    logging.info(f"Interop route: {dest} → Op{target_op} ({target_ip}:{target_port})")

    message_text = extract_text_via_sms_decode(entry.tpdu_hex)
    if not message_text:
        message_text = extract_text_from_tpdu(entry.tpdu_hex)
    if not message_text:
        logging.warning(f"Could not decode text from TPDU: {entry.tpdu_hex[:40]}...")
        message_text = "[undecoded]"

    success = send_interop_mt(
        target_ip=target_ip,
        target_port=target_port,
        dest_msisdn=dest_out,
        from_number=sender,
        message_text=message_text,
        sc_address=self.sc_address,
        sender_imsi=entry.imsi,
        retries=self.routing.retry_count,
        delay=self.routing.retry_delay,
        connect_timeout=self.routing.connect_timeout,
    )

    if success:
        logging.info(f"✓ Interop SMS: {sender} → {dest} (Op{target_op})")
    else:
        logging.error(f"✗ Interop SMS failed: {sender} → {dest} (Op{target_op})")

def _deliver_local(self, entry: MOLogEntry, dest_msisdn: str):
    dest_imsi = hlr_msisdn_to_imsi(dest_msisdn)
    if not dest_imsi:
        logging.warning(f"Local MSISDN {dest_msisdn} not found in HLR")
        return

    message_text = extract_text_via_sms_decode(entry.tpdu_hex)
    if not message_text:
        message_text = extract_text_from_tpdu(entry.tpdu_hex)
    if not message_text:
        message_text = "[undecoded]"

    inject_mt_sms(
```

```
            dest_imsi=dest_imsi,
            message_text=message_text,
            from_number=entry.sender_number,
            sc_address=self.sc_address
        )


# ═══════════════════════════════════════════════════════════════════
# Main
# ═══════════════════════════════════════════════════════════════════

def main():
    parser = argparse.ArgumentParser(description='SMS Interop Relay')
    parser.add_argument('--config', '-c', default='/etc/osmocom/sms-routing.conf')
    parser.add_argument('--port', '-p', type=int, default=RELAY_TCP_PORT)
    parser.add_argument('--mo-log')
    parser.add_argument('--operator-id', default=os.environ.get('OPERATOR_ID', '1'))
    # --node-id : LA CLE DE ROUTAGE, et elle n'est pas l'operateur.
    #
    # local_operator_id etait ecrase par --operator-id juste apres la lecture du
    # fichier, donc la cle du relais valait toujours le numero d'OPERATEUR. Avec
    # des MSISDN qui commencent par le noeud (<noeud>00<op><ms>), sms-routing.conf
    # est desormais indexe par NOEUD : l'operateur 1 du noeud 1 et celui du noeud
    # 2 portaient sinon la meme cle "1", et chacun tenait les numeros de l'autre
    # pour locaux - un SMS interop n'etait jamais emis, il etait injecte dans le
    # HLR local qui ne connait pas l'abonne.
    #
    # L'operateur, lui, garde ce qui le concerne : la SC-address (le numero du
    # centre SMS, qu'un abonne voit) et le chemin du journal MO. Les changer
    # renommerait un fichier que d'autres outils suivent, et un numero que des
    # mobiles ont deja en memoire.
    parser.add_argument('--node-id',
                        default=os.environ.get('OSMO_WAN_NODE',
                                os.environ.get('WAN_NODE_ID', '')))
    args = parser.parse_args()

    op_id = args.operator_id
    node_id = args.node_id if args.node_id not in ('', '0') else op_id
    sc_address = f"1999001{op_id}444"
    mo_log = args.mo_log or f"/var/log/osmocom/mo-sms-op{op_id}.log"

    old_factory = logging.getLogRecordFactory()
    def record_factory(*a, **kw):
        record = old_factory(*a, **kw)
        record.operator_id = op_id
        return record
    logging.setLogRecordFactory(record_factory)

    logging.info(f"SMS Interop Relay starting")
    logging.info(f"  Operator   : {op_id}")
    logging.info(f"  Node       : {node_id}   (cle de routage dans sms-routing.conf)")
    logging.info(f"  SC-address : {sc_address}")
    logging.info(f"  MO log     : {mo_log}")
    logging.info(f"  TCP port   : {args.port}")

    routing = RoutingTable(args.config)
    routing.local_operator_id = node_id

    logging.info(f"  Operators  : {routing.operators}")
    logging.info(f"  Routes     : {len(routing.routes)} entries")
    logging.info(f"  Retry      : {routing.retry_count}x, delay={routing.retry_delay}s,
↪   timeout={routing.connect_timeout}s")

    # Payload ISM : si demande, on monte la liaison radio MAINTENANT pour que le
    # recepteur (RX) tourne avant le premier SMS, et on le dit clairement.
    if payload_ism_enabled():
```

```
        link = get_payload_link()
        if link is not None:
            logging.info("  Payload    : ISM (texte par radio, signalisation par IP)")
        else:
            logging.warning("  Payload    : ISM demande mais liaison indisponible — repli IP")
    else:
        logging.info("  Payload    : IP (texte dans le relais TCP)")

    server = RelayServer(port=args.port, operator_id=op_id, sc_address=sc_address)
    server.start()

    watcher = MOLogWatcher(log_path=mo_log, routing=routing, operator_id=op_id, sc_address=sc_address)
    watcher.start()

    def signal_handler(sig, frame):
        logging.info("Shutting down...")
        watcher.running = False
        server.running = False
        sys.exit(0)

    signal.signal(signal.SIGINT, signal_handler)
    signal.signal(signal.SIGTERM, signal_handler)

    logging.info("Relay running. Ctrl+C to stop.")
    while True:
        time.sleep(1)


if __name__ == '__main__':
    main()
```

3.352 /opt/GSM/osmo-operator/scripts/sms-routing-setup.sh

27017 octets, 581 lignes → 581 lignes

```
#!/bin/bash
# sms-routing-setup.sh - Gestion complete du routage SMS inter-operateur
#
# Fonctions exportables appelees depuis start-docker.sh :
#   sms_routing_generate  <op_id> <n_ops> <destdir> [op1_nms op2_nms ...]
#   sms_routing_validate  <conf_file>
#   sms_routing_summary   <n_ops>         [op1_nms op2_nms ...]
#
# Formules COMMUNES (identiques a run.sh, hlr-feed-subscribers.sh) :
#   MSISDN  = <noeud>00<op_id><ms_idx>       (100101, 100102, 100201...)
#   IMSI    = MCC(3) + MNC(2) + printf('%04d%06d', op_id, ms_idx)
#   KI      = 00 11 22 33 44 55 66 77 88 99 aa bb cc dd <ms_hex> <op_hex>
#
# Architecture de routage :
#   ┌──────────────────────────────────────────────────────────────┐
#   │  MS (op1, ms1)  MSISDN=10001  envoie a  MSISDN=20002 (op2, ms2)   │
#   │    → proto-smsc-daemon (op1) → MO log                              │
#   │    → sms-interop-relay.py (op1) : lookup prefix 20002 → op2        │
#   │    → TCP 172.20.0.12:7890                                          │
#   │    → sms-interop-relay.py (op2) : MSISDN→IMSI via HLR VTY          │
#   │    → proto-smsc-sendmt (op2) → HLR (op2) → MS (op2, ms2)           │
#   └──────────────────────────────────────────────────────────────┘
#
# Usage autonome (test/debug) :
#   bash sms-routing-setup.sh generate 1 3 /tmp/sms-cfg 2 2 3
#   bash sms-routing-setup.sh validate /tmp/sms-cfg/sms-routing-op1.conf
#   bash sms-routing-setup.sh summary  3 2 2 3

set -euo pipefail
```

```
RED='\033[0;31m'; GREEN='\033[0;32m'; YELLOW='\033[1;33m'
CYAN='\033[0;36m'; BOLD='\033[1m'; NC='\033[0m'

# ═══════════════════════════════════════════════════════════════════════
# Helpers internes
# ═══════════════════════════════════════════════════════════════════════

_sms_op_backbone_ip() { echo "172.20.0.$((10 + $1))"; }
# MSISDN = <noeud>00<operateur><rang sur 2 chiffres> : le premier chiffre est
# l'adresse du noeud, c'est lui qui route. Le noeud vient du plan radio ecrit
# par generate_configs.sh, de l'environnement, ou vaut 1.
_sms_node_id() {
    local n="${OSMO_WAN_NODE:-${WAN_NODE_ID:-}}"
    [ -n "$n" ] || n="$(sed -n 's/^PLAN_NODE=//p' /etc/osmocom/radio-plan.env 2>/dev/null | tail -1)"
    case "$n" in [1-9]) ;; *) n=1 ;; esac
    printf '%s' "$n"
}
_sms_op_msisdn()      { echo $(( $(_sms_node_id) * 100000 + $1 * 100 + $2 )); }   # op_id ms_idx
_sms_op_ms_imsi()     {                                    # mcc mnc op_id ms_idx
    local mcc=$1 mnc=$2 op=$3 ms=$4
    printf '%s%s%04d%06d' "$mcc" "$mnc" "$op" "$ms"
}
_sms_sc_address()     { printf '1999001%s444' "$1"; }    # op_id

_log_ok()   { echo -e "  ${GREEN}✓${NC}  $*"; }
_log_warn() { echo -e "  ${YELLOW}⚠${NC}  $*"; }
_log_err()  { echo -e "  ${RED}✗${NC}  $*" >&2; }

# ═══════════════════════════════════════════════════════════════════════
# sms_routing_generate
#
# Genere configs/sms-routing-op<N>.conf pour chaque operateur.
# Le fichier de chaque operateur contient TOUTES les routes (locales + distantes)
# et est monte dans /etc/osmocom/sms-routing.conf du container.
#
# Usage :
#   sms_routing_generate <op_id> <n_ops> <destdir> [ms_counts...]
#   ms_counts : nombre de MS par operateur (1 valeur par operateur)
#               ex: "2 3 1" → op1=2MS, op2=3MS, op3=1MS
#
# Sortie :
#   <destdir>/sms-routing-op<op_id>.conf
#
# ═══════════════════════════════════════════════════════════════════════
sms_routing_generate() {
    local op_id="$1"
    local n_ops="$2"
    local destdir="$3"
    shift 3
    local ms_counts=("$@")   # tableau : ms_counts[0]=nMS_op1, [1]=nMS_op2, ...

    mkdir -p "$destdir"
    local outfile="${destdir}/sms-routing-op${op_id}.conf"

    # ── LES NUMEROS D OPERATEUR NE COMMENCENT PAS FORCEMENT A 1 ──────────
    # [2026-08-31] Tout ce fichier enumerait 1..n_ops. Sur le banc
    # multi-operateur, les CONTENEURS portent les rangs 2 et 3 - le 1 est au
    # natif (OP_ID_BASE=2). La table sortait donc "Op1 172.20.0.11 / Op2
    # 172.20.0.12" pour ce qui est en realite op2 et op3 : de mauvais MSISDN
    # (100101 au lieu de 100201), de mauvaises IMSI, et un relais annonce a une
    # adresse qui n est celle de personne (172.20.0.11 est le natif, qui n est
    # pas un conteneur).
    #
    # ⚠ ms_counts reste une liste POSITIONNELLE 0-based : son premier element
    # decrit le premier operateur enumere, quel que soit son rang. On decale
```

```
    # donc la NUMEROTATION, pas l indexation - d ou le (i - _b).
    local _b="${OP_ID_BASE:-1}"
    local _last=$(( _b + n_ops - 1 ))
    local -a nms
    for i in $(seq "$_b" "$_last"); do
        nms[$i]=${ms_counts[$((i - _b))]:-1}
    done

    local mcc="${MCC:-001}"

    cat > "$outfile" << HEADER
# ═══════════════════════════════════════════════════════════════════
# sms-routing-op${op_id}.conf
# Genere par sms-routing-setup.sh - $(date '+%Y-%m-%d %H:%M:%S')
#
# Operateur local  : ${op_id}
# Nombre d'operat. : ${n_ops}
# MS par operateur : $(for i in $(seq "${OP_ID_BASE:-1}" "$(( ${OP_ID_BASE:-1} + n_ops - 1 ))"); do printf
 ↪  'Op%s=%s ' "$i" "${nms[$i]}"; done)
#
# Routage MSISDN :
#   MSISDN = <noeud> × 100000 + op_id × 100 + ms_idx
#   Regle longest-prefix match (prefixe le plus long l'emporte)
#
# ═══════════════════════════════════════════════════════════════════

[local]
operator_id = ${op_id}
sc_address  = $(_sms_sc_address "$op_id")
hlr_vty_ip  = 127.0.0.1
hlr_vty_port = 4258
sendmt_socket = /tmp/sendmt_socket
mo_log = /var/log/osmocom/mo-sms-op${op_id}.log

[operators]
# operator_id = container_ip  (reseau backbone 172.20.0.0/24)
HEADER

    for i in $(seq "${OP_ID_BASE:-1}" "$(( ${OP_ID_BASE:-1} + n_ops - 1 ))"); do
        printf '%s = %s\n' "$i" "$(_sms_op_backbone_ip "$i")" >> "$outfile"
    done

    cat >> "$outfile" << ROUTES_HEADER

[routes]
# Format : prefix = operator_id
# Longest-prefix match : le prefixe le plus long l'emporte.
# Strategie :
#   1. MSISDN exact de chaque MS          → routage precis (priorite max)
#   2. Prefixe court par operateur        → fallback pour MSISDN inconnus
#   3. Prefixes E.164 (+336XX...)           → routage international
#
ROUTES_HEADER

    # ── Routes exactes par MS (priorite maximale) ──────────────────
    for i in $(seq "${OP_ID_BASE:-1}" "$(( ${OP_ID_BASE:-1} + n_ops - 1 ))"); do
        local mnc; mnc=$(printf '%02d' "$i")
        printf '\n# ── Operateur %s (SC=%s) ──────────────────────────────────\n' \
               "$i" "$(_sms_sc_address "$i")" >> "$outfile"
        printf '# %s MS declares\n' "${nms[$i]}" >> "$outfile"

        for ms in $(seq 1 "${nms[$i]}"); do
            local msisdn; msisdn=$(_sms_op_msisdn "$i" "$ms")
            local imsi; imsi=$(_sms_op_ms_imsi "$mcc" "$mnc" "$i" "$ms")
            # Commentaire IMSI pour tracabilite
            printf '%-8s = %s   # IMSI=%s\n' "$msisdn" "$i" "$imsi" >> "$outfile"
```

```
        done

        # Prefixe court de l'operateur (fallback MSISDN hors liste)
        local op_prefix="${i}0000"
        printf '# Fallback operateur %s (MSISDN inconnu)\n' "$i" >> "$outfile"
        printf '%-8s = %s\n' "$op_prefix" "$i" >> "$outfile"

        # Prefixe E.164 fictif : +336<op>X... (format francais simule)
        local e164_prefix; e164_prefix=$(printf '336%02d' "$i")
        printf '%-8s = %s   # E.164 +33 6%02d...\n' "$e164_prefix" "$i" "$i" >> "$outfile"
    done

    cat >> "$outfile" << ROUTES_FOOTER

[relay]
# Port TCP sur lequel ce relay ecoute les MT entrants d'autres operateurs.
port = 7890
# Timeout connexion vers un relay distant (secondes)
connect_timeout = 10
# Tentatives de reemission si le relay distant est indisponible
retry_count = 3
retry_delay = 5
ROUTES_FOOTER

    _log_ok "sms-routing-op${op_id}.conf → ${outfile}"
}


# ═══════════════════════════════════════════════════════════════════
# sms_routing_generate_all
#
# Genere les configs pour TOUS les operateurs en une seule passe.
# Appele depuis start-docker.sh en mode bridge.
#
# Usage :
#   sms_routing_generate_all <n_ops> <destdir> [ms_counts...]
#
# ═══════════════════════════════════════════════════════════════════
sms_routing_generate_all() {
    local n_ops="$1"
    local destdir="$2"
    shift 2
    local ms_counts=("$@")

    echo -e "${CYAN}${BOLD}── SMS Routing - generation configs (${n_ops} operateurs) ──${NC}"

    for i in $(seq "${OP_ID_BASE:-1}" "$(( ${OP_ID_BASE:-1} + n_ops - 1 ))"); do
        sms_routing_generate "$i" "$n_ops" "$destdir" "${ms_counts[@]}"
    done

    echo -e "  ${GREEN}✓ Configs generees dans ${destdir}${NC}"
}


# ═══════════════════════════════════════════════════════════════════
# sms_routing_validate
#
# Verifie la coherence d'un fichier sms-routing.conf :
#   • Section [local] presente
#   • Section [operators] non vide
#   • Section [routes] non vide
#   • Aucune collision de prefixe exact entre deux operateurs differents
#   • MSISDN locaux routes vers l'operateur local
#
# Retourne 0 si OK, 1 si erreurs.
#
```

```
# Usage :
#   sms_routing_validate <conf_file>
#
# ═══════════════════════════════════════════════════════════════════════
sms_routing_validate() {
    local conf="$1"
    local errors=0

    echo -e "${CYAN}Validation : ${conf}${NC}"

    if [ ! -f "$conf" ]; then
        _log_err "Fichier introuvable : $conf"
        return 1
    fi

    # ── Sections obligatoires ──────────────────────────────────────────
    for section in local operators routes; do
        if ! grep -q "^\[${section}\]" "$conf"; then
            _log_err "Section manquante : [${section}]"
            errors=$(( errors + 1 ))
        fi
    done

    # ── operator_id dans [local] ───────────────────────────────────────
    local local_op
    local_op=$(awk '/^\[local\]/,/^\[/' "$conf" \
               | grep -E '^\s*operator_id\s*=' \
               | head -1 | cut -d= -f2 | tr -d ' ')
    if [ -z "$local_op" ]; then
        _log_err "[local] : operator_id manquant"
        errors=$(( errors + 1 ))
    else
        _log_ok "[local] operator_id = ${local_op}"
    fi

    # ── Au moins 1 operateur declare ──────────────────────────────────
    local op_count
    op_count=$(awk '/^\[operators\]/,/^\[/' "$conf" \
               | grep -cE '^\s*[0-9]+\s*=' || echo 0)
    if [ "$op_count" -eq 0 ]; then
        _log_err "[operators] : aucun operateur declare"
        errors=$(( errors + 1 ))
    else
        _log_ok "[operators] : ${op_count} operateur(s)"
    fi

    # ── Routes non vides ──────────────────────────────────────────────
    local route_count
    route_count=$(awk '/^\[routes\]/,/^\[/' "$conf" \
                  | grep -cE '^\s*[0-9]+\s*=' || echo 0)
    if [ "$route_count" -eq 0 ]; then
        _log_err "[routes] : aucune route definie"
        errors=$(( errors + 1 ))
    else
        _log_ok "[routes] : ${route_count} entree(s)"
    fi

    # ── Collision de prefixes : meme MSISDN → deux operateurs differents ─
    local collision_count=0
    while IFS= read -r line; do
        [[ "$line" =~ ^[[:space:]]*# ]] && continue   # commentaire
        [[ "$line" =~ ^[[:space:]]*$ ]] && continue   # vide
        [[ "$line" =~ ^\[ ]]            && continue   # section

        local prefix op
        prefix=$(echo "$line" | cut -d= -f1 | tr -d ' ')
```

```
        op=$(echo "$line" | cut -d= -f2 | cut -d'#' -f1 | tr -d ' ')

        # Chercher si ce meme prefixe apparait avec un operateur different
        local other_op
        other_op=$(awk -v p="$prefix" -v o="$op" '
            /^\[routes\]/,/^\[/ {
                if ($0 ~ "^[[:space:]]*" p "[[:space:]]*=") {
                    split($0, a, "=")
                    gsub(/[[:space:]]/, "", a[2])
                    sub(/#.*/, "", a[2])
                    if (a[2] != o) { print a[2]; exit }
                }
            }' "$conf")

        if [ -n "$other_op" ]; then
            _log_err "Collision : prefixe ${prefix} → op${op} ET op${other_op}"
            collision_count=$(( collision_count + 1 ))
            errors=$(( errors + 1 ))
        fi
    done < <(awk '/^\[routes\]/,/^\[/' "$conf")

    [ "$collision_count" -eq 0 ] && _log_ok "Pas de collision de prefixes"

    # ── Routes MSISDN locaux → operateur local ──────────────────────────
    if [ -n "$local_op" ]; then
        local wrong_local=0
        while IFS= read -r line; do
            [[ "$line" =~ ^[[:space:]]*# ]] && continue
            [[ "$line" =~ ^[[:space:]]*$ ]] && continue
            [[ "$line" =~ ^\[ ]]           && continue

            local prefix op
            prefix=$(echo "$line" | cut -d= -f1 | tr -d ' ')
            op=$(echo "$line" | cut -d= -f2 | cut -d'#' -f1 | tr -d ' ')

            # Un MSISDN commencant par op_id devrait router vers op_id
            if [[ "$prefix" =~ ^${local_op}[0-9]* ]] && [ "$op" != "$local_op" ]; then
                _log_warn "MSISDN local ${prefix} route vers op${op} ≠ op${local_op}"
                wrong_local=$(( wrong_local + 1 ))
            fi
        done < <(awk '/^\[routes\]/,/^\[/' "$conf")
        [ "$wrong_local" -eq 0 ] && _log_ok "MSISDN locaux correctement routes"
    fi

    # ── Resultat ────────────────────────────────────────────────────────
    echo ""
    if [ "$errors" -eq 0 ]; then
        echo -e "  ${GREEN}${BOLD}✓ Validation OK${NC}"
        return 0
    else
        echo -e "  ${RED}${BOLD}✗ ${errors} erreur(s)${NC}"
        return 1
    fi
}


# ═══════════════════════════════════════════════════════════════════
# sms_routing_summary
#
# Affiche la table de routage complete en format lisible (debug / audit).
# Montre tous les MSISDN enregistres et leur operateur cible.
#
# Usage :
#   sms_routing_summary <n_ops> [ms_counts...]
#
# ═══════════════════════════════════════════════════════════════════
```

```
sms_routing_summary() {
    local n_ops="$1"
    shift
    local ms_counts=("$@")

    local mcc="${MCC:-001}"

    echo -e "${CYAN}${BOLD}"
    echo "╔══════════════════════════════════════════════════════════════════════╗"
    echo "║                  SMS Routing - Table complete                        ║"
    printf "║  %d operateur(s)                                                     ║\n" "$n_ops"
    echo "╚══════════════════════════════════════════════════════════════════════╝"
    echo -e "${NC}"

    # En-tete tableau
    printf "${BOLD}%-12s %-8s %-20s %-18s %-15s${NC}\n" \
           "MSISDN" "Op→" "IMSI" "Container IP" "SC-address"
    printf "%-12s %-8s %-20s %-18s %-15s\n" \
           "────────────" "────────" "────────────────────" "──────────────────" "───────────────"

    local total_ms=0
    for i in $(seq "${OP_ID_BASE:-1}" "$(( ${OP_ID_BASE:-1} + n_ops - 1 ))"); do
        # ms_counts est POSITIONNELLE (0-based) alors que i est un RANG
        # d operateur, qui ne commence plus forcement a 1. Le (i-1)
        # d origine lisait donc l element du VOISIN : avec les rangs 2 et
        # 3, op2 heritait du compte d op3 et op3 tombait sur un element
        # absent, donc sur le defaut de 1 MS. La table affichait "3 MS"
        # pour quatre abonnes reellement provisionnes.
        local n_ms=${ms_counts[$((i - ${OP_ID_BASE:-1}))]:-1}
        local mnc; mnc=$(printf '%02d' "$i")
        local container_ip; container_ip=$(_sms_op_backbone_ip "$i")
        local sc; sc=$(_sms_sc_address "$i")

        for ms in $(seq 1 "$n_ms"); do
            local msisdn; msisdn=$(_sms_op_msisdn "$i" "$ms")
            local imsi; imsi=$(_sms_op_ms_imsi "$mcc" "$mnc" "$i" "$ms")
            printf "%-12s ${CYAN}Op%-6s${NC} %-20s %-18s %-15s\n" \
                   "$msisdn" "$i" "$imsi" "$container_ip" "$sc"
            total_ms=$(( total_ms + 1 ))
        done

        # Ligne de separation entre operateurs
        [ "$i" -lt "$n_ops" ] && \
            printf "${YELLOW}%-12s %-8s %-20s %-18s %-15s${NC}\n" \
                   "──────────" "↓ Op$((i+1))" "" "" ""
    done

    echo ""
    printf "${BOLD}Total : %d MS  |  %d operateur(s)${NC}\n" "$total_ms" "$n_ops"

    echo ""
    echo -e "${CYAN}── Reseau inter-operateurs ──${NC}"
    for i in $(seq "${OP_ID_BASE:-1}" "$(( ${OP_ID_BASE:-1} + n_ops - 1 ))"); do
        # ms_counts est POSITIONNELLE (0-based) alors que i est un RANG
        # d operateur, qui ne commence plus forcement a 1. Le (i-1)
        # d origine lisait donc l element du VOISIN : avec les rangs 2 et
        # 3, op2 heritait du compte d op3 et op3 tombait sur un element
        # absent, donc sur le defaut de 1 MS. La table affichait "3 MS"
        # pour quatre abonnes reellement provisionnes.
        local n_ms=${ms_counts[$((i - ${OP_ID_BASE:-1}))]:-1}
        printf "  Op%-2s  %s  ← TCP 7890  (relay)  %d MS\n" \
               "$i" "$(_sms_op_backbone_ip "$i")" "$n_ms"
    done

    echo ""
    echo -e "${CYAN}── Exemple de flux SMS inter-op ──${NC}"
```

```
    if [ "$n_ops" -ge 2 ]; then
        local src_msisdn; src_msisdn=$(_sms_op_msisdn 1 1)
        local dst_msisdn; dst_msisdn=$(_sms_op_msisdn 2 1)
        printf "  %s (Op1) ──GSUP──► HLR(Op1) ──► proto-smsc-daemon\n" "$src_msisdn"
        printf "    ──► sms-interop-relay(Op1) [lookup %s → Op2]\n" "$dst_msisdn"
        printf "    ──TCP──► sms-interop-relay(Op2) @ %s:7890\n" "$(_sms_op_backbone_ip 2)"
        printf "    ──► HLR(Op2) VTY → IMSI lookup\n"
        printf "    ──► proto-smsc-sendmt(Op2) → MS\n"
    fi
    echo ""
}


# ═══════════════════════════════════════════════════════════════════════
# sms_routing_mount_args
#
# Retourne les arguments -v Docker pour monter la config SMS du bon operateur.
# Utilise dans start_operator() de start-docker.sh.
#
# Usage :
#   vol_args+=$(sms_routing_mount_args <op_id> <destdir>)
#
# ═══════════════════════════════════════════════════════════════════════
sms_routing_mount_args() {
    local op_id="$1"
    local destdir="$2"
    local conf="${destdir}/sms-routing-op${op_id}.conf"

    if [ -f "$conf" ]; then
        echo "-v ${conf}:/etc/osmocom/sms-routing.conf"
    else
        echo "" # pas de volume si le fichier n'existe pas encore
    fi
}


# ═══════════════════════════════════════════════════════════════════════
# sms_routing_wait_ready
#
# Attend que le relay SMS d'un operateur soit pret (port TCP 7890 en ecoute).
# Appele optionnellement apres start_operator() pour s'assurer que le relay
# est disponible avant d'injecter des SMS de test.
#
# Usage :
#   sms_routing_wait_ready <container_name> [timeout_sec]
#
# ═══════════════════════════════════════════════════════════════════════
sms_routing_wait_ready() {
    local container="$1"
    local timeout="${2:-60}"
    local elapsed=0

    echo -ne "  Attente relay SMS (${container}) "

    while [ "$elapsed" -lt "$timeout" ]; do
        if docker exec "$container" \
           bash -c "ss -tlnp | grep -q ':7890'" 2>/dev/null; then
            echo -e " ${GREEN}✓${NC}"
            return 0
        fi
        echo -n "."
        sleep 2
        elapsed=$(( elapsed + 2 ))
    done

    echo -e " ${YELLOW}timeout${NC}"
```

```
    return 1
}


# ============================================================================
# sms_routing_test_send
#
# Envoie un SMS de test entre deux MS de deux operateurs differents.
# Necessite que les containers soient demarres et les HLR alimentes.
#
# Usage :
#   sms_routing_test_send <src_container> <src_imsi> <dst_msisdn> <message>
#
# ============================================================================
sms_routing_test_send() {
    local src_container="$1"
    local src_imsi="$2"
    local dst_msisdn="$3"
    local message="$4"

    echo -e "${CYAN}Test SMS :${NC} ${src_container}  IMSI=${src_imsi} → MSISDN=${dst_msisdn}"

    if ! docker ps --format '{{.Names}}' | grep -q "^${src_container}$"; then
        _log_err "Container ${src_container} non demarre"
        return 1
    fi

    docker exec "$src_container" bash -c "
        OPERATOR_ID=\${OPERATOR_ID:-1}
        SC_ADDRESS=\"\$(_sms_sc_address \$OPERATOR_ID)\" 2>/dev/null \
            || SC_ADDRESS=\"1999001\${OPERATOR_ID}444\"
        sms-encode-text '${message}' \
            | gen-sms-deliver-pdu \"\$SC_ADDRESS\" \
            | proto-smsc-sendmt \"\$SC_ADDRESS\" '${src_imsi}' /tmp/sendmt_socket
    " && _log_ok "SMS envoye" || _log_err "Echec envoi SMS"
}


# ============================================================================
# Point d'entree CLI (appel direct en ligne de commande)
# ============================================================================
_usage() {
    cat << 'EOF'
Usage: sms-routing-setup.sh <commande> [args...]

Commandes :
  generate <op_id> <n_ops> <destdir> [ms_counts...]
      Genere sms-routing-op<op_id>.conf dans <destdir>.
      ms_counts : nombre de MS par operateur (ex: 2 3 1 pour 3 ops).

  generate-all <n_ops> <destdir> [ms_counts...]
      Genere les configs pour TOUS les operateurs.

  validate <conf_file>
      Valide un fichier sms-routing.conf et affiche les erreurs.

  summary <n_ops> [ms_counts...]
      Affiche la table de routage complete (audit / debug).

  wait-ready <container_name> [timeout_sec]
      Attend que le relay SMS soit actif dans un container.

  test-send <container> <src_imsi> <dst_msisdn> <message>
      Envoie un SMS de test via proto-smsc-sendmt.

Variables d'environnement :
```

```
  MCC   : Mobile Country Code (defaut: 001)

Exemples :
  bash sms-routing-setup.sh generate-all 3 /tmp/sms-cfg 2 2 3
  bash sms-routing-setup.sh validate /tmp/sms-cfg/sms-routing-op1.conf
  bash sms-routing-setup.sh summary 2 2 3
EOF
}

if [[ "${BASH_SOURCE[0]}" == "${0}" ]]; then
    # Execute directement (pas source)
    CMD="${1:-help}"; shift 2>/dev/null || true
    case "$CMD" in
        generate)      sms_routing_generate    "$@" ;;
        generate-all)  sms_routing_generate_all "$@" ;;
        validate)      sms_routing_validate     "$@" ;;
        summary)       sms_routing_summary      "$@" ;;
        wait-ready)    sms_routing_wait_ready   "$@" ;;
        test-send)     sms_routing_test_send    "$@" ;;
        help|-h|--help) _usage ;;
        *)
            echo -e "${RED}Commande inconnue : ${CMD}${NC}" >&2
            _usage; exit 1 ;;
    esac
fi
```

3.353 /opt/GSM/osmo-operator/scripts/sms_notify.py

2956 octets, 77 lignes → 77 lignes

```
#!/usr/bin/env python3
"""sms_notify.py - PREVENIR LES MODEMS DU BANC QU UN SMS ARRIVE.

POURQUOI CE FICHIER EXISTE. osmo-msc du banc tourne avec « sms-over-gsup » :
un SM-RP recu de la radio part au HLR (gsm_04_11.c:742) sans jamais passer par
gsm340_rx_tpdu(), donc sans jamais atteindre sms_route_mt_sms() ni le serveur
SMPP. Le « smpp-first » et les « route prefix ... 100101 » de osmo-msc.cfg sont
de la configuration morte tant que sms-over-gsup est la : un ESME lie sur le
port 2775 ne recoit RIEN. Et le MT qui revient par GSUP est pousse directement
a l abonne par la voie radio - le mobile osmocom-bb l a, les modems logiciels
(Android via oFono, la VM postmarketOS via son port serie PCI) ne l ont pas.

D ou cette voie de service : la ou la passerelle decide d une remise locale,
elle previent aussi les modems, qui presentent le message a leur systeme sous
la forme AT attendue. On ne touche pas au chemin GSUP, donc le mobile continue
de recevoir comme avant - c est bien pour cela qu on ne duplique RIEN ici.

Deux emplois :
    import            : from sms_notify import notifier      (dans le relais)
    ligne de commande : sms_notify.py <imsi> <from> <texte>   (send-mt-sms.sh)
"""
import json
import os
import socket
import sys

# Les boites d entree des modems. Le meme programme (osmo-phonesim-banc.py)
# tourne DEUX fois sur ce banc et il lui faut donc deux ports : celui du modem
# d Android (mode ecoute) et celui du modem de la VM (mode --connect).
DEFAUT = "127.0.0.1:12348,127.0.0.1:12349"
TIMEOUT = 2.0


def cibles():
```

```
    val = os.environ.get("OSMO_SMS_NOTIFY", DEFAUT)
    out = []
    for morceau in val.split(","):
        morceau = morceau.strip()
        if not morceau:
            continue
        hote, _, port = morceau.rpartition(":")
        try:
            out.append((hote or "127.0.0.1", int(port)))
        except ValueError:
            continue
    return out


def notifier(imsi, sender, text, log=None):
    """Remet une copie du SMS aux modems. Best effort : un modem absent n est
    pas une erreur (la VM peut etre eteinte), on ne leve jamais."""
    charge = (json.dumps({"imsi": imsi, "from": sender, "text": text},
                         ensure_ascii=False) + "\n").encode()
    touches = 0
    for hote, port in cibles():
        try:
            s = socket.create_connection((hote, port), timeout=TIMEOUT)
        except OSError:
            continue                       # pas de modem sur ce port : normal
        try:
            s.sendall(charge)
            touches += 1
        except OSError:
            pass
        finally:
            s.close()
    if log:
        log("modems prevenus : %d" % touches)
    return touches


if __name__ == "__main__":
    if len(sys.argv) < 4:
        print("usage: sms_notify.py <imsi> <from> <texte>", file=sys.stderr)
        sys.exit(2)
    n = notifier(sys.argv[1], sys.argv[2], " ".join(sys.argv[3:]))
    print("modems prevenus : %d" % n)
```

3.354 /opt/GSM/osmo-operator/scripts/smsc-start.sh

5634 octets, 144 lignes → 144 lignes

```
#!/bin/bash
# smsc-start.sh - Lance le proto-smsc-daemon + le relay interop
# Fix: attente HLR GSUP robuste + verification socket sendmt avec retry

set -euo pipefail

OPERATOR_ID="${OPERATOR_ID:-1}"
HLR_IP="${HLR_IP:-127.0.0.2}"
SMSC_IPA_NAME="SMSC-OP${OPERATOR_ID}"
MO_LOG="/var/log/osmocom/mo-sms-op${OPERATOR_ID}.log"
SENDMT_SOCKET="/tmp/sendmt_socket"
ROUTING_CONF="/etc/osmocom/sms-routing.conf"
RELAY_PORT="${RELAY_PORT:-7890}"

GREEN='\033[0;32m'; CYAN='\033[0;36m'; YELLOW='\033[0;33m'; RED='\033[0;31m'; NC='\033[0m'
```

```
echo -e "${GREEN}╔══════════════════════════════════════════════════════╗${NC}"
echo -e "${GREEN}║  SMS Gateway - Operateur ${OPERATOR_ID}                          ║${NC}"
echo -e "${GREEN}╚══════════════════════════════════════════════════════╝${NC}"
echo -e "  HLR        : ${CYAN}${HLR_IP}${NC}"
echo -e "  IPA name   : ${CYAN}${SMSC_IPA_NAME}${NC}"
echo -e "  MO log     : ${CYAN}${MO_LOG}${NC}"
echo -e "  MT socket  : ${CYAN}${SENDMT_SOCKET}${NC}"
echo -e "  Relay port : ${CYAN}${RELAY_PORT}${NC}"
echo -e "  Routing    : ${CYAN}${ROUTING_CONF}${NC}"

rm -f "$SENDMT_SOCKET"
mkdir -p /var/log/osmocom

# ── Attente HLR GSUP (port 4222) - robuste ──────────────────────
echo -ne "${YELLOW}[SMSC] Attente OsmoHLR GSUP (4222)"
retries=60
while [ $retries -gt 0 ]; do
    if bash -c "echo >/dev/tcp/${HLR_IP}/4222" 2>/dev/null; then
        echo -e " ${GREEN}✓${NC}"
        break
    fi
    # Fallback: verifier aussi via ss
    if ss -tln 2>/dev/null | grep -q ":4222 "; then
        echo -e " ${GREEN}✓${NC}"
        break
    fi
    echo -n "."
    sleep 1
    retries=$(( retries - 1 ))
done
if [ $retries -eq 0 ]; then
    echo -e " ${RED}TIMEOUT - HLR non disponible${NC}"
    echo -e "${YELLOW}[SMSC] Tentative de demarrage quand meme...${NC}"
fi

sleep 3

# ── 1. proto-smsc-daemon (background) ──────────────────────────
echo -e "${GREEN}[1/2] proto-smsc-daemon${NC}"

# Verifier que le binaire existe
if ! command -v proto-smsc-daemon >/dev/null 2>&1; then
    echo -e "  ${RED}proto-smsc-daemon introuvable dans PATH${NC}"
    echo -e "  ${YELLOW}SMS local uniquement via MSC VTY${NC}"
    # Lancer quand meme le relay pour les SMS entrants via TCP
else
    proto-smsc-daemon "$HLR_IP" "$SMSC_IPA_NAME" "$MO_LOG" "$SENDMT_SOCKET" &
    DAEMON_PID=$!
    echo -e "  PID: ${CYAN}${DAEMON_PID}${NC}"

    # Attente du socket sendmt avec timeout
    echo -ne "  Socket sendmt"
    for i in $(seq 1 30); do
        if [ -S "$SENDMT_SOCKET" ]; then
            echo -e " ${GREEN}✓${NC}"
            break
        fi
        echo -n "."
        sleep 1
    done
    if [ ! -S "$SENDMT_SOCKET" ]; then
        echo -e " ${YELLOW}non cree (les MT SMS ne fonctionneront pas)${NC}"
    fi
fi

# ── 2. Relay interop (background) ──────────────────────────────
```

```
echo -e "${GREEN}[2/2] sms-interop-relay${NC}"

# Verifier que le script existe
RELAY_SCRIPT="/etc/osmocom/sms-interop-relay.py"
if [ ! -f "$RELAY_SCRIPT" ]; then
    echo -e "  ${RED}${RELAY_SCRIPT} introuvable${NC}"
    echo -e "  ${YELLOW}SMS inter-operateur desactive${NC}"
else
    # --node-id : la cle de routage de sms-routing.conf est le NOEUD, pas
    # l'operateur (MSISDN = <noeud>00<operateur><rang>). Sans elle, le relais
    # tient pour locaux les numeros des autres noeuds et n'emet jamais.
    python3 "$RELAY_SCRIPT" \
        --config "$ROUTING_CONF" \
        --port "$RELAY_PORT" \
        --mo-log "$MO_LOG" \
        --operator-id "$OPERATOR_ID" \
        --node-id "${OSMO_WAN_NODE:-${WAN_NODE_ID:-1}}" &
    RELAY_PID=$!
    echo -e "  PID: ${CYAN}${RELAY_PID}${NC}"

    # Verifier que le port TCP est en ecoute
    echo -ne "  Relay TCP ${RELAY_PORT}"
    for i in $(seq 1 15); do
        if ss -tlnp 2>/dev/null | grep -q ":${RELAY_PORT} "; then
            echo -e " ${GREEN}✓${NC}"
            break
        fi
        echo -n "."
        sleep 1
    done
fi

echo ""
echo -e "${GREEN}═══════════════════════════════════════════════════${NC}"
echo -e "${GREEN}  SMS Gateway actif - Op${OPERATOR_ID}${NC}"
echo -e "${GREEN}  MT local : send-mt-sms.sh <imsi> 'msg'${NC}"
echo -e "${GREEN}  MO log   : tail -f ${MO_LOG}${NC}"
echo -e "${GREEN}═══════════════════════════════════════════════════${NC}"

cleanup() {
    echo -e "\n${YELLOW}[SMSC] Arret...${NC}"
    [ -n "${DAEMON_PID:-}" ] && kill $DAEMON_PID 2>/dev/null || true
    [ -n "${RELAY_PID:-}" ] && kill $RELAY_PID 2>/dev/null || true
    rm -f "$SENDMT_SOCKET"
    exit 0
}
trap cleanup SIGINT SIGTERM

# Attendre fin d'un processus
if [ -n "${DAEMON_PID:-}" ] && [ -n "${RELAY_PID:-}" ]; then
    wait -n $DAEMON_PID $RELAY_PID 2>/dev/null || true
elif [ -n "${DAEMON_PID:-}" ]; then
    wait $DAEMON_PID 2>/dev/null || true
elif [ -n "${RELAY_PID:-}" ]; then
    wait $RELAY_PID 2>/dev/null || true
else
    # Aucun processus a attendre - juste dormir
    while true; do sleep 60; done
fi

echo -e "${YELLOW}[SMSC] Processus termine, arret...${NC}"
cleanup
```

3.355 /opt/GSM/osmo-operator/scripts/status.sh

338 octets, 14 lignes → 14 lignes

```
#!/bin/bash
cmd="${1:-status}"
set -ex
systemctl daemon-reload
systemctl $cmd osmo-hlr \
               osmo-msc \
               osmo-mgw \
               osmo-stp \
               osmo-bsc \
               osmo-ggsn \
               osmo-sgsn \
               osmo-sip-connector \
               osmo-bts-trx \
               osmo-pcu
```

3.356 /opt/GSM/osmo-operator/scripts/voix-forks.sh

4316 octets, 75 lignes → 75 lignes

```
#!/bin/bash
# ============================================================================
#  scripts/voix-forks.sh - les deux reglages de la voix qui vivent dans les FORKS
# ============================================================================
#
#  [2026-09-09] POURQUOI CE SCRIPT EXISTE.
#  La voix du banc depend de deux lignes qui ne sont PAS dans osmo-operator :
#  elles sont dans qosmo (run_modules, cfgs), un depot que la mise a jour
#  resynchronise par « git reset --hard FETCH_HEAD ». Tout correctif pose a la
#  main dans ces arbres disparait au prochain demarrage. On les repose donc,
#  apres chaque fetch, depuis ici - et ce fichier est le SEUL a savoir comment.
#
#  1. « io-tch-format rtp » dans cfgs/mobile_group1.cfg.
#     C est le gabarit que run.sh (run_modules/20-mobile-cfg.sh) RECOPIE
#     par-dessus ~/.osmocom/bb/mobile_group1.cfg a chaque demarrage - il
#     ecrase donc ce que start-native.sh a genere. Sans cette ligne, le mobile
#     lit et ecrit ses trames TCH dans la disposition Texas Instruments des
#     vrais Calypso, alors que pont.py les code avec libosmocoding, c est-a-dire
#     en disposition RTP/RFC3551 (33 octets ouverts par la signature 0xD). Les
#     deux bouts ne se comprennent DANS AUCUN SENS, aucun compteur ne bronche,
#     et la voix sort robotisee. Constate le 09/09 : les TROIS configurations
#     du banc tournaient sans elle.
#
#  2. CALYPSO_PULSE_LATENCY_MSEC dans run_modules/70-l2.sh et
#     68-sidecar-mobile.sh. Le mobile ecrit une trame GSM (160 echantillons,
#     20 ms) par snd_pcm_writei ; si le tampon du greffon ALSA-PulseAudio est
#     trop court, chaque ecriture rend -EPIPE, et pq_alsa.c y repond par un
#     snd_pcm_prepare() muet qui DEMONTE ET REMONTE le flux PulseAudio.
#     Mesure du 09/09, ecrivain cadence a 20 ms comme le mobile, 10 s :
#         80 ms -> 16 remontages   120 -> 19   160 -> 1   200 -> 0   240 -> 0
#     Pendant un appel reel : 50 par seconde. start-native.sh exporte 320, ce
#     qui suffit ; on aligne le defaut des forks pour qui lance run.sh seul.
#
#  Appelants : update.sh (osmo_poser_voix_forks), le service osmo-update
#  genere par iso_modules/81-cloture-systeme.sh (juste apres le fetch), et
#  iso_modules/70-scripts.sh (sur le ROOTFS, a la construction de l ISO).
#
#  Usage : voix-forks.sh [prefixe]     (prefixe = un ROOTFS, vide par defaut)
#  Idempotent, sans effet si les forks sont absents, toujours exit 0.
#
```

```
#  JUGE : `grep io-tch-format ~/.osmocom/bb/mobile_group1.cfg` apres un
#  demarrage ; et pendant un appel etabli, l index du sink-input du mobile doit
#  rester FIXE (`pactl list short sink-inputs`, deux releves a 1 s d intervalle).
# ------------------------------------------------------------------------------
set -u
ROOT="${1:-}"
VOIX_LATENCE="${CALYPSO_PULSE_LATENCY_MSEC_DEFAUT:-320}"
VOIX_TCH_FORMAT="${MS_TCH_FORMAT:-rtp}"

voix_forks_poser() {
    local f n fait=0
    for f in "$ROOT/opt/GSM/qosmo/cfgs/mobile_group1.cfg"; do
        [ -f "$f" ] || continue
        grep -q '^[[:space:]]*io-tch-format' "$f" && continue
        # Juste apres « io-handler », dans le bloc tch-voice, meme indentation.
        if sed -i '/^[[:space:]]*tch-voice[[:space:]]*$/,/^[[:space:]]*no shutdown/
         ↪  s/^\([[:space:]]*\)io-handler\(.*\)$/\1io-handler\2\n\1io-tch-format '"$VOIX_TCH_FORMAT"'/' "$f" \
           && grep -q "io-tch-format ${VOIX_TCH_FORMAT}" "$f"; then
            echo "  [voix] ${f#"$ROOT"} : io-tch-format ${VOIX_TCH_FORMAT} repose (voix robotisee sinon)"; fait=1
        else
            echo "  [voix] ${f#"$ROOT"} : io-tch-format NON pose - a regarder a la main" >&2
        fi
    done
    for f in "$ROOT/opt/GSM/qosmo/run_modules/70-l2.sh" \
             "$ROOT/opt/GSM/qosmo/run_modules/68-sidecar-mobile.sh"; do
        [ -f "$f" ] || continue
        n="$(sed -n 's/^: "${CALYPSO_PULSE_LATENCY_MSEC:=\([0-9]*\)}"/\1/p' "$f" | head -1)"
        [ -n "$n" ] || continue
        [ "$n" -ge "$VOIX_LATENCE" ] 2>/dev/null && continue
        sed -i "s|^: \"\${CALYPSO_PULSE_LATENCY_MSEC:=[0-9]*}\"|:
         ↪  \"\\\${CALYPSO_PULSE_LATENCY_MSEC:=${VOIX_LATENCE}}\"|" "$f" \
            && { echo "  [voix] ${f#"$ROOT"} : tampon du mobile ${n} -> ${VOIX_LATENCE} ms (flux PulseAudio
             ↪  remonte sinon)"; fait=1; }
    done
    return 0
}
voix_forks_poser
exit 0
```

3.357 /opt/GSM/osmo-operator/scripts/wslg-audio-bridge.sh

15820 octets, 320 lignes → 320 lignes

```
#!/bin/bash
# wslg-audio-bridge.sh - Fait sortir l'audio des conteneurs Osmocom sur les
# haut-parleurs de l'HOTE, depuis l'INTERIEUR du Docker. Marche dans deux cas :
#
#   • WSL2 + WSLg  : l'hote audio = PulseServer WSLg (/mnt/wslg/PulseServer),
#                    qui relaie lui-meme vers Windows.
#   • Linux natif  : l'hote audio = le PulseAudio/PipeWire de la session
#                    graphique (/run/user/<uid>/pulse/native).
#
# Principe (verifie) :
#   host-relay  → expose le pulse de l'hote en TCP (module-native-protocol-tcp)
#   container   → pont AUDIO direct dans le conteneur :
#                     parec(gsm_audio.monitor) | paplay(--server=tcp:<gw>:4713)
#
# On N'utilise PAS module-tunnel-sink : sur TCP/WSL son horloge derive (la
# latence grimpe a >1 s → son hache "tout pete"). parec|paplay = un seul
# serveur d'horloge (celui de l'hote) + simple pipe d'octets → audio propre.
#
# Le conteneur garde son PulseAudio autonome ; on ne touche pas a gsm_audio
# (le dashboard continue de lire gsm_audio.monitor).
#
```

```
# Usage :
#   wslg-audio-bridge.sh host-relay              # hote : ouvre le pulse en TCP
#   wslg-audio-bridge.sh container <name> [gw]   # conteneur : lance le pont
#   wslg-audio-bridge.sh test <name>             # bip 880Hz via gsm_audio
#   wslg-audio-bridge.sh down <name>             # stoppe le pont
#   wslg-audio-bridge.sh all <name> [gw]         # host-relay + container + test
set -euo pipefail

TCP_PORT="${WSLG_TCP_PORT:-4713}"
# ACL : tout le bloc prive 172.16/12 (couvre les subnets docker gsm-inter/op)
TCP_ACL="${WSLG_TCP_ACL:-127.0.0.1;172.16.0.0/12}"

log()  { echo -e "  [host-audio] $*"; }
die()  { echo -e "  [host-audio][FAIL] $*" >&2; exit 1; }

# IP de l'hote vue depuis le conteneur = gateway de sa route par defaut.
detect_gw() {
    docker exec "$1" sh -c "ip route 2>/dev/null | awk '/default/{print \$3; exit}'" 2>/dev/null || true
}

# Utilisateur de la session graphique (proprietaire du pulse natif).
# [2026-08-12] Le critere n'est pas le nom mais la PRESENCE du socket pulse :
# sous `sudo -i` SUDO_USER est perdu et `id -un` rend root, dont le
# /run/user/0/pulse/native n'existe quasiment jamais - on partait alors chercher
# un pulse inexistant au lieu de celui, bien vivant, de la session graphique.
# On balaye donc /run/user/* en dernier recours. Meme logique que
# session_pulse_user() de start-docker.sh.
session_user() {
    local u uid sock
    for u in "${HOST_PULSE_USER:-}" "${SUDO_USER:-}" "$(logname 2>/dev/null || true)" "$(id -un)"; do
        [ -n "$u" ] || continue
        uid="$(id -u "$u" 2>/dev/null)" || continue
        [ -S "/run/user/${uid}/pulse/native" ] && { echo "$u"; return 0; }
    done
    for sock in /run/user/*/pulse/native; do
        [ -S "$sock" ] || continue
        uid="${sock#/run/user/}"; uid="${uid%%/*}"
        u="$(getent passwd "$uid" 2>/dev/null | cut -d: -f1)"
        [ -n "$u" ] && { echo "$u"; return 0; }
    done
    echo "${HOST_PULSE_USER:-${SUDO_USER:-$(id -un)}}"   # rien trouve : ancien defaut
}

# Execute `pactl ...` sur le pulse de l'hote natif, en tant qu'utilisateur session.
pactl_native() {
    local u uid rt
    u="$(session_user)"; uid="$(id -u "$u")"; rt="/run/user/${uid}"
    if [ "$(id -un)" = "$u" ]; then
        XDG_RUNTIME_DIR="$rt" PULSE_SERVER="unix:${rt}/pulse/native" pactl "$@"
    else
        sudo -u "$u" XDG_RUNTIME_DIR="$rt" PULSE_SERVER="unix:${rt}/pulse/native" pactl "$@"
    fi
}

# Execute un outil pulse quelconque (pactl, parec, timeout parec...) contre le
# pulse de l'HOTE, en WSLg comme en natif. Generalise pactl_native.
host_pa() {
    local bin="$1"; shift
    if [ -S /mnt/wslg/PulseServer ]; then
        PULSE_SERVER="unix:/mnt/wslg/PulseServer" "$bin" "$@"
    else
        local u uid rt
        u="$(session_user)"; uid="$(id -u "$u")"; rt="/run/user/${uid}"
        if [ "$(id -un)" = "$u" ]; then
            XDG_RUNTIME_DIR="$rt" PULSE_SERVER="unix:${rt}/pulse/native" "$bin" "$@"
        else
```

```
            sudo -u "$u" XDG_RUNTIME_DIR="$rt" PULSE_SERVER="unix:${rt}/pulse/native" "$bin" "$@"
        fi
    fi
}

# ── HOTE : choisir une source micro qui a REELLEMENT du signal ──────────────
#
# [2026-08-12] Panne vecue : le micro du dashboard web (bouton -> WebSocket ->
# pacat -> sink gsm_mic) debitait au bon rythme (49400 o/3 s a 8 kHz s16le) mais
# ne transportait que du ZERO. Cause : le navigateur capture la source PAR
# DEFAUT du systeme, et sur cette machine le defaut pointait sur
# alsa_input...hw_Generic_1__source - presente, NON mutee, volume 93 %, et
# pourtant silence numerique integral. La vraie entree etait hw_acp__source.
# `pactl` ne permet pas de distinguer les deux : il faut ECOUTER.
#
# On ne code donc aucun nom de peripherique (ils changent d'une machine a
# l'autre) : on mesure. Critere volontairement binaire - un micro vivant a
# TOUJOURS un plancher de bruit, donc des octets non nuls ; une entree mal
# routee ne sort que des zeros. `tr -d '\0' | wc -c` suffit : pas de dependance
# python/sox, ca tourne partout ou parec existe.
# ATTENTION AU FAUX NEGATIF : une source SUSPENDUE met ~2 s a delivrer ses
# premiers octets (mesure : timeout 1s -> 0 octet, timeout 3s -> 32000 octets
# dont 31946 non nuls, sur la MEME source vivante). Juger sur une fenetre courte
# revient a declarer muet tout peripherique au repos. On distingue donc deux
# cas distincts, jamais confondus :
#   - RIEN capture            -> le peripherique n'a pas demarre (indetermine)
#   - capture mais TOUT A ZERO -> silence numerique (la vraie panne)
# On lit jusqu'a MIC_PROBE_BYTES puis on coupe (head ferme le tube, parec recoit
# SIGPIPE) : rapide sur un peripherique vivant, seul un mort coute le timeout.
MIC_PROBE_SECS="${MIC_PROBE_SECS:-4}"
# Seconde chance pour un peripherique QUI N'A RIEN RENDU (verdict 2). Le
# commentaire ci-dessus mesure ~2 s de reveil sur une source suspendue ; une
# carte USB derriere un hub, ou un bluetooth qui doit negocier son profil, en
# demande davantage. Ce parametre existe pour ne pas condamner ces machines-la.
MIC_PROBE_SECS_SLOW="${MIC_PROBE_SECS_SLOW:-12}"
MIC_PROBE_BYTES="${MIC_PROBE_BYTES:-16000}"   # 1 s @ 8 kHz s16le mono
MIC_MIN_BYTES="${MIC_MIN_BYTES:-4000}"        # en deca : pas de verdict possible
MIC_MIN_NONZERO="${MIC_MIN_NONZERO:-100}"
MIC_MAX_PROBES="${MIC_MAX_PROBES:-8}"
# 1 = ne JAMAIS toucher au defaut systeme de l'hote (on se contente de dire ce
# qu'on a mesure). Utile sur un poste de travail : la selection ci-dessous est
# permanente, elle survit a l'arret d'osmo-operator.
MIC_KEEP_DEFAULT="${MIC_KEEP_DEFAULT:-0}"

# Renvoie "<octets_captures> <octets_non_nuls>" sur stdout.
# $2 = duree de la fenetre (defaut MIC_PROBE_SECS).
probe_source() {
    local src="$1" secs="${2:-$MIC_PROBE_SECS}" tmp raw nz
    tmp="$(mktemp)"
    # `|| true` : timeout sort en 124, head provoque un SIGPIPE - sous
    # `set -o pipefail` l'un ou l'autre tuerait le script.
    { host_pa timeout "$secs" parec -d "$src" \
        --format=s16le --rate=8000 --channels=1 2>/dev/null || true; } \
      | head -c "$MIC_PROBE_BYTES" > "$tmp" 2>/dev/null || true
    raw="$(wc -c < "$tmp")"; nz="$(tr -d '\000' < "$tmp" | wc -c)"
    rm -f "$tmp"
    echo "${raw:-0} ${nz:-0}"
}

# 0 = signal, 1 = silence numerique, 2 = indetermine (rien capture)
source_verdict() {
    local r raw nz secs="${2:-$MIC_PROBE_SECS}"
    r="$(probe_source "$1" "$secs")"; raw="${r%% *}"; nz="${r##* }"
    [ "$raw" -ge "$MIC_MIN_BYTES" ] || return 2
    [ "$nz"  -ge "$MIC_MIN_NONZERO" ] || return 1
    return 0
```

```
}

# [2026-08-12] Le code distinguait soigneusement le verdict 1 (silence
# numerique = LA panne) du verdict 2 (rien capture = INDETERMINE)... puis
# traitait les deux pareil : dans les deux cas on partait chercher ailleurs et
# on reecrivait le defaut systeme. Un peripherique simplement lent a demarrer se
# faisait donc evincer au profit d'une entree qui, elle, n'a rien prouve de plus.
# "Indetermine" veut dire qu'il faut REGARDER PLUS LONGTEMPS, pas conclure.
verdict_retry() {
    local src="$1" rc=0
    source_verdict "$src" || rc=$?
    [ "$rc" = "2" ] || return "$rc"
    log "[mic]   ${src} : rien en ${MIC_PROBE_SECS}s - seconde chance sur ${MIC_PROBE_SECS_SLOW}s"
    rc=0; source_verdict "$src" "$MIC_PROBE_SECS_SLOW" || rc=$?
    return "$rc"
}

host_mic() {
    command -v parec >/dev/null 2>&1 \
        || { log "[mic] parec absent - selection micro ignoree"; return 0; }
    host_pa pactl info >/dev/null 2>&1 \
        || { log "[mic] pulse hote injoignable - selection micro ignoree"; return 0; }

    # RAPPEL DE PORTEE, sinon cette fonction se fait crediter d'un pouvoir
    # qu'elle n'a pas : elle regle le defaut du PULSE DE L'HOTE. Le micro du
    # tableau de bord, lui, est celui du POSTE QUI AFFICHE LA PAGE (getUserMedia
    # -> WebSocket -> pacat -> gsm_mic). Les deux ne coincident que si l'on
    # navigue depuis la machine hote. Depuis un autre poste, c'est le selecteur
    # de la page (web/index.html, #mic-dev) qui decide, et rien d'autre.
    log "[mic] selection de l'entree du PULSE DE L'HOTE (le navigateur, lui,"
    log "[mic]   utilise le micro de SON poste - selecteur dans le tableau de bord)."

    local cur rc; cur="$(host_pa pactl get-default-source 2>/dev/null || true)"
    if [ -n "$cur" ]; then
        # `|| rc=$?` OBLIGATOIRE : sous `set -e` un appel nu de source_verdict
        # renvoyant 1 (silence) ou 2 (rien capture) tuerait le script sur place,
        # sans une ligne de log - panne observee, tres deroutante.
        rc=0; verdict_retry "$cur" || rc=$?
        case "$rc" in
            0) log "[mic] source par defaut deja valide : ${cur}"; return 0 ;;
            1) log "[mic] defaut '${cur}' : SILENCE NUMERIQUE - recherche d'une autre entree..." ;;
            2) log "[mic] defaut '${cur}' : aucune donnee meme en ${MIC_PROBE_SECS_SLOW}s - recherche..." ;;
        esac
    fi

    if [ "$MIC_KEEP_DEFAULT" = "1" ]; then
        log "[mic] MIC_KEEP_DEFAULT=1 - defaut systeme laisse tel quel, pas de bascule."
        return 0
    fi

    local src i=0
    for src in $(host_pa pactl list short sources 2>/dev/null | awk '{print $2}' | grep -v '\.monitor$'); do
        [ "$src" = "$cur" ] && continue
        i=$((i + 1)); [ "$i" -gt "$MIC_MAX_PROBES" ] && break
        rc=0; verdict_retry "$src" || rc=$?        # cf. remarque set -e ci-dessus
        if [ "$rc" = "0" ]; then
            if host_pa pactl set-default-source "$src" >/dev/null 2>&1; then
                log "[mic] source par defaut -> ${src} (signal detecte)"
                # La bascule est PERMANENTE : pulse la garde apres l'arret
                # d'osmo-operator. On journalise l'ancienne valeur pour que le
                # retour en arriere soit une commande, pas une enquete.
                [ -n "$cur" ] && log "[mic]   ancien defaut : ${cur}"
                [ -n "$cur" ] && log "[mic]   restaurer : pactl set-default-source ${cur}"
                log "[mic]   (MIC_KEEP_DEFAULT=1 desactive cette bascule)"
                log "[mic]   un onglet qui tient deja le micro reste epingle sur"
                log "[mic]   l'ancien peripherique : recouper/relancer le bouton micro."
```

```
                return 0
            fi
        else
            log "[mic]   ${src} : $([ "$rc" = 1 ] && echo 'silence numerique' || echo 'aucune donnee')"
        fi
    done

    log "[mic][!] AUCUNE source micro ne produit de signal (${i} testee(s))."
    log "[mic][!]   -> micro coupe materiellement (touche/interrupteur) ?"
    log "[mic][!]   -> mauvais profil UCM : verifier 'pactl list cards' / alsamixer."
    log "[mic][!]   Sans ca l'echo-test rendra le HP mais jamais la voix."
    return 0
}

# ── HOTE : expose le pulse de l'hote en TCP (idempotent) ──────────────
host_relay() {
    if [ -S /mnt/wslg/PulseServer ]; then
        # WSL2 + WSLg : socket monde-accessible (root OK)
        export PULSE_SERVER="unix:/mnt/wslg/PulseServer"
        if pactl list short modules 2>/dev/null | grep -q "protocol-tcp.*port=${TCP_PORT}"; then
            log "relai TCP deja actif (WSLg) :${TCP_PORT}"; return 0
        fi
        pactl load-module module-native-protocol-tcp \
            "port=${TCP_PORT}" "auth-ip-acl=${TCP_ACL}" auth-anonymous=1 >/dev/null \
            || die "load module-native-protocol-tcp (WSLg)"
        log "relai TCP WSLg → Windows active :${TCP_PORT}"
    else
        # Linux natif : pulse de la session graphique
        local u uid; u="$(session_user)"; uid="$(id -u "$u")"
        [ -S "/run/user/${uid}/pulse/native" ] \
           || die "pulse de session introuvable (/run/user/${uid}/pulse/native) - session audio demarree ? user=$u"
        if pactl_native list short modules 2>/dev/null | grep -q "protocol-tcp.*port=${TCP_PORT}"; then
            log "relai TCP deja actif (natif, user=$u) :${TCP_PORT}"; return 0
        fi
        pactl_native load-module module-native-protocol-tcp \
            "port=${TCP_PORT}" "auth-ip-acl=${TCP_ACL}" auth-anonymous=1 >/dev/null \
            || die "load module-native-protocol-tcp (natif)"
        log "relai TCP natif active (user=$u) :${TCP_PORT}"
    fi
}

# ── CONTENEUR : pont parec|paplay supervise vers le pulse de l'hote ──────────
container_bridge() {
    local name="$1" gw="${2:-}"
    [ -n "$name" ] || die "nom de conteneur requis"
    [ -z "$gw" ] && gw="$(detect_gw "$name")"
    [ -n "$gw" ] || die "gateway introuvable pour $name"
    docker exec -i "$name" sh -s -- "tcp:${gw}:${TCP_PORT}" <<'EOSH'
set -eu
RELAY="$1"
pactl info >/dev/null 2>&1 || { echo "  [host-audio][FAIL] pas de daemon pulse dans le conteneur" >&2; exit 1; }
# Le relai hote doit etre joignable
if ! pactl --server="$RELAY" info >/dev/null 2>&1; then
    echo "  [host-audio][FAIL] relai hote injoignable ($RELAY) - lancer 'host-relay' d'abord" >&2; exit 1
fi
# Stoppe un pont precedent (idempotent)
pkill -f "paplay --server=${RELAY}" 2>/dev/null || true
[ -f /run/host-audio.pid ] && kill -- "-$(cat /run/host-audio.pid)" 2>/dev/null || true
mkdir -p /var/log/osmocom
# Pont supervise : redemarre si parec/paplay tombe. setsid = nouveau groupe → kill propre.
setsid sh -c '
  while true; do
    parec -d gsm_audio.monitor --latency-msec=30 --format=s16le --rate=8000 --channels=1 \
      | paplay --server='"${RELAY}"' --latency-msec=250 --raw --format=s16le --rate=8000 --channels=1
    sleep 1
  done' >/var/log/osmocom/host-audio.log 2>&1 &
```

```
echo $! > /run/host-audio.pid
echo "  [host-audio] pont parec|paplay gsm_audio.monitor -> ${RELAY} (pgid $!)"
EOSH
}

# ── CONTENEUR : bip de test via gsm_audio (preuve bout-en-bout) ──────────────
container_test() {
    docker exec -i "$1" sh -s <<'EOSH'
set -eu
ffmpeg -y -hide_banner -loglevel error -f lavfi -i "sine=frequency=880:duration=1" -ar 8000 -ac 1
 ↪  /tmp/host_test.wav
paplay -d gsm_audio /tmp/host_test.wav
echo "  [host-audio] bip 880Hz injecte dans gsm_audio -> hote"
EOSH
}

container_down() {
    docker exec -i "$1" sh -s <<'EOSH'
set -eu
[ -f /run/host-audio.pid ] && kill -- "-$(cat /run/host-audio.pid)" 2>/dev/null || true
pkill -f 'parec -d gsm_audio.monitor' 2>/dev/null || true
rm -f /run/host-audio.pid
echo "  [host-audio] pont arrete"
EOSH
}

cmd="${1:-}"; shift || true
case "$cmd" in
    host-relay) host_relay ;;
    host-mic)   host_mic ;;
    container)  container_bridge "${1:-}" "${2:-}" ;;
    test)       container_test "${1:-}" ;;
    down)       container_down "${1:-}" ;;
    all)        host_relay; host_mic; container_bridge "${1:-}" "${2:-}"; container_test "${1:-}" ;;
    *) echo "usage: $0 {host-relay|host-mic|container <name> [gw]|test <name>|down <name>|all <name> [gw]}" >&2;
     ↪  exit 2 ;;
esac
```

3.358 /opt/GSM/osmo-operator/services/80-osmo-operator.preset

2015 octets, 53 lignes → 53 lignes

```
# 80-osmo-operator.preset - POLITIQUE PAR DEFAUT : AUCUN SERVICE TELECOM AU BOOT
#
# [2026-10-23] Un reseau mobile se DEMARRE (icone, start.sh, systemctl start) :
# ce n est pas un effet de bord du demarrage de la machine. Ce preset systemd
# (pose dans /usr/lib/systemd/system-preset/, lu AVANT 90-systemd.preset) dit
# « disable » pour tout ce qui fait tourner un element de reseau : le banc et
# ses variantes, le coeur Osmocom, les BTS, Asterisk, oFono, Open5GS, srsRAN.
# Il ne touche a rien d autre : les consoles web (osmo-egprs-web,
# osmo-reglages, open5gs-webui), le son (osmo-pulse), le plan d adresses
# (osmo-ip-plan), la purge, les mises a jour et le bureau gardent leur etat.
#
# Applique par tools/osmo-telecom-boot.sh (ISO : iso_modules/82-services.sh ;
# machines deja installees : update.sh, une seule fois). Un service active
# ensuite a la main (systemctl enable osmo-banc) le reste : ce fichier ne sert
# qu a l installation et a `systemctl preset`, jamais en tache de fond.

# les bancs
disable osmo-banc.service
disable osmo-multi.service
disable osmo-lte.service
disable osmo-egprs-dsp.service
disable osmo-dsp-rom-max.service
```

```
disable osmo-interstp.service
# le coeur Osmocom et les stations de base
disable osmo-msc.service
disable osmo-bsc.service
disable osmo-hlr.service
disable osmo-mgw.service
disable osmo-stp.service
disable osmo-sgsn.service
disable osmo-ggsn.service
disable osmo-gtphub.service
disable osmo-gbproxy.service
disable osmo-pcu.service
disable osmo-sip-connector.service
disable osmo-cbc.service
disable osmo-smlc.service
disable osmo-hnbgw.service
disable osmo-hnodeb.service
disable osmo-bts-*.service
disable osmo-trx-*.service
disable osmo-nitb.service
# la voix et les modems
disable asterisk.service
disable ofono.service
disable kamailio.service
disable yate.service
# 4G / 5G (la console web open5gs-webui n est pas un element de reseau)
disable open5gs-*d.service
disable srsenb.service
disable srsepc.service
disable srsue.service
disable srsran*.service
```

3.359 /opt/GSM/osmo-operator/services/open5gs-webui.service

2414 octets, 50 lignes → 50 lignes

```
[Unit]
# ============================================================================
#  open5gs-webui.service - LA CONSOLE DES ABONNES 4G (Open5GS WebUI), sur :9999
# ============================================================================
#  [2026-09-10] Le banc portait la 4G complete (tools/osmo-lte.sh : Open5GS,
#  srsENB, srsUE) mais AUCUNE console : provisionner un abonne LTE se faisait a
#  la main dans MongoDB, ou par « osmo-epc subscribers » qui rejoue le JSON du
#  depot. Le WebUI d Open5GS est fait pour ca, il vit dans les sources
#  (webui/ du depot open5gs) et c est un Next.js : node server/index.js.
#
#  ELLE NE LIT PAS LES DEMONS, ELLE LIT MONGODB - la meme base que le HSS et le
#  PCRF (DB_URI). D ou After=mongod : sans base, le WebUI demarre et sert des
#  pages vides. Il ne DEPEND pas d Open5GS lui-meme : on provisionne des
#  abonnes coeur eteint, c est meme le moment habituel.
#
#  CONDITION, PAS DEPENDANCE. L unite est ACTIVEE par defaut (ISO et natif),
#  mais le WebUI n est pas toujours construit : les .deb du banc
#  (osmo-build-open5gs) n embarquent que le prefixe d installation, pas les
#  sources ni node_modules. Sans server/index.js, systemd saute l unite au lieu
#  de la marquer en echec - un service rouge sur un banc en parfait etat est
#  une fausse alarme, et le Conky les compte. « osmo-lte-install --webui » le
#  construit.
Description=Open5GS WebUI - console des abonnes 4G (:9999)
Documentation=https://open5gs.org/open5gs/docs/
After=network.target mongod.service
Wants=mongod.service
ConditionPathExists=/opt/LTE/open5gs/webui/server/index.js
```

```
[Service]
Type=simple
WorkingDirectory=/opt/LTE/open5gs/webui
Environment=NODE_ENV=production
# 0.0.0.0 : la console se regarde depuis le poste de l operateur, pas seulement
# depuis la machine du banc.
Environment=HOSTNAME=0.0.0.0
Environment=PORT=9999
# LA MEME base que hss.yaml et pcrf.yaml (configs/open5gs/) : une console qui
# ecrirait ailleurs provisionnerait des abonnes que le HSS ne verrait jamais.
Environment=DB_URI=mongodb://localhost/open5gs
# node vit en /usr/local/bin sur le banc (image) et en /usr/bin sur un systeme
# apt : env le trouve dans les deux cas, ExecStart exige un chemin absolu.
ExecStart=/usr/bin/env node server/index.js
Restart=on-failure
RestartSec=5
StandardOutput=journal
StandardError=journal
SyslogIdentifier=open5gs-webui

[Install]
WantedBy=multi-user.target
```

3.360 /opt/GSM/osmo-operator/services/osmo-banc.service

6661 octets, 106 lignes → 106 lignes

```
[Unit]
Description=osmo-operator - banc GSM autonome (start.sh, session tmux « calypso »)
Documentation=file:/opt/GSM/osmo-operator/README.md
# Le banc a besoin du son systeme (osmo-pulse), des adresses privees du noeud
# et du dashboard (le pont y pousse la FFT). Pas de Requires= : un de ces
# services absent ne doit pas empecher la radio de monter.
# [2026-09-04] PLUS D ATTENTE RESEAU AU BOOT. network-online.target etait tire
# ici ; il coute jusqu a 2 min de demarrage (systemd-networkd-wait-online,
# mesure : systemd-analyze blame) sur un banc dont les interfaces - apn0, les
# tun/veth du coeur paquet - ne sont jamais "en ligne" au sens de cette unite,
# et qui demarre souvent sans cable. network.target suffit : les interfaces
# sont configurees, la route par defaut peut arriver apres.
After=network.target osmo-pulse.service osmo-ip-plan.service osmo-egprs-web.service osmo-qemu-link.service
# osmo-egprs-web est TIRE par le banc : il porte PartOf=osmo-banc.service et
# systemd l arrete donc avec lui. Ce Wants= est la contrepartie obligatoire -
# sans lui, un cycle arret/relance laisserait le tableau de bord eteint.
Wants=osmo-pulse.service osmo-egprs-web.service
ConditionPathExists=/opt/GSM/osmo-operator/start.sh
# Un seul banc natif a la fois : le multi-operateur DEPEND de celui-ci
# (Requires= dans osmo-multi.service), il ne le remplace pas.
#
# [2026-09-05] POSEE, PAS ACTIVEE. Cette unite partait au boot depuis le debut :
# une machine fraichement installee montait QEMU/Calypso et tout le coeur avant
# meme l'ouverture de session - plusieurs minutes, et un banc lance sur une
# configuration (noeud, --dsp, topologie) que personne n'avait choisie.
#   systemctl start osmo-banc           pour une session (ou l'icone du bureau)
#   systemctl enable --now osmo-banc    pour le demarrer a chaque boot
# Au build : OSMO_ISO_BANC=1 (ou --banc) active l'unite dans l'image.

[Service]
# ── ONESHOT + RemainAfterExit : LE BANC VIT DANS TMUX, PAS DANS ExecStart ──
# start.sh passe la main a run.sh (qosmo-grgsm), qui monte tout dans
# une session tmux puis SORT sans s'y attacher (pas de tty). Avec Type=simple,
# systemd prendrait cette sortie pour la fin du service et tuerait le cgroup -
# donc la pile entiere. En oneshot + RemainAfterExit, l'unite reste « active
# (exited) » et les processus du cgroup (tmux, demons, qemu, pont) vivent
# jusqu'a ExecStop.
```

```
Type=oneshot
RemainAfterExit=yes
WorkingDirectory=/opt/GSM/osmo-operator
# ── LE MEME ENVIRONNEMENT QU'UN TERMINAL DE ROOT ──────────────────────────
# HOME : mobile.cfg vit dans /root/.osmocom/bb. PATH : le venv python
# (/root/.env) que ~/.bashrc active, gr-gsm et les ponts en dependent ; systemd
# ne lit pas ~/.bashrc. /etc/osmocom/coeur.env (N_MS) est charge par
# start.sh lui-meme.
Environment=HOME=/root USER=root LOGNAME=root
Environment=VIRTUAL_ENV=/root/.env
Environment=PATH=/root/.env/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
Environment=PULSE_SERVER=unix:/var/run/pulse/native
# Pas de menu, pas d'attache tmux, pas de couleurs : il n'y a personne devant.
Environment=NO_MENU=1 CALYPSO_NO_ATTACH=1 NO_COLOR=1 OSMO_NO_TERM=1 OSMO_LAUNCH_APPS=0
# Options de lancement (--grgsm, --op 2, --node 3, --regen...) ; le DSP est
# le defaut de start.sh depuis le 2026-10-01, --grgsm l ancienne chaine :
#   - durables : OSMO_BANC_ARGS="--grgsm" dans /etc/default/osmo-banc
#   - pour une fois : launch.sh les pose via systemctl set-environment
EnvironmentFile=-/etc/default/osmo-banc
# --no-attach : l option de start.sh (meme effet que CALYPSO_NO_ATTACH=1
# ci-dessus). Elle va ICI, sur la ligne de commande - pas sur
# ConditionPathExists, qui ne prend qu un chemin : « chemin --no-attach »
# n existe pas, et l unite est alors sautee sans jamais demarrer.
ExecStart=/bin/bash /opt/GSM/osmo-operator/start.sh --no-attach $OSMO_BANC_ARGS
ExecStop=/bin/bash /opt/GSM/osmo-operator/start.sh --stop
# LE COEUR ET LE TABLEAU DE BORD SUIVENT LE BANC. osmo-hlr et osmo-egprs-web
# sont des unites a part, demarrees l une par run.sh, l autre au boot : elles
# survivaient a l arret du banc. On se retrouvait avec un HLR qui garde ses
# abonnes et un tableau de bord qui affiche un banc qui n existe plus.
# `-` en tete : une unite deja arretee, ou absente, n a pas a faire echouer
# l arret.
# [2026-09-04] osmo-egprs-web n est plus nomme ici : il porte desormais
# PartOf=osmo-banc.service et systemd l arrete tout seul, y compris sur un
# `systemctl restart osmo-banc` que cette ligne ne couvrait pas. Un
# `systemctl stop` lance depuis l ExecStop de l unite qu on arrete se met en
# plus en file derriere son propre job : autant le laisser a systemd.
ExecStop=-/usr/bin/systemctl stop osmo-hlr.service
# L ENCART DU BUREAU SUIT LE BANC. osmo-fft-snap.py lit /psd et le mobile.log
# du run : un banc neuf, ce sont de nouveaux chemins sous /run/user/0 et un
# nouveau journal. On le relance donc avec le banc, sans jamais le rendre
# obligatoire - `-` en tete et --no-block : l encart est un ornement, il ne doit
# ni faire echouer le demarrage ni le retarder. try-restart ne fait rien si
# l unite n est pas la (image sans bureau).
ExecStartPost=-/usr/bin/systemctl --no-block try-restart osmo-fft-snap.service
# [2026-09-25] Plus de « image neuve a chaque banc neuf » : osmo-wallpaper
# n est plus relance ici, le fond reste l illustration brute
# (/usr/share/backgrounds/gsm-lab-wallpaper.jpg).
# QEMU + Calypso + rattachement des mobiles : plusieurs minutes sur une VM.
TimeoutStartSec=900
# [2026-09-04] 180 s -> 30 s. A l arret de la machine, systemd attend ce delai
# AVANT de tuer ce qui reste : trois minutes d ecran noir par unite, et il y en
# a deux. La pile s arrete en quelques secondes ; passe 30 s, elle est bloquee
# et un SIGKILL est la bonne reponse - pas une attente de plus.
TimeoutStopSec=30
# A l'arret, ce que start.sh --stop n'a pas demonte est termine par
# systemd (SIGTERM puis SIGKILL sur le cgroup) : pas de reste de run precedent.
KillMode=control-group
StandardOutput=journal
StandardError=journal
SyslogIdentifier=osmo-banc

# ── REPRENDRE LA MAIN ─────────────────────────────────────────────────────
#   tmux attach -t calypso        (root ; Ctrl-b d pour detacher)
#   journalctl -u osmo-banc -f    (la sortie de start.sh / run.sh)
# Le socket tmux est celui de root par defaut (/tmp/tmux-0/default) : PrivateTmp
# n'est PAS active ici, sinon la session serait invisible hors du service.
```

```
[Install]
WantedBy=multi-user.target
```

3.361 /opt/GSM/osmo-operator/services/osmo-bts-trx.service

1329 octets, 35 lignes → 35 lignes

```
[Unit]
Description=Osmocom GSM Base Transceiver Station (BTS)
# S'assure que le réseau et l'interface de bouclage sont prêts
# [2026-09-04] PLUS D ATTENTE RESEAU AU BOOT. network-online.target etait tire
# ici ; il coute jusqu a 2 min de demarrage (systemd-networkd-wait-online,
# mesure : systemd-analyze blame) sur un banc dont les interfaces - apn0, les
# tun/veth du coeur paquet - ne sont jamais "en ligne" au sens de cette unite,
# et qui demarre souvent sans cable. network.target suffit : les interfaces
# sont configurees, la route par defaut peut arriver apres.
After=network.target
# Si tu utilises osmo-bsc sur la même machine, décommente la ligne suivante
# After=osmo-bsc.service

[Service]
Type=simple
# Adapte l'utilisateur (ex: root ou ton utilisateur actuel 'nirvana')
User=root
Group=root

# Chemin vers l'exécutable (vérifie avec 'which osmo-bts-trx')
# -c spécifie le fichier de configuration
# -r 1 permet d'activer la priorité temps réel si le noyau le permet
ExecStart=/usr/bin/osmo-bts-trx -c /etc/osmocom/osmo-bts-trx.cfg -r 1

# Redémarrage automatique en cas de crash (utile pour la stabilité radio)
Restart=always
RestartSec=5

# Limites de ressources pour éviter les "hangs" système
LimitNOFILE=65535
CPUSchedulingPolicy=rr
CPUSchedulingPriority=1

[Install]
WantedBy=multi-user.target
```

3.362 /opt/GSM/osmo-operator/services/osmo-dsp-rom-max.service

846 octets, 22 lignes → 22 lignes

```
[Unit]
Description=osmo-operator - workflow « la ROM fait le maximum » : profil ROM max, banc DSP, echelle de tests,
 ↪  bilan ROM/hote
Documentation=file:/opt/GSM/osmo-operator/scripts/dsp-rom-max.sh
After=network.target
ConditionPathExists=/opt/GSM/osmo-operator/scripts/dsp-rom-max.sh

[Service]
# A la demande (systemctl start) : une campagne de tests n a rien a faire au
# boot. Le profil ROM max, lui, vit dans /etc/default/osmo-banc et s applique a
# chaque demarrage d osmo-banc.
Type=oneshot
WorkingDirectory=/opt/GSM/osmo-operator
Environment=HOME=/root NO_COLOR=1
```

```
EnvironmentFile=-/etc/default/osmo-dsp-rom-max
ExecStart=/bin/bash /opt/GSM/osmo-operator/scripts/dsp-rom-max.sh run $DSP_ROM_MAX_ARGS
TimeoutStartSec=3600
StandardOutput=journal
StandardError=journal
SyslogIdentifier=osmo-dsp-rom-max

[Install]
WantedBy=multi-user.target
```

3.363 /opt/GSM/osmo-operator/services/osmo-egprs-dsp.service

1317 octets, 31 lignes → 31 lignes

```
[Unit]
Description=osmo-operator - workflow EGPRS : cellule paquet virtuelle + modem GPRS, sur le banc DSP (ROM TI)
Documentation=file:/opt/GSM/osmo-operator/scripts/egprs-dsp.sh
# Le banc (coeur, BTS 0 sur le C54x, SGSN/GGSN) doit etre la : Requires=, pas
# seulement Wants= - sans lui, ce workflow n a rien a attacher.
Requires=osmo-banc.service
After=osmo-banc.service network.target
ConditionPathExists=/opt/GSM/osmo-operator/scripts/egprs-dsp.sh

[Service]
# Oneshot + RemainAfterExit : le script lance la cellule et le modem en unites
# transitoires (systemd-run : osmo-bts-virtual-gprs, osmo-pcu-gprs,
# osmo-virtphy-gprs, osmo-modem-gprs), puis sort. L unite reste « active
# (exited) » ; ExecStop les demonte.
Type=oneshot
RemainAfterExit=yes
WorkingDirectory=/opt/GSM/osmo-operator
Environment=HOME=/root NO_COLOR=1 EGPRS_BANC=1
# EGPRS_L1=auto|virtphy|dsp, EGPRS_IMSI..., OSMOCOM_CFG : a poser ici.
EnvironmentFile=-/etc/default/osmo-egprs-dsp
ExecStart=/bin/bash /opt/GSM/osmo-operator/scripts/egprs-dsp.sh start
ExecStop=/bin/bash /opt/GSM/osmo-operator/scripts/egprs-dsp.sh stop
# Le banc est deja monte (Requires) ; ici : BTS 2, PCU, attach, PDP.
TimeoutStartSec=600
TimeoutStopSec=30
StandardOutput=journal
StandardError=journal
SyslogIdentifier=osmo-egprs-dsp

[Install]
WantedBy=multi-user.target
```

3.364 /opt/GSM/osmo-operator/services/osmo-egprs-web-install.service

2109 octets, 42 lignes → 42 lignes

```
[Unit]
# Joue install-web-service.sh AU BOOT DU CONTENEUR, pas au build de l'image.
#
# POURQUOI PAS UN `RUN` DANS LE Dockerfile. Deux raisons, chacune suffisante :
#   1. le script finit par `systemctl daemon-reload` + `systemctl restart` ;
#      pendant `docker build` il n'y a pas de systemd (« System has not been
#      booted with systemd »), et le script est en `set -eu` : le build
#      echouerait ;
#   2. il genere un certificat TLS auto-signe. Une cle privee posee au build
#      serait IDENTIQUE pour quiconque tire l'image — n'importe qui pourrait se
#      faire passer pour la console. Le script le dit lui-meme dans son § 3.
#      On la genere donc une fois par conteneur, au premier boot.
```

```
#
# PAS DE `Before=osmo-egprs-web.service` — c'est deliberé. Le script se termine
# par `systemctl restart osmo-egprs-web` : ordonner cette unite AVANT le service
# creerait un cycle (systemctl attendrait un job qui attend cette unite).
# Consequence assumee et visible au journal : au tout premier boot le dashboard
# demarre une fois SANS TLS (« HTTPS non arme : certificat absent »), puis cette
# unite pose le certificat et le redemarre AVEC. Aux boots suivants le
# certificat est deja la (le script est idempotent, garde de 30 jours) et il n'y
# a pas de second demarrage.
#
# ⚠ Le certificat vit dans /opt/GSM/osmo-egprs-web/tls, DANS le conteneur : il ne
# survit pas a un `docker rm` (les conteneurs sont lances avec --rm). C'est
# voulu — il est regenere au boot suivant, avec les IP du moment dans le SAN.
Description=Installe/rafraichit le dashboard osmo-egprs-web (runtime, unit, certificat TLS)
Documentation=file:/opt/GSM/osmo-egprs-web/install-web-service.sh
After=network.target local-fs.target
ConditionPathIsDirectory=/opt/GSM/osmo-egprs-web

[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/bin/bash /opt/GSM/osmo-egprs-web/install-web-service.sh
# Le script peut passer par un `npm install` : large, mais borne.
TimeoutStartSec=600
StandardOutput=journal
StandardError=journal
SyslogIdentifier=osmo-egprs-web-install

[Install]
WantedBy=multi-user.target
```

3.365 /opt/GSM/osmo-operator/services/osmo-egprs-web.service

5597 octets, 90 lignes → 90 lignes

```
[Unit]
Description=osmo-nitb-for-calypso Operations Console (web dashboard)
Documentation=https://github.com/bbaranoff/osmo-egprs-web
After=network.target
# ── LE TABLEAU DE BORD S ARRETE AVEC LE BANC ──────────────────────────
# [2026-09-04] PartOf : systemd propage a CETTE unite les arrets et relances de
# osmo-banc.service. Sans lui, le dashboard restait debout sur :8080 devant une
# pile morte - l etat le plus trompeur qui soit, puisqu il se lit exactement
# comme "le banc tourne". PartOf et PAS After=osmo-banc.service : osmo-banc a
# deja Wants= et After= vers cette unite (le dashboard demarre AVANT le banc),
# ajouter l ordre inverse ferait un cycle que systemd casserait au hasard.
# Le geste direct (start-native.sh --stop, le bouton du bureau, start-multi.sh)
# est couvert de son cote, dans start-native.sh.
# [2026-10-22] PartOf RETIRE : le tableau de bord reste en ligne par defaut, banc arrete
# compris (demande explicite). L'arret avec le banc : CALYPSO_DASHBOARD_SUIT_BANC=1
# (startup_scripts/10-identite-noeud.sh) ; pour systemd, remettre PartOf=osmo-banc.service.

[Service]
Type=simple
WorkingDirectory=/opt/GSM/osmo-egprs-web
# Le TLS prend le 80 (demande explicite du 12/08). Le clair descend sur 8080 :
# les deux listeners ne peuvent pas partager un port, et supprimer le clair
# priverait `curl http://localhost` de tout point d'entree de diagnostic.
Environment=HTTP_PORT=8080
Environment=OSMO_NATIVE=1
Environment=OSMO_OP_IDS=1
# ⚠ NE PAS RETIRER : interface de capture du dashboard (GSMTAP/SCTP). La copie
# de ce unit qui vit dans le depot osmo-egprs-web l'avait perdue ; comme
# install-web-service.sh reinstalle le unit depuis CE fichier (le Dockerfile le
```

```
# recopie dans /opt/GSM/osmo-egprs-web/, cf. Dockerfile.run), la source unique est
# ici. Sans CAP_IFACE la capture se rabat sur un defaut et l'onglet trafic se
# tait — panne silencieuse, aucun message d'erreur.
Environment=CAP_IFACE=any
Environment=PULSE_SERVER=unix:/var/run/pulse/native
Environment=AUDIO_SOURCE=gsm_audio.monitor
# --- TLS (12/08/2026) ------------------------------------------------------
# server.js:1161-1183 n'arme le listener HTTPS que si CES DEUX fichiers existent ;
# sinon il journalise « HTTPS non arme : certificat absent » et reste en clair.
# Ils sont poses par install-web-service.sh (auto-signe, jamais dans l'image :
# une cle privee dans une image serait partagee par tous ceux qui la tirent).
# Le meme gestionnaire de requetes et la MEME instance WebSocket servent les deux
# portes -- on ajoute une entree, on ne deplace rien : http:// continue de marcher.
# ⚠ TLS sur le 80, pas sur le 443 : l'URL doit donc s'ecrire `https://<ip>:80`
# explicitement (un navigateur qui lit `https://<ip>` irait sur 443, ferme), et
# `http://<ip>` ne repond plus en clair — c'est du TLS qui ecoute la.
Environment=HTTPS_PORT=80
# ⚠ /etc/osmo-web-tls, PAS /opt/GSM/osmo-egprs-web/tls. C'est LA que
# install-web-service.sh ecrit le certificat (son § 3, variable TLS_DIR), et
# c'est aussi le defaut de server.js:1805. Ce unit nommait un troisieme
# chemin, que personne n'ecrit jamais : le certificat existait, le listener
# HTTPS ne s'armait pas, le journal disait « HTTPS non arme : certificat
# absent » en citant un repertoire absent, et le dashboard restait en clair.
# Consequence directe et sans rapport apparent : depuis une IP, le bouton
# micro etait REFUSE PAR LE NAVIGATEUR - navigator.mediaDevices n'existe pas
# hors contexte securise. Le diagnostic partait invariablement sur Firefox.
Environment=TLS_CERT=/etc/osmo-web-tls/cert.pem
Environment=TLS_KEY=/etc/osmo-web-tls/key.pem
# ⚠ /usr/bin/node, PAS /usr/local/bin/node. Dans le conteneur, node vient de
# install-web-service.sh, qui deballe /opt/node et pose les liens dans
# /usr/local/bin. Dans l'ISO il vient du paquet nodesource, donc de
# /usr/bin/node, et build-iso.sh SUPPRIME les liens /usr/local/bin/node,npm,npx
# parce qu'ils y pendent dans le vide. Cette unite nommait l'autre chemin :
# apres installation sur disque, le service sortait en 203/EXEC a chaque
# demarrage, osmo-egprs-web-install echouait dans la foulee, et le dashboard
# n'avait ni TLS ni micro. /usr/bin/node existe des DEUX cotes (le paquet
# nodesource s'y installe aussi) : c'est le seul chemin qui vaut partout.
# ── LE CHEMIN DE NODE N EST PAS LE MEME PARTOUT ──────────────────────────
# [2026-08-31] Cette ligne portait /usr/bin/node en dur. C est vrai sur l HOTE,
# faux dans les conteneurs osmo-operator-*, ou node vit en /usr/local/bin/node
# (lien vers /opt/node/bin/node, celui qu install-web-service.sh depose quand
# node manque). systemd sortait alors en
#     status=203/EXEC
# et relançait en boucle (Restart=on-failure). 203/EXEC ne dit pas « fichier
# introuvable » mais « je n ai pas pu executer » : on cherche des droits ou du
# SELinux, jamais un chemin.
#
# /usr/bin/env resout node dans le PATH, donc a l un OU l autre emplacement.
# Le PATH est pose explicitement juste au-dessus : celui de systemd ne contient
# pas toujours /usr/local/bin, et un env sans PATH sur mesure retomberait
# exactement sur le probleme qu on corrige.
Environment=PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
ExecStart=/usr/bin/env node /opt/GSM/osmo-egprs-web/server.js --verbose
Restart=on-failure
RestartSec=3
StandardOutput=journal
StandardError=journal
SyslogIdentifier=osmo-egprs-web

[Install]
WantedBy=multi-user.target
```

3.366 /opt/GSM/osmo-operator/services/osmo-lte.service

2372 octets, 48 lignes → 48 lignes

```
[Unit]
Description=osmo-operator - la 4G du banc (Open5GS + srsENB + srsUE ZeroMQ, via osmo-lte)
Documentation=file:/opt/GSM/osmo-operator/tools/osmo-lte.sh
# [2026-09-08] LA 4G EN SERVICE, LANCEE PAR L ICONE. Jusqu ici la 4G ne se
# lancait qu a la main (« sudo osmo-lte start ») ; la voici en unite, comme
# osmo-banc : un etat, un journal, un arret propre, et une icone du bureau
# (data/desktop/osmo-lte.desktop -> « osmo-lte toggle ») qui la bascule.
# ExecStart = osmo-lte start : le coeur Open5GS (osmo-epc, abonnes restaures
# s il le faut), puis srsENB, puis srsUE dans le netns ue1 - le PPP du
# telephone pmOS y aboutit (osmo-pmos-setup).
# Le coeur a besoin de MongoDB (abonnes) : on le tire, sans l exiger.
After=network.target mongod.service
Wants=mongod.service
ConditionPathExists=/opt/GSM/osmo-operator/tools/osmo-lte.sh
# Pas de srsRAN (hub interstp, arm64) : l unite est sautee, pas en echec.
ConditionPathExists=/usr/local/bin/srsenb
# POSEE, PAS ACTIVEE (meme regle que osmo-banc) :
#   systemctl start osmo-lte            une session (ou l icone)
#   systemctl enable --now osmo-lte     a chaque boot
# Au build : OSMO_ISO_LTE=1 active l unite dans l image.

[Service]
# oneshot + RemainAfterExit : osmo-lte start lance eNB/UE en setsid et les
# demons Open5GS en -D, puis SORT. En Type=simple systemd prendrait cette
# sortie pour la fin du service et tuerait le cgroup - toute la 4G.
Type=oneshot
RemainAfterExit=yes
WorkingDirectory=/opt/GSM/osmo-operator
Environment=HOME=/root USER=root LOGNAME=root
Environment=PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
# Reglages durables (OSMO_LTE_SRATE, OSMO_LTE_NETNS, SRS_WAN...) :
EnvironmentFile=-/etc/default/osmo-lte
ExecStart=/bin/bash /opt/GSM/osmo-operator/tools/osmo-lte.sh start
# L arret eteint TOUT : eNB et UE (osmo-lte stop), puis le coeur (osmo-epc
# stop) - a la main, « osmo-lte stop » laisse le coeur, pas le service.
ExecStop=-/bin/bash /opt/GSM/osmo-operator/tools/osmo-lte.sh stop
ExecStop=-/bin/bash /opt/GSM/osmo-operator/tools/osmo-epc.sh stop
# L attach de l UE est attendu 30 s par osmo-lte start ; le coeur avant.
TimeoutStartSec=120
TimeoutStopSec=30
# Ce que les deux stop n ont pas demonte est termine par systemd.
KillMode=control-group
StandardOutput=journal
StandardError=journal
SyslogIdentifier=osmo-lte

[Install]
WantedBy=multi-user.target
```

3.367 /opt/GSM/osmo-operator/services/osmo-multi.service

4310 octets, 74 lignes → 74 lignes

```
[Unit]
Description=osmo-operator - banc multi-operateur (start-multi.sh : natif + conteneurs + inter-STP)
Documentation=file:/opt/GSM/osmo-operator/README.md
# Le multi-operateur RACCORDE le natif, il ne le lance pas : c'est osmo-banc
# qui le tient.
# [2026-09-05] Requires= -> Wants=. C'ETAIT UN SUICIDE. start-multi.sh applique
```

```
# « un clic = un banc neuf » : il fait `systemctl stop osmo-banc` a chaque
# lancement (tout_arreter), puis le relance (lancer_natif_si_absent). Or
# Requires= propage l'arret EXPLICITE d'une dependance a l'unite qui en depend :
# ce stop du natif arretait osmo-multi lui-meme, en plein ExecStart -
# `code=killed, status=15/TERM`, l'unite en echec sans une ligne d'erreur utile.
# Wants= tire le natif au boot dans le meme ordre (After= plus bas) mais NE
# propage PLUS l'arret : start-multi.sh peut recycler osmo-banc sans se tuer. La
# garantie « pas de natif = un ASP sur trois » reste tenue par start-multi.sh,
# qui relance osmo-banc si besoin - ce n'est pas a systemd de la porter.
Wants=osmo-banc.service
# [2026-09-04] PLUS D ATTENTE RESEAU AU BOOT. network-online.target etait tire
# ici ; il coute jusqu a 2 min de demarrage (systemd-networkd-wait-online,
# mesure : systemd-analyze blame) sur un banc dont les interfaces - apn0, les
# tun/veth du coeur paquet - ne sont jamais "en ligne" au sens de cette unite,
# et qui demarre souvent sans cable. network.target suffit : les interfaces
# sont configurees, la route par defaut peut arriver apres.
After=osmo-banc.service docker.service network.target
Wants=docker.service
ConditionPathExists=/opt/GSM/osmo-operator/launch/start-multi.sh
# ── ET LA TOPOLOGIE DOIT EXISTER ──────────────────────────────────────
# [2026-09-04] Sur une ISO fraiche, un demarrage de cette unite serait un echec :
# ni docker ni l image osmocom-run ne sont dans le squashfs - ils arrivent avec
# le supplement SS7 (addition.sh), qui est aussi ce qui ecrit ce fichier.
# Une condition non remplie n est PAS un echec pour systemd : l unite est
# sautee, en silence, et `systemctl --failed` reste vide (ce que le Conky
# affiche desormais nommement). La condition vaut pour le boot comme pour un
# `systemctl start` a la main : tant que la topologie n existe pas, il n y a
# rien a monter.
ConditionPathExists=/etc/osmocom/osmo-multi.conf
# [2026-09-05] NI CELLE-CI NI osmo-banc NE SONT ACTIVEES PAR DEFAUT. Les deux
# sont posees par l'ISO (iso_modules/82-services.sh) et par addition.sh, et
# c'est l'operateur qui demarre son banc - pas le boot, sur une configuration
# qu'il n'a pas encore choisie.
#   systemctl start osmo-multi          pour une session
#   systemctl enable --now osmo-multi   pour le demarrer a chaque boot
# Au build : OSMO_ISO_MULTI=1 (ou --multi) active l'unite dans l'image.

[Service]
Type=oneshot
RemainAfterExit=yes
WorkingDirectory=/opt/GSM/osmo-operator
Environment=HOME=/root USER=root LOGNAME=root
Environment=VIRTUAL_ENV=/root/.env
Environment=PATH=/root/.env/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
# OSMO_NO_TERM : start-multi.sh ne doit ni ouvrir un emulateur de terminal ni
# retenir une fenetre. OSMO_BANC_SERVICE : le natif est une unite systemd,
# start-multi.sh le (re)lance par systemctl et non par le raccourci du bureau.
Environment=NO_COLOR=1 OSMO_NO_TERM=1 OSMO_BANC_SERVICE=osmo-banc.service
EnvironmentFile=-/etc/default/osmo-multi
ExecStart=/bin/bash /opt/GSM/osmo-operator/launch/start-multi.sh $OSMO_MULTI_ARGS
# --stop n'arrete que les conteneurs et le hub : le natif reste a osmo-banc.
ExecStop=/bin/bash /opt/GSM/osmo-operator/launch/start-multi.sh --stop
TimeoutStartSec=1200
# [2026-09-04] 180 s -> 30 s. A l arret de la machine, systemd attend ce delai
# AVANT de tuer ce qui reste : trois minutes d ecran noir par unite, et il y en
# a deux. La pile s arrete en quelques secondes ; passe 30 s, elle est bloquee
# et un SIGKILL est la bonne reponse - pas une attente de plus.
TimeoutStopSec=30
KillMode=control-group
StandardOutput=journal
StandardError=journal
SyslogIdentifier=osmo-multi

# Reprendre la main : tmux attach -t calypso (natif) ; docker logs -f osmo-op2
# (conteneurs) ; journalctl -u osmo-multi -f.
```

```
[Install]
WantedBy=multi-user.target
```

3.368 /opt/GSM/osmo-operator/services/osmo-reglages.service

704 octets, 20 lignes → 20 lignes

```
[Unit]
Description=Page web de reglage de la pile Osmocom (http://127.0.0.1:8090/)
After=network.target
# [2026-10-22] En ligne par defaut, independante du banc : globals.conf et les
# fichiers de config se reglent banc arrete ; les onglets Demons/Etat disent
# simplement qu'aucun demon ne repond. 127.0.0.1 seulement : la console VTY
# de la page execute tout (voir tools/osmo-reglages.py).

[Service]
Type=simple
WorkingDirectory=/opt/GSM/osmo-operator
ExecStart=/usr/bin/python3 /opt/GSM/osmo-operator/tools/osmo-reglages.py --port 8090 --bind 127.0.0.1
Restart=on-failure
RestartSec=3
StandardOutput=journal
StandardError=journal
SyslogIdentifier=osmo-reglages

[Install]
WantedBy=multi-user.target
```

3.369 /opt/GSM/osmo-operator/services/osmo-supplements.service

851 octets, 21 lignes → 21 lignes

```
[Unit]
Description=osmo-operator - les supplements choisis dans l installeur (premier demarrage)
Documentation=file:/opt/GSM/osmo-operator/tools/osmo-supplements-firstboot.sh
# [2026-10-24] Activee par le job Calamares osmosupplements quand une case de la page
# « Supplements » est cochee ; se desactive d elle-meme une fois les supplements poses.
Wants=network-online.target
After=network-online.target
ConditionPathExists=/etc/osmo-supplements

[Service]
# simple, PAS oneshot : un oneshot de multi-user.target retiendrait le demarrage (et
# l ecran de connexion) jusqu a la fin - une heure pour l image operateur.
Type=simple
Environment=HOME=/root
ExecStart=/bin/bash /opt/GSM/osmo-operator/tools/osmo-supplements-firstboot.sh
StandardOutput=journal
StandardError=journal
SyslogIdentifier=osmo-supplements

[Install]
WantedBy=multi-user.target
```

3.370 /opt/GSM/osmo-operator/startup_scripts/01-defauts-options.sh

34986 octets, 543 lignes → 543 lignes

```
# 01-defauts-options.sh - partie de start.sh (lignes 41-542 de start-native.sh d'origine)
# Sourcee par start.sh dans le shell principal : meme ordre, memes variables.

# ── LE NOMBRE D'ABONNES NE PASSE PAS PAR ~/.bashrc ──────────────────────
# [2026-09-04] /etc/osmocom/coeur.env (N_MS, ecrit par build-iso.sh) n'etait
# charge que par /root/.bashrc - donc SEULEMENT dans un shell interactif de
# root. Depuis l'icone (launch.sh -> pkexec), depuis une unite systemd ou
# depuis start-multi.sh, personne ne le lisait : run_modules/21-abonnes-hlr.sh
# retombait sur N_MS=1, le HLR n'avait que 100101, et le second mobile se
# faisait creer a la volee par « subscriber-create-on-demand 5 » avec un
# MSISDN aleatoire a cinq chiffres (31804) et SANS Ki - donc injoignable et
# rejete a l'authentification. On le charge ici, une fois pour toutes ; le
# « := » du fichier laisse gagner un N_MS=3 pose dans l'environnement.
if [ -f "${OSMOCOM_CFG:-/etc/osmocom}/coeur.env" ]; then
    set -a; . "${OSMOCOM_CFG:-/etc/osmocom}/coeur.env"; set +a
fi
# --- options ----------------------------------------------------------------
DRY=0 VERBOSE=0 ACTION=start PROFILE="${CALYPSO_PROFILE:-faketrx-qemu}" FORCE=0
# [2026-10-14] SAP desactive par defaut : la SIM du mobile est la SIM de test.
# --sap : la SIM est SERVIE au mobile (sim sap) par le serveur SAP de softSIM,
# lance ici avant la pile. SAP_TYPE : pcsc = la carte du lecteur PC/SC (HID
# Omnikey, sysmoISIM programmee par tools/osmo-sim.sh), sim = la SIM logicielle
# sim-op1.xml (IMSI du MS#1, COMP128v1, Ki du HLR), auto = pcsc si une carte
# est dans un lecteur, sinon sim. SAP_SOCK : le socket (celui de MS#1).
: "${SAP_MODE:=0}"
: "${SAP_TYPE:=auto}"
: "${SAP_SOCK:=/tmp/osmocom_sap}"
# 1 des que l'operateur a nomme un profil (--profile, ou le mode en positionnel).
# Sert a --dsp, qui choisit `qemu` SEULEMENT si personne n'a choisi avant lui.
PROFILE_CHOISI=0
[ -n "${CALYPSO_PROFILE:-}" ] && PROFILE_CHOISI=1
# WAN : jamais par defaut. --wan (ou WAN_AUTO=1 dans /etc/osmo-wan.conf, ce que
# pose une ISO construite avec --wan) le monte avant de passer la main a run.sh.
WAN_MESH=0
# Ce que la table retiendra (WAN_AUTO_SAVE, network/wan-nodes.sh) : 1 seulement
# si c'est WAN_AUTO=1 qui monte le WAN ; un --wan sur la ligne vaut pour CE
# lancement et laisse WAN_AUTO=0, pour que le suivant demarre sans WAN.
WAN_AUTO_KEEP=0
# Inter-STP (hub SS7) lance en standalone a cote du WAN : auto = seulement si le
# hub n'existe pas encore sur le reseau (ni adresse locale, ni reponse ping).
# --interstp force, --no-interstp passe. Un seul hub pour tout le WAN : sur le
# noeud qui ne le porte pas, le hub distant est reconnu et laisse en place.
INTERSTP=auto
# --wan FICHIER : fiches de noeuds a importer (network/node-conf.sh).
WAN_IMPORT_FILES=()
# --regen-conf : la table WAN est reconstruite DE ZERO. La table en place
# (/etc/osmo-wan.conf) survit aux relances et garde les noeuds qu'on y a vus un
# jour ; un pair eteint y laisse une route qui passe PROHIB au STP. On met la
# table de cote (sauvegarde horodatee) et on repart de ce que la ligne donne
# (--wan-nodes), des fiches (--wan FICHIER) ou de la question interactive.
REGEN_WAN_CONF=0
# --payload-ism : le TEXTE des SMS part par radio ISM, le reste (signalisation,
# SS7, SIP) par IP comme --wan. PAYLOAD_ISM_LINK = backend si donne (--payload-ism=serial).
PAYLOAD_ISM=0
PAYLOAD_ISM_LINK=""
GEN_CONF_OUT=""
# --virtualbox : monte le segment et les VM avant le WAN. Refuse si l'on tourne
# DANS une VM - c'est l'hote qui pilote VirtualBox (le script le verifie).
VBOX_INTERCO=0
VBOX_NODES=""
VBOX_HOST_NODE=1
# --node : le numero de CE noeud, choisi au lancement et pas a la construction.
# C'est ce qui permet une ISO unique pour les neuf noeuds.
NODE_ID=""
NODE_OP=1
HUB_IP=""
```

```
# Les gabarits ne sont PAS regeneres par defaut : run.sh le ferait au demarrage
# et effacerait l'identite SS7 posee juste avant. --regen retablit l'ancien
# comportement pour qui veut repartir des gabarits.
REGEN_GABARITS=0
# --menu : demande les choix au lieu de les deviner. Voir le bloc « Menu » plus bas.
MENU_MODE=0
# ── LA COUCHE 1 PAR DEFAUT : LE DSP ──────────────────────────────────────
# [2026-10-01] Le banc demarre en DSP (c54x_exe, mask-ROM TI) sans qu on le
# demande : c est le banc qu on veut voir au premier clic, en service comme en
# terminal, en standalone comme sous osmo-multi (qui lance le natif par
# osmo-banc.service, donc par ce defaut). --grgsm remet la couche 1 gr-gsm.
# L environnement gagne (DSP_MODE=0 ./start-native.sh), comme pour le reste.
# Si c54x_exe n est pas la, le banc retombe sur gr-gsm et le dit (plus bas).
: "${DSP_MODE:=1}"
usage() {
    cat <<'USAGE'
Usage : ./start-native.sh [options] [mode]
  Modes (profils) :
    faketrx-qemu   coeur + BTS#0 QEMU + BTS#1 faketrx (defaut)
    faketrx        coeur + fake_trx + trxcon + mobile
    qemu           pipeline Calypso QEMU seul
    noproc         coeur seul (no-process)
    core           alias noproc
    hybrid         alias faketrx-qemu
  Options :
    --docker            banc en conteneurs : passe la main a
                        launch/start-docker.sh avec les autres arguments
    --dsp               (defaut) banc DSP C54x : le TMS320C54x tourne hors QEMU
                        (/opt/GSM/c54x_exe, mask-ROM TI) et decode lui-meme
                        FCCH/SCH/BCCH ; aucun decodeur gr-gsm n'est lance.
                        Le mobile campe (SI1-4, lai=001-01-1) et fait sa mise
                        a jour de localisation. La pile monte comme en mode
                        grgsm (meme profil hybride, coeur, BTS, side-car) ;
                        seuls les modules qemu,pty,osmocon,l2 sont retires et
                        repris par c54x_exe/run.sh. BANC_DSP=none pour n'avoir
                        que la pile.
    --grgsm             couche 1 gr-gsm dans QEMU (qosmo) + pont grgsm_exe
                        (l ancien defaut ; DSP_MODE=0 dans l environnement vaut pareil)
    --bts-classic       [2026-10-21] (alias : --bts-classique) la BTS CLASSIQUE : osmo-bts-trx parle a
↪   pont/pont_dsp.py
                        (TRXD 5700-5702), sans Calypso TRX dans QEMU ; le mobile emule
                        (QEMU + c54x_exe, MODE=dsp) est branche sur le pont. C'est le montage
                        par defaut de --dsp, nomme ; OSMO_TRX=0 force (meme si l environnement
                        l exporte). Exclusif avec --bts / --ts.
                            ./start-native.sh --bts-classic [--ms]           mobile osmocom-bb
                            ./start-native.sh --bts-classic --ms-magnetite   le C139 Magnetite
    --bts               [2026-10-20] (alias : --osmo-trx) le Calypso de QEMU est le
                        TRANSCEIVER de la BTS#0 : firmware trx.highram.elf
                        (osmocom-bb-transceiver, appli « trx ») + appli hote
                        `transceiver` (L1CTL -> TRXD 5700-5702) a la place de
                        mobile et du pont ; osmo-bts-trx s'y branche comme sur le
                        pont. Implique --dsp (c54x_exe/run.sh, OSMO_TRX=1). Par
                        defaut UN Calypso sert TS0 et TS1 (OSMO_TRX_TCH=1, masques
                        0x03) ; voir --ts.
    --ts N              [2026-10-20] intervalles de temps du TRX Calypso (implique --bts) :
                            --ts 1   TS0 seul (CCCH + SDCCH/4, pas d'appel) : OSMO_TRX_TCH=0
                            --ts 2   TS0 + TS1, UN CALYPSO PAR INTERVALLE : le TS1 (TCH/F,
                                     les appels) est servi par un deuxieme telephone TRX
                                     (c54x_exe/run.sh OSMO_TRX_TS1=1, instance 3,
                                     transceiver -2 -L /tmp/osmocom_l2_3) ; le premier
                                     garde le TS0 seul (OSMO_TRX_TCH=0). [21/10] PAS D'APPEL
                                     dans ce mode : le fn du 2e telephone n'est pas aligne
                                     (TCH descendant a ~80 bits faux, ASSIGNMENT FAILURE).
                            --ts 2/1 TS0 + TS1 par UN SEUL Calypso (OSMO_TRX_TCH=1, masques
                                     0x03, OSMO_TRX_TS1=0) : le montage qui a passe un
                                     appel audible le 20/10 21:08.
```

```
                    Sans --ts : comme --ts 2/1 (21/10 ; du 20/10 23:10 au 21/10 c'etait --ts 2).
--ms                [2026-10-20] le telephone emule est un MOBILE. Seul (ou sans
                    option) : le montage habituel, mobile + pont vers osmo-bts-trx.
                    AVEC --bts : un DEUXIEME Calypso (c54x_exe/run.sh INSTANCE=2,
                    alias banc-mobile) campe sur le premier par l'air (pont/air.py) ;
                    il est lance quand le firmware trx est en mode BTS.
                        ./start-native.sh --bts          la radio de la BTS seule
                        ./start-native.sh --bts --ms     la BTS et le mobile, par l'air
--ms-magnetite      [2026-10-21] comme --ms, mais le mobile est le VRAI firmware TI :
                    FreeCalypso Magnetite (cible c139, config hybrid-ui-vo, compile
                    avec le TMS470 sous Wine ; doc/magnetite.md) en flash a la place
                    du layer1 osmocom-bb (c54x_exe/run.sh FIRMWARE_FLASH=...). Pile
                    L23 et interface dans le firmware : pas d'osmocon ni de mobile,
                    rvinterf decode les traces, ecran couleur du C139 dans
                    tools/osmo-c123-gtk.py --modele c139. MAGNETITE_FLASH=<image>
                    (defaut /opt/GSM/firmware/magnetite/c139-flash.img).
                        ./start-native.sh --bts --ms-magnetite   la BTS et le C139 Magnetite, par l'air
                        ./start-native.sh --ms-magnetite         le C139 Magnetite seul, par le pont
--vocoder-rom       [2026-10-22] DEFAUT. Le codec FR de la mask-ROM du DSP vocode (port voix
                    SP0 modelise, CALYPSO_VSP=1, pont/vsp_audio.py) : BANC_VOCODEUR=rom.
                    Le C54x emule doit alors tenir le temps reel AVEC le codec.
--gapk              [2026-10-22] Le vocodeur tourne sur l'hote (osmo-gapk dans le mobile L23) :
                    BANC_VOCODEUR=hote, CALYPSO_VSP=0, TCH montant code par l'hote
                    (PONT_UL_RAW_TCH=0). Le DSP ne fait plus que le canal : il tient le temps
                    reel. Mobile osmocom seulement : le firmware TI (--ms-magnetite) vocode
                    toujours dans la ROM.
--sap, --softsim    SIM via SAP : serveur softSIM (Ruby, /opt/GSM/softsim) sur
                    /tmp/osmocom_sap, lance avant la pile ; le MS#1 passe en
                    « sim sap ». Sert la carte du lecteur PC/SC (HID Omnikey,
                    sysmoISIM : tools/osmo-sim.sh programmer) si une carte y
                    est, sinon la SIM logicielle sim-op1.xml (IMSI du MS#1).
--sap=pcsc          SIM via SAP : la carte du lecteur, obligatoirement
                    (--omnikey, --sim-reader : pareil)
--sap=sim           SIM via SAP : la SIM logicielle, meme avec une carte
--no-sap            pas de SIM via SAP (la SIM de test du mobile), par defaut
--launcher <bin>    lanceur C de QEMU a utiliser (defaut : /usr/local/bin/qosmo).
                    Il remplace l'appel direct a qemu-system-arm : memes
                    defauts, sockets et pty publies (<RUN_DIR>/modem.pty),
                    osmocon direct. Se compile dans
                    /opt/GSM/qosmo/tools/qosmo-launch (make ALIAS=qosmo install).
--list              affiche le plan (delegue a run.sh --list)
--dry-run           deroule sans effet de bord
--profile <nom>     force le profil
--stop              arrete la pile (delegue a run.sh --stop)
--status            interroge l'etat (delegue a run.sh --status)
--force             relance meme les modules deja demarres
--verbose           montre la sortie des modules
--assembly-logs     trace asm de l'ARM emule (= ASSEMBLY_LOGS=1) : QEMU
                    -d in_asm -D /var/tmp/c54x-pont/qemu-asm.log (lien
                    /tmp/c54x-pont/qemu-asm.log ; ASSEMBLY_LOGS_DIR change le
                    dossier, sur le disque : /tmp est un tmpfs de 4,7 Go),
                    chaque bloc une fois, a sa traduction, PLUS l'anneau des
                    blocs executes (plugin TCG qosmo/tools/qosmo-tbring) :
                    64 Gio creux dans le meme dossier (qemu-tbring), le PC de
                    chaque bloc execute, chaines compris, relu a chaud par
                    tbring-dump -n 500 -e <elf> -a qemu-asm.log. Rien de texte
                    dans la boucle : l'ARM garde son rythme. ASSEMBLY_LOGS_RING=0
                    coupe l'anneau. ASSEMBLY_LOGS_FLAGS
                    change les -d, ASSEMBLY_LOGS_FILTRE pose un -dfilter (ex.
                    0x00820000+0x40000). ASSEMBLY_LOGS_FLAGS=in_asm,exec,nochain
                    journalise aussi chaque execution en texte (150 Mo/s,
                    20 Go en 130 s ; QEMU_LOG_BUFFERED=4 Mio vide a chaque
                    arret de la VM) ; avant la correction de cpu_idle_park
                    du 2026-10-19 (qosmo calypso_trx.c) :
                    l'ARM passe de 0,4 a 62 ms par trame, le DSP en pas-a-pas
                    suit, l'horloge du pont (pont/trx.py) descend jusqu'a
```

```
                    PONT_HORLOGE_MAX x le temps reel (16 avec ASSEMBLY_LOGS=1,
                    4 sinon) et les IND CLOCK attendent la reprise de la BTS
                    (osmo-bts-trx s'arrete au-dela de 50 trames de pause).
                    Les temporisateurs de mobile (T3210 20 s) et du reseau
                    (T3101) restent muraux : a 14 x, l'attach peut encore
                    buter dessus. Runs du 2026-10-19 02:06 et 02:20.
--icount[=SPEC]     QEMU avec -icount SPEC (defaut : auto ; = QEMU_ICOUNT),
                    en --dsp (c54x_exe) comme en --grgsm (run_modules/40-qemu).
--no-attach         ne s'attache pas a tmux a la fin (= CALYPSO_NO_ATTACH=1 ;
                    c'est ce que pose osmo-banc.service : personne devant)
--check-paths       verifie les dependances declarees
--wan               monte le WAN a N noeuds (1 a 9) AVANT run.sh et demande :
                      nombre de noeuds, IP de chaque noeud, indicatif de
                      chaque noeud, numero du noeud construit par ce lancement
--wan mynode1.conf [--wan mynode3.conf ...]
                    WAN a partir des FICHES des autres noeuds (ecrites par
                    leur run : /etc/osmocom/mynode<N>.conf). Ce noeud est
                    ajoute a la table avec son adresse locale ; aucune
                    question. Le maillage SIP + SMS suit comme avec --wan.
--gen-conf [FICHIER] ecrit la fiche de CE noeud (defaut
                    /etc/osmocom/mynode<N>.conf) SANS lancer le banc. Le
                    run l'ecrit de toute facon, juste avant run.sh.
--wan-nodes "1:IP:IND ..."   meme chose sans question (scriptable)
--wan-id N          numero du noeud local (sinon deduit des IP locales)
--wan-conf FICHIER  table a lire/ecrire (defaut /etc/osmo-wan.conf)
--regen-conf        reconstruit la table WAN DE ZERO (implique --wan) : la
                    table en place est sauvegardee puis effacee, on repart
                    de --wan-nodes / des fiches / de la question. A utiliser
                    quand un pair eteint laisse une route PROHIB au STP.
--payload-ism[=BACKEND]
                    le TEXTE des SMS entre noeuds part par une liaison RADIO
                    en bande ISM ; tout le reste (signalisation, SS7, SIP)
                    reste sur IP, comme --wan (a combiner avec --wan). Le
                    relais met un jeton dans le TCP et le texte sur la radio.
                    BACKEND = serial|exec|fifo|loopback. Sans BACKEND, lit la
                    section [payload] deja ecrite par network/setup-payload-ism.sh.
                    Peripherique/commandes par l'env : ISM_SERIAL, ISM_TX,
                    ISM_RX, ISM_TX_CMD, ISM_RX_CMD. Voir wiki/Payload-ISM.md.
--node N            numero de CE noeud, 1 a 9. DEDUIT AUTOMATIQUEMENT s'il
--menu              demande les choix au lieu de les deviner : profil,
                    noeud, operateur, inter-STP, WAN, regeneration des
                    configs. Les valeurs proposees sont
                    celles que le lanceur aurait prises tout seul, donc
                    valider sans rien changer equivaut a ne pas passer
                    l'option. Sans terminal, l'option est ignoree.
--regen             regenere les configs depuis les gabarits (par defaut on
                    CONSERVE celles deja en place : run.sh les ecraserait)
                    est omis : environnement, puis /etc/osmo-role, puis la
                    table WAN comparee aux adresses locales. Cette option
                    ne sert qu'a forcer. Reecrit son identite SS7
                    (point codes 1.<noeud><op>.<role>, routing contexts) et
                    pointe son ASP sur l'inter-STP. Une seule ISO suffit
                    alors pour les neuf noeuds - le numero se choisit ici.
--op N              operateur porte par ce noeud (defaut 1)
--hub-ip ADRESSE    inter-STP a joindre (defaut : selon docker ou VM ; natif : 192.168.1.49)
--interstp          lance le inter-STP en standalone sur ce noeud (force)
--no-interstp       ne lance pas le inter-STP (defaut : auto, avec --wan)
--air-mesh[=PORT]   maillage de BURSTS entre noeuds : le milieu radio
                    devient commun et un mobile peut entendre - et choisir -
                    la BTS d'un autre noeud. Les pairs viennent de la table
                    WAN, la portee de data/air-mesh.txt. Declare deux
                    mobiles de plus pour s'y brancher (voir pont/airmesh.py).
--virtualbox[=N]    WAN entre CETTE machine et N-1 VM VirtualBox (implique
                    --wan). A lancer depuis l'hote, pas depuis une VM.
--vbox-node N       numero de noeud porte par cette machine (defaut 1)
--mobile            client de couche 2 = mobile (defaut) -- pile L2/L3
```

```
                      complete derriere le firmware QEMU
    --ccch_scan       client de couche 2 = ccch_scan -- ecoute le CCCH
    --bcch_scan       client de couche 2 = bcch_scan -- releve les BCCH
    --cell_log        client de couche 2 = cell_log -- journal de cellules
                      Ces quatre-la sont EXCLUSIFS et ECRASENT le mobile de
                      QEMU : ils prennent sa socket L1CTL et sa VTY.
    -h, --help        cette aide

  Sans --wan : aucun WAN. Avec --wan le profil reste faketrx-qemu (hybride,
  1 faketrx + 1 QEMU Calypso), qui est deja le defaut de ce script.
  Composer <indicatif><numero> joint ce numero sur le noeud correspondant.
Toute variable CALYPSO_* passee en prefixe est transmise a run.sh / QEMU :
  CALYPSO_LANG=en ./start-native.sh       CALYPSO_NO_ATTACH=1 ./start-native.sh
USAGE
}
while [ $# -gt 0 ]; do
    case "$1" in
        --list)        ACTION=list ;;
        --dry-run)     DRY=1 ;;
        --profile)     if [ $# -lt 2 ] || [ -z "${2:-}" ]; then
                           printf '%s\n\n' "--profile attend un nom de profil" >&2
                           usage >&2; exit 2
                       fi
                       PROFILE="$2"; PROFILE_CHOISI=1; shift ;;
        --stop)        ACTION=stop ;;
        --status)      ACTION=status ;;
        --force)       FORCE=1 ;;
        --verbose)     VERBOSE=1 ;;
        --assembly-logs) ASSEMBLY_LOGS=1 ;;
        --icount)      QEMU_ICOUNT=auto; export QEMU_ICOUNT ;;
        --icount=*)    QEMU_ICOUNT="${1#*=}"; export QEMU_ICOUNT ;;
        # [2026-09-04] Equivalent de CALYPSO_NO_ATTACH=1, en option de ligne de
        # commande pour que l'unite systemd le dise sur son ExecStart. Avant, cette
        # option n'existait pas : posee sur ExecStart elle tombait dans « option
        # inconnue » (exit 2), et posee ailleurs dans l'unite elle cassait la ligne
        # qui la portait (ConditionPathExists=... --no-attach : chemin inexistant,
        # unite jamais lancee).
        --no-attach)   CALYPSO_NO_ATTACH=1; export CALYPSO_NO_ATTACH ;;
        --check-paths) ACTION=checkpaths ;;
        --wan)         WAN_MESH=1
                       # --wan FICHIER : la fiche d'un autre noeud (network/node-conf.sh).
                       if [ -n "${2:-}" ] && [ -f "$2" ]; then WAN_IMPORT_FILES+=("$2"); shift; fi ;;
        --gen-conf)    ACTION=genconf
                       if [ -n "${2:-}" ] && [ "${2#-}" = "$2" ]; then GEN_CONF_OUT="$2"; shift; fi ;;
        --gen-conf=*)  ACTION=genconf; GEN_CONF_OUT="${1#*=}" ;;
        # [2026-09-22] --dsp ne change NI de fork NI de profil : la pile monte
        # exactement comme en mode grgsm (coeur, BTS, side-car, tmux), et seuls
        # les quatre modules de la chaine Calypso (qemu, pty, osmocon, l2) sont
        # remis a /opt/GSM/c54x_exe, ou le C54x tourne hors QEMU sur la mask-ROM
        # TI. [2026-09-25] Les forks qosmo-dsp/qosmo-grgsm sont retires : un
        # seul arbre QEMU, /opt/GSM/qosmo (run.sh et run_modules y sont), et la
        # couche 1 est c54x_exe (--dsp) ou gr-gsm + grgsm_exe (--grgsm).
        --dsp)         DSP_MODE=1 ;;
        --grgsm)       DSP_MODE=0 ;;
        # [2026-10-19] --osmo-trx : le banc DSP lance trx.highram.elf et l appli
        # hote transceiver (voir c54x_exe/run.sh, OSMO_TRX=1) ; implique --dsp.
        --osmo-trx|--bts) DSP_MODE=1; OSMO_TRX=1; export OSMO_TRX; BTS_TRX_CHOISI="$1" ;;
        # [2026-10-21] --bts-classic : osmo-bts-trx <-> pont_dsp.py, pas de Calypso TRX (voir l'aide)
        --bts-classic|--bts-classique) DSP_MODE=1; BTS_CLASSIC=1 ;;
        # [2026-10-20] --ts N : 1 = TS0 seul ; 2 = TS0 + TS1, un Calypso TRX par intervalle
        # (OSMO_TRX_TS1=1 : instance 3 pour le TS1). Implique --bts. Exporte plus bas (export_trx_ts).
        --ts)          if [ $# -lt 2 ] || ! [[ "${2:-}" =~ ^(1|2|2/1)$ ]]; then
                           printf '%s\n\n' "--ts attend 1 (TS0 seul), 2 (TS0 + TS1, un Calypso par intervalle) ou
                            ↪  2/1 (TS0 + TS1, un seul Calypso)" >&2
                           usage >&2; exit 2
                       fi
```

```
                OSMO_TRX_TS="$2"; DSP_MODE=1; OSMO_TRX=1; export OSMO_TRX; BTS_TRX_CHOISI=--ts; shift ;;
--ts=*)         OSMO_TRX_TS="${1#*=}"
                [[ "$OSMO_TRX_TS" =~ ^(1|2|2/1)$ ]] || { printf '%s\n\n' "--ts attend 1, 2 ou 2/1 (recu :
                 ↪  $OSMO_TRX_TS)" >&2; usage >&2; exit 2; }
                DSP_MODE=1; OSMO_TRX=1; export OSMO_TRX; BTS_TRX_CHOISI=--ts ;;
# [2026-10-20] --ms : sans --bts c'est le defaut (mobile + pont) ; avec --bts, l'instance 2 par l'air
--ms)           MS_AIR=1 ;;
# [2026-10-21] --ms-magnetite : --ms avec le firmware TI (Magnetite) en flash, voir l'aide
--ms-magnetite) MS_AIR=1; MS_MAGNETITE=1
                MAGNETITE_FLASH="${MAGNETITE_FLASH:-/opt/GSM/firmware/magnetite/c139-flash.img}"
                [ -r "$MAGNETITE_FLASH" ] || { printf '%s\n' "--ms-magnetite : image flash absente :
                 ↪  $MAGNETITE_FLASH (doc/magnetite.md : build fc-magnetite c139 hybrid-ui-vo)" >&2; exit
                 ↪  2; }
              # [2026-10-21 11:10] PAS d'export ici : exporte, FIRMWARE_FLASH atteignait aussi l'instance 1
                # (la radio de la BTS) qui bootait Magnetite au lieu du firmware trx. Il n'est passe qu'au
                # mobile (exec ci-dessous).
                # [2026-10-21 23:20] MS_MAGNETITE, lui, EST exporte : qosmo/run_modules/08-gabarits.sh
                 ↪  regenere
               # les gabarits dans run.sh (autre bash), et sans lui generate_configs.sh posait SDCCH8 au TS1
                # (__TS1_PCHAN_BTS0__) : IMM ASS sur TS1, SABM de Magnetite sur TS0, LU en Timeout.
                export MS_MAGNETITE
                ;;
# [2026-10-22] vocodeur : la ROM du DSP (defaut, 14-demarrage.sh) ou GAPK sur l'hote. Exportes ici :
# /etc/default/osmo-banc (BANC_VOCODEUR=rom, CALYPSO_VSP=1, PONT_UL_RAW_TCH=1) ne remplit que le vide.
--vocoder-rom|--vocodeur-rom) BANC_VOCODEUR=rom; CALYPSO_VSP=1; export BANC_VOCODEUR CALYPSO_VSP ;;
--gapk|--vocodeur-hote) BANC_VOCODEUR=hote; CALYPSO_VSP=0; PONT_UL_RAW_TCH=0
                export BANC_VOCODEUR CALYPSO_VSP PONT_UL_RAW_TCH ;;
--sap)          SAP_MODE=1; export SAP_MODE ;;
--sap=*)        SAP_MODE=1; SAP_TYPE="${1#*=}"; export SAP_MODE SAP_TYPE ;;
--softsim)      SAP_MODE=1; export SAP_MODE ;;   # alias de --sap
--omnikey|--sim-reader) SAP_MODE=1; SAP_TYPE=pcsc; export SAP_MODE SAP_TYPE ;;
--no-sap)       SAP_MODE=0; export SAP_MODE ;;
--launcher)     QOSMO_LAUNCHER="${2:-}"; export QOSMO_LAUNCHER; shift ;;
--launcher=*)   QOSMO_LAUNCHER="${1#*=}"; export QOSMO_LAUNCHER ;;
--wan=*)        WAN_MESH=1
                if [ -f "${1#*=}" ]; then WAN_IMPORT_FILES+=("${1#*=}"); else WAN_NODES="${1#*=}"; fi ;;
--wan-nodes)    WAN_MESH=1; WAN_NODES="${2:-}"; shift ;;
--wan-nodes=*) WAN_MESH=1; WAN_NODES="${1#*=}" ;;
--wan-id)       WAN_MESH=1; WAN_NODE_ID="${2:-}"; shift ;;
--wan-id=*)     WAN_MESH=1; WAN_NODE_ID="${1#*=}" ;;
--wan-conf)     WAN_CONF_FILE="${2:-}"; shift ;;
--wan-conf=*)   WAN_CONF_FILE="${1#*=}" ;;
--payload-ism)    PAYLOAD_ISM=1; export PAYLOAD_ISM ;;
--payload-ism=*) PAYLOAD_ISM=1; PAYLOAD_ISM_LINK="${1#*=}"; export PAYLOAD_ISM ;;
--virtualbox)    WAN_MESH=1; VBOX_INTERCO=1 ;;
--virtualbox=*) WAN_MESH=1; VBOX_INTERCO=1; VBOX_NODES="${1#*=}" ;;
--vbox-node)     VBOX_HOST_NODE="${2:-1}"; shift ;;
--vbox-node=*)   VBOX_HOST_NODE="${1#*=}" ;;
--node)          NODE_ID="${2:-}"; shift ;;
--node=*)        NODE_ID="${1#*=}" ;;
--air-mesh)      AIR_MESH=1 ;;
--air-mesh=*)    AIR_MESH=1; AIRMESH_PORT="${1#*=}" ;;
--regen)         REGEN_GABARITS=1 ;;
--regen-conf)    WAN_MESH=1; REGEN_WAN_CONF=1 ;;
--menu)          MENU_MODE=1 ;;
--op)            NODE_OP="${2:-1}"; shift ;;
--op=*)          NODE_OP="${1#*=}" ;;
--hub-ip)        HUB_IP="${2:-}"; shift ;;
--hub-ip=*)      HUB_IP="${1#*=}" ;;
--interstp)      INTERSTP=1 ;;
--no-interstp)   INTERSTP=0 ;;
# MS GPRS (modem osmocom-bb) en second MS du fake_trx du side-car, avec
# son PCU (/tmp/pcu_bts2) : qosmo run_modules/{58,61,69,71}.
# [2026-10-16] INACTIF par defaut : le banc demarre sans data (full ROM
# DSP seul, voix/SMS). --gprs-modem remonte le lot data a la demande.
--gprs-modem)    MODEM_GPRS=1 ;;
```

```
        --no-gprs-modem) MODEM_GPRS=0 ;;
        --mobile|--ccch_scan|--bcch_scan|--cell_log)
                        # run_modules/70-l2.sh gere deja les quatre clients via
                        # CALYPSO_L2_CLIENT ; il manquait de quoi les choisir.
                        # Un seul a la fois : ils se disputeraient la socket
                        # L1CTL (/tmp/osmocom_l2) et la VTY 4247, et le second
                        # lance resterait muet pour toujours.
                        _l2c="${1#--}"
                        if [ -n "${L2_CLIENT_CHOISI:-}" ] && \
                           [ "$L2_CLIENT_CHOISI" != "$_l2c" ]; then
                            printf '%s\n\n' \
                              "--$_l2c et --$L2_CLIENT_CHOISI sont exclusifs : un seul client de couche 2" >&2
                            usage >&2; exit 2
                        fi
                        L2_CLIENT_CHOISI="$_l2c" ;;
        -h|--help)      usage; exit 0 ;;
        faketrx-qemu|faketrx|qemu|noproc|core|hybrid)
            PROFILE="$1"; PROFILE_CHOISI=1
            [ "$PROFILE" = "hybrid" ] && PROFILE=faketrx-qemu
            [ "$PROFILE" = "core" ]   && PROFILE=noproc
            ;;
        *) printf 'option inconnue : %s\n\n' "$1" >&2; usage >&2; exit 2 ;;
    esac
    shift
done

# [2026-10-21] --bts-classic : la BTS est osmo-bts-trx + pont_dsp.py ; un OSMO_TRX=1 herite de
# l'environnement ne doit pas rallumer le Calypso TRX. --ms / --ms-magnetite restent valables : sans
# OSMO_TRX, 14-demarrage.sh branche le mobile (ou Magnetite, FIRMWARE_FLASH) sur le pont.
if [ "${BTS_CLASSIC:-0}" = 1 ]; then
    if [ -n "${BTS_TRX_CHOISI:-}" ]; then
        printf '%s\n\n' "--bts-classic et $BTS_TRX_CHOISI sont exclusifs : BTS classique (pont_dsp.py) ou Calypso
         ↪  TRX" >&2
        usage >&2; exit 2
    fi
    OSMO_TRX=0; export OSMO_TRX
fi

# [2026-10-20] --ts N -> variables de c54x_exe/run.sh (OSMO_TRX=1). Sans --ts, run.sh garde son defaut
# (OSMO_TRX_TCH=1 : un Calypso, TS0 + TS1, masques 0x03). trx_ts_desc : le libelle des resumes.
export_trx_ts() {
    # [2026-10-21] 2/1 et le defaut : un seul Calypso pour TS0 + TS1 (le seul montage qui passe un appel) ;
    # 2 : un Calypso par intervalle (LU OK, appels KO : fn du telephone 2 non aligne).
    case "${OSMO_TRX_TS:-}" in
        1)      export OSMO_TRX_TCH=0 OSMO_TRX_TS1=0 ;;
        2)      export OSMO_TRX_TCH=0 OSMO_TRX_TS1=1 ;;
        2/1|"") export OSMO_TRX_TCH=1 OSMO_TRX_TS1=0 ;;
    esac
}
trx_ts_desc() {
    if [ "${OSMO_TRX_TS1:-0}" = 1 ]; then
        printf 'TS0 + TS1 par un 2e Calypso (--ts 2 : OSMO_TRX_TS1=1, instance %s, transceiver -2)'
         ↪  "${OSMO_TRX_TS1_INSTANCE:-3}"
    elif [ "${OSMO_TRX_TCH:-0}" = 0 ]; then
        printf 'TS0 seul (--ts 1 : OSMO_TRX_TCH=0, masque 0x01)'
    else
        printf 'TS0 + TS1 par un seul Calypso (OSMO_TRX_TCH=1, masques 0x03)'
    fi
}
export_trx_ts

# Le lot GPRS du side-car (fake_trx --trx, second PCU, trxcon + modem) lit
# MODEM_GPRS dans l'environnement de run.sh : exporte ici, 0 (sans data) par
# defaut, 1 avec --gprs-modem. L'environnement gagne (MODEM_GPRS=1 ./start-native.sh).
export MODEM_GPRS="${MODEM_GPRS:-0}"
```

```
# --- --dsp garde le profil par defaut ---------------------------------------
# [2026-09-22] --dsp forcait le profil `qemu` (pipeline Calypso seul, sans
# side-car), au motif qu'un banc DSP n'a pas a dependre d'une seconde station
# de base. Retire : le banc DSP doit monter la MEME pile que le mode grgsm,
# side-car compris, et c'est l'hybride qui est le defaut. Un profil nomme
# explicitement gagne toujours, comme avant.

# --- ou tourne-t-on ? -------------------------------------------------------
# La question n'est pas cosmetique : le plan d'adressage EN DEPEND.
#   docker  → l'inter-STP est le conteneur osmo-inter-stp (172.20.0.10), le
#             noeud a deja son IP quand les demons demarrent.
#   VM/ISO  → l'inter-STP est une machine a part (192.168.1.49 sur le banc,
#             ou une host-only 192.168.56.x selon le montage), et l'adresse
#             du noeud vient du DHCP : elle peut manquer au lancement.
# Prendre l'un pour l'autre donne un ASP qui ne s'attache jamais, avec pour
# seule trace un "connection refused" vers une adresse inexistante ici.
#
# /.dockerenv seul ne suffit pas - il est vrai dans n'importe quel conteneur.
# On exige aussi /etc/docker-entrypoint-cmd, que scripts/entrypoint.sh depose :
# c'est ce couple qui identifie un conteneur DE CE DEPOT.
detect_runtime_env() {
    if [ -f /.dockerenv ] && [ -f /etc/docker-entrypoint-cmd ]; then
        echo docker; return
    fi
    if [ -f /.dockerenv ] || grep -qa 'docker\|containerd' /proc/1/cgroup 2>/dev/null; then
        echo docker; return
    fi
    # systemd-detect-virt SORT EN CODE 1 quand il ne trouve rien. Un
    # "$(cmd || echo none)" produit alors DEUX lignes - la sortie "none" du
    # programme, plus celle du repli - et aucun motif ne correspond : une
    # machine physique se retrouvait classee "vm". On ignore le code de
    # sortie et on ne garde que la premiere ligne.
    local v
    v="$(systemd-detect-virt 2>/dev/null | head -1)" || true
    v="${v:-none}"
    case "$v" in
        oracle|kvm|qemu|vmware|microsoft|xen|bochs|parallels) echo vm ;;
        none|"") echo bare ;;
        *) echo vm ;;
    esac
}
RUNTIME_ENV="${RUNTIME_ENV:-$(detect_runtime_env)}"

# --- quel noeud sommes-nous ? -----------------------------------------------
# Le numero de noeud decide des point codes, du routing context et de l'indicatif.
# Le taper a chaque lancement est une occasion de se tromper - et une erreur ici
# ne se voit pas : la pile demarre, elle presente simplement au hub l'adresse SS7
# d'un autre noeud. On le deduit donc, et on DIT d'ou vient la reponse.
#
# Ordre, du plus explicite au plus deduit. Le premier qui repond gagne :
#   1. --node passe en ligne de commande
#   2. l'environnement - c'est ainsi que start-docker.sh le transmet au conteneur
#   3. /etc/osmo-role, fige dans l'image par build-iso.sh --node N
#   4. la table WAN : l'adresse de ce noeud y figure, et nous l'avons
#   5. rien - on ne touche alors PAS a l'identite SS7, plutot que d'en inventer une
# Renvoie "numero|origine". Pas deux variables : la fonction est appelee dans
# une substitution de commande, donc dans un SOUS-SHELL - toute variable qu'elle
# poserait y resterait, et l'appelant n'afficherait "?" qu'apres coup.
detect_node_id() {
    local n

    n="${WAN_NODE_ID:-${OSMO_WAN_NODE:-}}"
    if [[ "$n" =~ ^[1-9]$ ]]; then echo "${n}|environnement"; return 0; fi

    if [ -r /etc/osmo-role ]; then
        n="$(awk -F= '/^OSMO_WAN_NODE=/{print $2}' /etc/osmo-role 2>/dev/null | tr -d ' \r')"
```

```
        if [[ "$n" =~ ^[1-9]$ ]]; then echo "${n}|/etc/osmo-role"; return 0; fi
    fi

    # La table WAN et nos propres adresses : c'est la source la plus fiable en VM,
    # ou une seule ISO sert les neuf noeuds et ou seule l'IP les distingue.
    local conf="${WAN_CONF_FILE:-/etc/osmo-wan.conf}"
    if [ -r "$conf" ] && [ -r "$HERE/network/wan-nodes.sh" ]; then
        n="$(
            # Sous-shell : wan-nodes.sh pose des variables et des tableaux qu'on
            # ne veut pas voir deborder dans le lanceur.
            set +u
            . "$HERE/network/wan-nodes.sh" 2>/dev/null || exit 0
            WAN_NODE_ID=0
            wan_nodes_load "$conf" 2>/dev/null || exit 0
            wan_nodes_detect_self 2>/dev/null || exit 0
            printf '%s' "$WAN_NODE_ID"
        )"
        if [[ "$n" =~ ^[1-9]$ ]]; then echo "${n}|table WAN (IP locale)"; return 0; fi
    fi

    return 1
}
```

3.371 /opt/GSM/osmo-operator/startup_scripts/02-noeud-couleurs.sh

5296 octets, 88 lignes → 88 lignes

```
# 02-noeud-couleurs.sh - partie de start.sh (lignes 543-628 de start-native.sh d'origine)
# Sourcee par start.sh dans le shell principal : meme ordre, memes variables.
NODE_ID_SRC=""
if [ -n "$NODE_ID" ]; then
    NODE_ID_SRC="option --node"
else
    _detected="$(detect_node_id || true)"
    NODE_ID="${_detected%%|*}"
    [ -n "$NODE_ID" ] && NODE_ID_SRC="${_detected#*|}"
    unset _detected
fi
# L'adresse du hub arrive par le MEME chemin que le noeud : start-docker.sh la pose
# dans l'environnement du conteneur (OSMO_HUB_IP, et INTER_STP_IP pour les
# scripts plus anciens). Sans ce repli, HUB_IP restait vide alors que la
# reponse etait deja la : un conteneur lance par docker exec -d n'a pas de
# terminal, la question du hub est donc sautee et l'adresse restait nulle
# jusqu'aux configs. Le repli est pose ici, apres la boucle d'options, pour
# que --hub-ip garde le dernier mot.
[ -n "$HUB_IP" ] || HUB_IP="${OSMO_HUB_IP:-${INTER_STP_IP:-}}"
# ── DERNIER REPLI : L ADRESSE QUE L ASP COMPOSE DEJA ──────────────────────
# [2026-08-31] Sans HUB_IP, network/set-node-id.sh retombe sur le hub WAN par
# defaut, 192.168.1.49, calcule la source qui y route (192.168.1.31 ici) et
# l ecrit en local-ip de l ASP. Or sur un banc a une machine le hub vit dans un
# conteneur et l ASP compose 127.0.0.1:2908 - le port SCTP que le hub publie
# sur la boucle. La source calculee pour 192.168.1.49 ne mene donc pas au hub,
# et l ASP reste ASP_DOWN. Vu le 31/08 :
#     asp-to-inter  as-inter  ASP_DOWN  client
#                   Local 172.20.0.11:2910  Remote 127.0.0.1:2908
# (172.20.0.11 etant, lui, la valeur du gabarit jamais corrigee - l adresse du
# PREMIER CONTENEUR, qui n existe meme pas sur l hote.)
# Rien de tout cela ne se signale : le journal ne dit qu un timeout.
#
# La config SAIT ou est le hub - c est son remote-ip. On le lit plutot que de
# laisser deviner : la source et la destination viennent alors du meme
# endroit, et `ip route get` rend la bonne patte locale.
# --hub-ip et OSMO_HUB_IP gardent la priorite, ce repli ne sert qu au silence.
if [ -z "$HUB_IP" ] && [ -r "${OSMOCOM_CFG:-/etc/osmocom}/osmo-stp.cfg" ]; then
```

```
    HUB_IP="$(awk '/^[[:space:]]*asp[[:space:]]+asp-to-inter[[:space:]]/{a=1; next}
                   a && /^[[:space:]]*remote-ip[[:space:]]/{print $2; exit}' \
              "${OSMOCOM_CFG:-/etc/osmocom}/osmo-stp.cfg" 2>/dev/null)"
    # [2026-10-18] Une config jamais regeneree (installation fraiche : gabarit
    # copie tel quel) porte le MARQUEUR __INTER_STP_IP__, pas une adresse. Le
    # prendre pour le hub le passait a set-node-id.sh --hub-ip, qui le refusait,
    # et --regen echouait des le premier lancement. Seule une IPv4 compte.
    [[ "$HUB_IP" =~ ^[0-9]{1,3}(\.[0-9]{1,3}){3}$ ]] || HUB_IP=""
    [ -n "$HUB_IP" ] && echo "  hub        deduit du remote-ip de l ASP : $HUB_IP" >&2
fi
# Pour les scripts qui ne connaissent que deux mondes : docker, ou tout le reste.
case "$RUNTIME_ENV" in docker) NODE_MODE=docker ;; *) NODE_MODE=native ;; esac

# --- affichage (identique a run.sh) -----------------------------------------
if [ -t 1 ] && [ -z "${NO_COLOR:-}" ]; then
    TTY=1
    C_OK=$'\033[32m'; C_KO=$'\033[31m'; C_SK=$'\033[33m'
    C_DIM=$'\033[2m'; C_CYAN=$'\033[36m'; C_BOLD=$'\033[1m'; C_Z=$'\033[0m'
else
    TTY=0
    C_OK=""; C_KO=""; C_SK=""; C_DIM=""; C_CYAN=""; C_BOLD=""; C_Z=""
fi
# ── TUER LES PYTHON DU BANC, ET SEULEMENT EUX ──────────────────────────
# [2026-09-04] Le demontage faisait `killall -9 python3` : TOUS les python3 de
# la machine, encart du bureau compris. On a d abord EPARGNE une liste de
# python du bureau (osmo-fft-snap, osmo-panel...) - liste toujours en retard
# d un outil : le 2026-10-12, le fondu du fond d ecran (tools/osmo-fond.py), la
# zone haute et la sonde des timeslots mouraient encore a chaque --stop.
# [2026-10-12] ON INVERSE : plus de killall, plus de liste d epargnes. On ne
# tue que les python3 DU BANC, nommes un par un par leur ligne de commande :
#   - fake_trx.py et le reste du trx_toolkit d osmocom-bb (radio virtuelle) ;
#   - pont/pont.py (le pont TRX-UDP) et pont/airmesh.py ;
#   - le relais SMS inter-operateurs (scripts/smsc-start.sh) ;
#   - tools/osmo-ril-atmodem.py (le modem AT du telephone).
# Tout autre python3 (bureau, pmbootstrap, un build a cote) n est pas concerne.
# CALYPSO_PY_BANC_EXTRA='motif|motif' ajoute des motifs (regex ERE) sans
# toucher au script.
# [2026-10-20 22:35] + l'air, le pont DSP, le pont audio VSP et les passerelles gdb des instances Calypso :
# un air.py de 22:12 survivait a deux relances et lisait un anneau supprime (plus de LU).
PY_BANC='fake_trx\.py|trx_toolkit/|pont/pont\.py|pont/airmesh\.py|sms-interop|osmo-ril-
↪  atmodem\.py|pont/air\.py|pont/pont_dsp\.py|pont\.vsp_audio|gdb-telnet\.py'
[ -n "${CALYPSO_PY_BANC_EXTRA:-}" ] && PY_BANC="$PY_BANC|$CALYPSO_PY_BANC_EXTRA"
killall_python() {
    local sig="${1:--TERM}" pid tue=0
    for pid in $(pgrep -x python3 2>/dev/null); do
        [ "$pid" = "$$" ] && continue
        tr '\0' ' ' < "/proc/$pid/cmdline" 2>/dev/null | grep -Eq "$PY_BANC" || continue
        kill "$sig" "$pid" 2>/dev/null && tue=1
    done
    [ "$tue" = 1 ]
}
```

3.372 /opt/GSM/osmo-operator/startup_scripts/03-affichage.sh

2862 octets, 57 lignes → 57 lignes

```
# 03-affichage.sh - partie de start.sh (lignes 629-683 de start-native.sh d'origine)
# Sourcee par start.sh dans le shell principal : meme ordre, memes variables.
say_begin() {
    if [ $TTY -eq 1 ]; then
        printf '[ %s.. %s] %s' "$C_DIM" "$C_Z" "$1"
    else
        printf '[ .. ] %s\n' "$1"
    fi
```

```
}
say_end() { # $1=tag $2=couleur $3=libelle $4=detail
    if [ $TTY -eq 1 ]; then printf '\r\033[K'; fi
    printf '[%s%s%s] %s' "$2" "$1" "$C_Z" "$3"
    [ -n "${4:-}" ] && printf ' %s(%s)%s' "$C_DIM" "$4" "$C_Z"
    printf '\n'
}
banner() {
    echo -e "${C_CYAN}"
    echo "╔═══════════════════════════════════════════════════════╗"
    echo "║   Calypso GSM - bootstrap → run.sh                    ║"
    echo "╚═══════════════════════════════════════════════════════╝"
    echo -e "${C_Z}"
}
# --- 1. configuration ------------------------------------------------------
# [2026-08-03] globals.conf - les reglages reseau (MCC/MNC/ARFCN/KI/IMSI/A5...).
# Genere par ./generate_configs.sh cote hote ; ici on se contente de le lire.
# [2026-09-22] « ARFCN=520 ./start-native.sh surcharge sans toucher au
# fichier » : c'etait ecrit ici, et c'etait FAUX. globals.conf n'utilise pas
# l'idiome ":=" mais l'affectation simple (`ENCRYPTION="a5 1"`), et il est
# source avec `set -a` : il ECRASE l'environnement. Son propre en-tete
# l'annonce d'ailleurs (« il est lu en dernier, les 26 reglages ci-dessous
# ecrasent leur homonyme dans l'environnement »), en contradiction avec ce
# commentaire-ci. Constate sur le banc :
#     ENCRYPTION="a5 0" ./start-native.sh --dsp   ->   chiffrement a5 1
# Deux contrats opposes dans le meme depot, et c'est le silencieux qui gagnait.
#
# On tranche du cote de la ligne de commande, qui est la regle partout ailleurs
# dans ce script : on releve les variables de globals.conf que l'appelant a
# DEJA posees et non vides, on lit le fichier, puis on les repose. globals.conf
# reste la reference pour tout ce qui n'a pas ete dit ; un vide continue de
# signifier « valeur calculee », donc ne surcharge rien.
if [ -r "$HERE/globals.conf" ]; then
    _gc_forcees=()
    while IFS= read -r _gc_v; do
        [ -n "${!_gc_v:-}" ] && _gc_forcees+=("$_gc_v=${!_gc_v}")
    done < <(sed -n 's/^\([A-Za-z_][A-Za-z0-9_]*\)=.*/\1/p' "$HERE/globals.conf")
    set -a
    . "$HERE/globals.conf"
    for _gc_kv in "${_gc_forcees[@]}"; do
        eval "${_gc_kv%%=*}=\${_gc_kv#*=}"
    done
    set +a
    [ ${#_gc_forcees[@]} -gt 0 ] && \
        printf '  %sglobals.conf%s surcharge par l environnement : %s\n' \
            "${C_DIM:-}" "${C_Z:-}" "${_gc_forcees[*]}"
    unset _gc_v _gc_kv
fi
```

3.373 /opt/GSM/osmo-operator/startup_scripts/04-environnement.sh

4763 octets, 82 lignes → 82 lignes

```
# 04-environnement.sh - partie de start.sh (lignes 684-763 de start-native.sh d'origine)
# Sourcee par start.sh dans le shell principal : meme ordre, memes variables.
say_begin "Chargement de l'environnement"
if [ -f "$HERE/environment/load.env" ]; then
    set -a; . "$HERE/environment/load.env"; set +a
    say_end " OK " "$C_OK" "Chargement de l'environnement" "environment/load.env"
elif [ -f "$HERE/env/load.env" ]; then
    set -a; . "$HERE/env/load.env"; set +a
    say_end " OK " "$C_OK" "Chargement de l'environnement" "env/load.env"
else
    # Fallback minimal (chemins typiques du depot)
```

```
    : "${GSM_ROOT:=/opt/GSM}"
    : "${NITB_TREE:=$HERE}"
    # Meme selection d'arbre qu'environment/paths.env, repliquee pour le cas ou
    # load.env est absent (ISO, conteneur nu). Un seul arbre QEMU : qosmo ; --dsp
    # ne change pas d'arbre, il passe la couche 1 a c54x_exe.
    : "${CALYPSO_FORK:=qosmo}"
    if [ -x "$HERE/../$CALYPSO_FORK/run.sh" ]; then
        : "${OQC_ROOT:=$(cd "$HERE/../$CALYPSO_FORK" && pwd)}"
    else
        : "${OQC_ROOT:=$GSM_ROOT/$CALYPSO_FORK}"
    fi
    say_end " OK " "$C_OK" "Chargement de l'environnement" "fallback minimal"
fi
: "${RUN_DIR:=/run/osmo-direct}"
# Repli si load.env est absent. Les journaux restent sous RUN_DIR (tmpfs) :
# le defilement tmux en depend - cf. environment/paths.env.
: "${LOG_DIR:=$RUN_DIR/logs}"
# A5/1 par defaut : c'est le chiffrement que la maquette valide de bout en bout
# (Calypso ↔ BTS), et ce que start-docker.sh impose deja dans son hand-off vers ce
# script. Un defaut a "a5 0" faisait diverger le lancement direct du lancement
# par start-docker.sh - meme stack, deux chiffrements, selon la porte d'entree.
#
# ATTENTION : globals.conf est lu AVANT et fait autorite sur cette variable (son
# en-tete le dit). Ce ":=" n'est donc qu'un repli quand globals.conf est absent
# - l'ISO, un conteneur nu. La valeur qui s'applique en pratique vient de la-bas,
# et c'est pourquoi elle y a ete changee aussi.
: "${ENCRYPTION:=a5 1}"
# Le pont TRX par defaut : un transceiver unique (pont/pont.py) qui se presente
# a osmo-bts-trx en TRX-UDP, decode le DL vers GSMTAP 4730/4731 et encode l'UL.
# CALYPSO_BRIDGE n'est pas declare dans globals.conf (les CALYPSO_* "passent au
# travers, intact" d'apres son en-tete) : l'environnement gagne donc vraiment.
#   CALYPSO_BRIDGE=none ./start-native.sh   -> pas de pont (QEMU + BTS seuls)
#
# [2026-09-22] DEUX CHAINES, SELECTIONNEES PAR --dsp / --grgsm. [2026-10-01]
# LE DEFAUT EST LE DSP (DSP_MODE=1 en tete de fichier). gr-gsm (qosmo +
# grgsm_exe) reste a un --grgsm : la couche 1 est alors un decodeur gr-gsm
# dans QEMU, et c'est la pile qui va de bout en bout jusqu'a l'appel voix.
#
# `--dsp` ne designe PLUS un fork de QEMU. Le C54x tourne maintenant HORS de
# QEMU, dans /opt/GSM/c54x_exe (mask-ROM TI, coeur emule par l1-dsp), et QEMU
# n'est plus que l'ARM, lance avec CALYPSO_DSP_EXTERN=1 depuis l'arbre
# /opt/GSM/qosmo. Les cinq etapes sont dans c54x_exe/run.sh :
#   1. c54x_exe --arm        le DSP, socket /tmp/calypso_dsp.sock + API RAM
#                            partagee, bursts DL en UDP 6702
#   2. qemu-system-arm       l'ARM + layer1 osmocom-bb, couche 1 gr-gsm coupee
#   3. osmocon               romload sur le pty serie, relais L1CTL
#   4. mobile                couches 2/3 osmocom-bb
#   5. pont.py               TRXD 5700-5702 face a osmo-bts-trx, bursts vers
#                            le DSP, montant par les side-bands /dev/shm
# Ce qu'il fait : le mobile acquiert FB et SB, decode SI1-4 (lai=001-01-1),
# campe, et mene sa mise a jour de localisation. L'ancienne note « le decodage
# SCH ne passe pas, le mobile ne campe pas » datait de l'ancien fork qosmo-dsp
# et n'est plus vraie de cette chaine-la.
#
# [2026-09-25] Les anciens forks qosmo-dsp et qosmo-grgsm ne sont plus
# references : run.sh, run_modules, cfgs et environnement ont ete repris dans
# /opt/GSM/qosmo, la ROM DSP est dans /opt/GSM/c54x_exe/rom (et en
# /opt/GSM/calypso_dsp.*.bin, ou c54x_exe la lit par defaut).
#
# --dsp : PAS DE PONT LANCE PAR CE SCRIPT. c54x_exe/run.sh lance le sien
# (etape 5) sur 5700-5702 ; un second pont sur les memes ports mourrait sur
# « Address already in use » et laisserait un cadavre a chaque demarrage.
# `none` = rien a lancer ici. Ne PAS mettre `ipc` : dans l'ancien qosmo-dsp
# cette valeur voulait dire « MS#1 via osmo-trx-ms-ipc, firmware QEMU non lance ».
# Une valeur posee par l'operateur reste respectee.
#
```

```
# [2026-10-01] Le DSP etant le defaut, `none` ne vaut que si le banc DSP est
# VRAIMENT lance (DSP_BANC=1, decide juste en dessous) : un c54x_exe absent
# fait retomber sur gr-gsm, qui a besoin de son pont. Sinon la chaine gr-gsm
# montait sans pont, et personne ne campait.
```

3.374 /opt/GSM/osmo-operator/startup_scripts/05-run-sh.sh

4797 octets, 82 lignes → 82 lignes

```
# 05-run-sh.sh - partie de start.sh (lignes 764-843 de start-native.sh d'origine)
# Sourcee par start.sh dans le shell principal : meme ordre, memes variables.
: "${MS_COUNT:=2}"
: "${HOST_IP:=127.0.0.1}"
mkdir -p "$RUN_DIR" "$LOG_DIR" /root/.osmocom/bb 2>/dev/null || true
# --- 2. detection des chemins / binaires -------------------------------------
say_begin "Resolution de run.sh"
# [2026-08-08] Le chemin etait cable en dur ICI, ce qui rendait MORTE toute la
# resolution de OQC_ROOT faite plus haut (trois candidats testes pour rien) et
# contredisait le contrat annonce en tete de fichier ("la ligne de commande
# gagne toujours") : RUN_SH etait la seule variable non surchargeable.
# Ordre : RUN_SH explicite > OQC_ROOT resolu > chemin historique.
# [2026-09-22] LE BANC DSP S'AJOUTE AU PIPELINE, IL NE LE REMPLACE PAS.
# Premiere version : --dsp mettait RUN_SH = c54x_exe/run.sh. La pile osmocom ne
# montait alors plus du tout -- le coeur, la BTS, le side-car et tmux sont dans
# les run_modules du fork, que ce run.sh-la ne connait pas. Rapport du banc :
# « il faut lancer la pile osmo aussi comme mode=grgsm ».
# Donc : RUN_SH reste celui du fork, il monte tout comme d'habitude, et on lui
# retire seulement les quatre modules de la chaine Calypso que c54x_exe assure
# (qemu, pty, osmocon, l2 — soit les etapes 2, 3 et 4 de son run.sh). Le banc
# DSP est lance juste apres, avec ses cinq etapes.
# BANC_DSP=<chemin> pour en designer un autre ; BANC_DSP=none le desactive.
DSP_BANC=0
DSP_MODULES_RETIRES="qemu,pty,osmocon,l2"
: "${BANC_DSP:=$GSM_ROOT/c54x_exe/run.sh}"
if [ "${DSP_MODE:-0}" = 1 ] && [ "$BANC_DSP" != none ] && [ -x "$BANC_DSP" ]; then
    DSP_BANC=1
elif [ "${DSP_MODE:-0}" = 1 ] && [ "$BANC_DSP" != none ]; then
    # Le defaut est le DSP, mais sans son banc : on le dit, et la chaine gr-gsm
    # prend le relais (CALYPSO_BRIDGE reste `pont`).
    printf '  %s!%s banc DSP absent (%s) - couche 1 gr-gsm a la place\n' "${C_KO:-}" "${C_Z:-}" "$BANC_DSP" >&2
fi
if [ "$DSP_BANC" = 1 ]; then
    : "${CALYPSO_BRIDGE:=none}"
fi
: "${CALYPSO_BRIDGE:=pont}"
: "${RUN_SH:=${OQC_ROOT:+$OQC_ROOT/run.sh}}"
: "${RUN_SH:=$GSM_ROOT/${CALYPSO_FORK:-qosmo}/run.sh}"
if [ ! -x "$RUN_SH" ]; then
    say_end "FAIL" "$C_KO" "Resolution de run.sh" "$RUN_SH introuvable ou non executable"
    printf '       %s→ Verifiez que %s existe et est executable%s\n' "$C_DIM" "$RUN_SH" "$C_Z"
    printf '       %s→ run.sh vit dans /opt/GSM/qosmo (repris de l ancien qosmo-grgsm)%s\n' "$C_DIM" "$C_Z"
    exit 1
fi
say_end " OK " "$C_OK" "Resolution de run.sh" "$RUN_SH"
RUN_ROOT="$(dirname "$RUN_SH")"

# ── LE BANC DSP : SON VOCABULAIRE ──────────────────────────────────────────
# c54x_exe/run.sh ne prend que --stop | --status | --logs | --step N, et rien
# d'autre : lui passer --profile le fait sortir sur « option inconnue ».
# Le coeur reseau, la BTS et le side-car restent montes par le plan du fork ;
# ce banc-ci n'assure que la chaine Calypso (ses cinq etapes).
# [2026-09-22] L'ARRET NE DEMANDE PAS --dsp.
# `./start-native.sh --stop` sans --dsp laissait c54x_exe, QEMU, osmocon, le
# mobile et pont.py debout : DSP_BANC vaut 0 quand --dsp n'est pas retape, et
```

```
# la pile paraissait arretee alors que cinq processus tenaient encore les ports
# 5700-5702 et /tmp/calypso_dsp.sock -- de quoi faire echouer le demarrage
# suivant sur un etat qu'on croit propre. On arrete donc le banc des qu'il
# existe ; son --stop ne fait rien de mal quand il ne tourne pas.
#
# ⚠ MODE=dsp EXPLICITE A CHAQUE APPEL. start-native exporte son propre MODE
# (le profil : « faketrx-qemu »), et c54x_exe/run.sh lit la MEME variable pour
# choisir dsp|grgsm. Sans cela il sort sur
#     [run] ECHEC : MODE=faketrx-qemu inconnu (dsp|grgsm)
# et l'arret du banc ne se fait pas -- silencieusement, puisque l'appelant
# ignore son code de retour. Deux scripts, deux sens pour le meme nom : on ne
# renomme pas celui du banc (run_real.sh et les habitudes s'en servent), on
# pose la valeur a chaque fois.
# [2026-09-23] PulseAudio allege a chaque start : speech-dispatcher arrete,
# annuleur speex au lieu de webrtc, osmo_rec en 8 kHz mono (lib/audio.sh,
# alleger_audio). audio-chain.sh est idempotent ; AUDIO_ALLEGER=0 pour ne
# rien toucher, AUDIO_AEC_METHOD=webrtc pour garder l ancien annuleur.
audio_start() {
    [ -f "$HERE/scripts/audio-chain.sh" ] || return 0
    say_begin "Audio (PulseAudio allege)"
    if bash "$HERE/scripts/audio-chain.sh" 5 >/tmp/osmo-audio-chain.log 2>&1; then
        say_end " OK " "$C_OK" "Audio (PulseAudio allege)" "journal /tmp/osmo-audio-chain.log"
    else
        say_end " !! " "$C_KO" "Audio (PulseAudio allege)" "voir /tmp/osmo-audio-chain.log"
    fi
}
```

3.375 /opt/GSM/osmo-operator/startup_scripts/06-banc-dsp-menu.sh

35372 octets, 674 lignes → 674 lignes

```
# 06-banc-dsp-menu.sh - partie de start.sh (lignes 844-1511 de start-native.sh d'origine)
# Sourcee par start.sh dans le shell principal : meme ordre, memes variables.
banc_dsp() {
    MODE=dsp bash "$BANC_DSP" "$@"
}
banc_dsp_arreter() {
    [ -x "$BANC_DSP" ] || return 0
    banc_dsp --stop "$@"
}
# [2026-10-20 22:35] TOUTES les instances Calypso (INSTANCE=N : /tmp/c54x-pontN, le mobile par l'air = 2,
# le telephone 2 du TRX = 3, les essais = 4...). run.sh --stop (instance 1) entraine deja 2 et 3 ; ceci
# ramasse le reste, puis les orphelins par leur nom (c54x_exe, QEMU Calypso, transceiver) : un c54x_exe
# de l'instance 2 vieux de 19 minutes etait encore la apres deux relances.
# [2026-10-21] banc-rssi (instance 4) et Magnetite (/dev/shm/magnetite, QEMU « -M calypso » comme le banc)
# sont SEPARES : on n'arrete ni l'un ni l'autre ici. Les pid proteges sont lus dans leurs fichiers pid,
# les pkill ci-dessous passent par ce filtre au lieu de tuer tout ce qui matche.
banc_dsp_pids_proteges() {
    cat /tmp/c54x-pont4/*.pid /dev/shm/magnetite/*.pid 2>/dev/null | tr '\n' ' '
}
banc_dsp_tuer_sauf_proteges() {
    local nom p garde
    garde=" $(banc_dsp_pids_proteges) "
    # pgrep -x : nom du processus (comm), pas la ligne de commande : un shell qui contient ce texte n'est pas vise
    for nom in "$@"; do
        for p in $(pgrep -x -- "$nom" 2>/dev/null); do
            case "$garde" in *" $p "*) ;; *) kill "$p" 2>/dev/null || true ;; esac
        done
    done
}
banc_dsp_arreter_instances() {
    local d n
    for d in /tmp/c54x-pont[0-9]*; do
        [ -d "$d" ] || continue
```

```
        n="${d#/tmp/c54x-pont}"
        [ "$n" = 4 ] && continue                    # banc-rssi : arret par banc-rssi-stop seulement
        ls "$d"/*.pid >/dev/null 2>&1 || continue
        [ -x "$BANC_DSP" ] && INSTANCE="$n" AIR=1 OSMO_TRX=0 MODE=dsp bash "$BANC_DSP" --stop >/dev/null 2>&1 ||
         ↪  true
        rm -f "$d"/*.pid
    done
    banc_dsp_tuer_sauf_proteges 'c54x_exe --arm' 'qemu-system-arm -M calypso' 'transceiver -l /tmp/osmocom_l2'
    for k in /tmp/calypso_keys[0-9]*.sock /tmp/osmocom_l2_[0-9]*; do
        case "$k" in /tmp/calypso_keys4.sock|/tmp/osmocom_l2_4) continue ;; esac
        rm -f "$k" 2>/dev/null
    done
    return 0
}

# --- 2b. lanceur C de QEMU (qosmo) ------------------------------------------
# [2026-09-03] QEMU n'est plus appele directement par 40-qemu.sh : l'arbre qosmo
# porte un lanceur C, tools/qosmo-launch, installe dans /usr/local/bin/qosmo. Il fixe les defauts qui marchent
 ↪  (machine calypso + ROMs DSP,
# -cpu arm946, -gdb tcp::1234, -serial pty x2, -monitor unix:$RUN_DIR/...),
# lit l1s/last_rach dans l'ELF, et publie des liens stables vers les pty
# ($RUN_DIR/modem.pty pour osmocon, $RUN_DIR/irda.pty). Options : --bind,
# --trx-port, --iq-tee, --l1ctl, --monitor, --gdb (voir `qosmo --help`).
# Le nom est passe a run.sh par QOSMO_LAUNCHER ; s'il manque, 40-qemu.sh
# retombe sur la ligne qemu-system-arm historique - on le dit, sans bloquer.
say_begin "Lanceur QEMU"
: "${QOSMO_LAUNCHER:=/usr/local/bin/${CALYPSO_FORK:-qosmo}}"
export QOSMO_LAUNCHER
if [ -x "$QOSMO_LAUNCHER" ]; then
    say_end " OK " "$C_OK" "Lanceur QEMU" "$QOSMO_LAUNCHER ($("$QOSMO_LAUNCHER" -V 2>/dev/null | head -1))"
elif [ -f "$RUN_ROOT/tools/qosmo-launch/qosmo-launch.c" ] && command -v gcc >/dev/null 2>&1 \
     && make -s -C "$RUN_ROOT/tools/qosmo-launch" install >/dev/null 2>&1 && [ -x "$QOSMO_LAUNCHER" ]; then
    say_end " OK " "$C_OK" "Lanceur QEMU" "$QOSMO_LAUNCHER (compile a l'instant depuis
     ↪  $RUN_ROOT/tools/qosmo-launch)"
else
    say_end "WARN" "$C_SK" "Lanceur QEMU" "$QOSMO_LAUNCHER absent : 40-qemu.sh appellera qemu-system-arm
     ↪  directement"
    printf '       %s-> make -C %s/tools/qosmo-launch install%s\n' "$C_DIM" "$RUN_ROOT" "$C_Z"
fi
# == --menu : on DEMANDE, au lieu de deviner ================================
# Le lanceur resout tout seul le profil, le noeud, l'inter-STP, le WAN. C'est ce
# qu'il faut pour une ISO qui demarre sans personne devant. Mais devant un banc,
# la meme resolution silencieuse oblige a relire le script pour savoir ce qui a
# ete choisi - et a relancer avec les bonnes options quand ce n'etait pas ca.
#
# Ce menu ne CHANGE aucune regle : il PRE-REMPLIT chaque question avec la valeur
# que le lanceur aurait prise seul, et n'ecrit que ce que l'operateur modifie.
# Valider tout sans rien toucher donne donc exactement le lancement par defaut.
#
# Il s'efface de lui-meme quand personne ne peut repondre (pas de terminal,
# --dry-run, NO_MENU=1 que start-docker.sh pose devant un conteneur) : une question
# sans lecteur, c'est un lancement qui se fige sans le dire.
menu_interactif() {
    [ "${MENU_MODE:-0}" -eq 1 ] || return 0
    [ "${NO_MENU:-0}" = "1" ] && return 0
    [ "$DRY" -eq 1 ] && return 0
    [ -t 0 ] || { printf "  %smenu%s       ignore : pas de terminal\n" "${C_DIM:-}" "${C_Z:-}"; return 0; }

    local has_wt=0
    command -v whiptail >/dev/null 2>&1 && has_wt=1

    # Une question, une valeur par defaut, une reponse. Sans whiptail on
    # retombe sur read : une ISO minimale n'a pas toujours newt.
    _ask() {   # $1 = titre  $2 = question  $3 = defaut
        local rep
        if [ "$has_wt" -eq 1 ]; then
```

```
        rep="$(whiptail --title "$1" --inputbox "$2" 11 70 "$3" 3>&1 1>&2 2>&3)" || rep="$3"
    else
        printf "  %s [%s] : " "$2" "$3" >&2
        read -r rep </dev/tty || rep=""
    fi
    printf "%s" "${rep:-$3}"
}
_yesno() { # $1 = titre  $2 = question  $3 = defaut (0=oui 1=non)
    if [ "$has_wt" -eq 1 ]; then
        if [ "$3" -eq 0 ]; then whiptail --title "$1" --yesno "$2" 10 70; else
            whiptail --title "$1" --defaultno --yesno "$2" 10 70; fi
        return $?
    fi
    local d="o" rep
    [ "$3" -eq 0 ] || d="n"
    printf "  %s [%s] : " "$2" "$d" >&2
    read -r rep </dev/tty || rep=""
    case "${rep:-$d}" in [oOyY]*) return 0 ;; *) return 1 ;; esac
}

# Un choix parmi une liste FERMEE. `_ask` est une saisie libre : elle
# convient a un numero de noeud, pas a « gr-gsm ou DSP », ou toute frappe
# hors liste part en lancement casse. whiptail --radiolist rend le jeton
# choisi ; sans newt, on numerote et on lit un chiffre.
#   $1 titre  $2 question  $3 defaut  $4.. = jeton libelle jeton libelle...
_menu() {
    local titre="$1" q="$2" def="$3"; shift 3
    local rep
    if [ "$has_wt" -eq 1 ]; then
        local args=() j l
        while [ $# -gt 0 ]; do
            j="$1"; l="$2"; shift 2
            if [ "$j" = "$def" ]; then args+=("$j" "$l" ON); else args+=("$j" "$l" OFF); fi
        done
        rep="$(whiptail --title "$titre" --radiolist "$q" 15 74 4 "${args[@]}" 3>&1 1>&2 2>&3)" || rep="$def"
    else
        local -a jetons=() i=1
        printf "  %s\n" "$q" >&2
        while [ $# -gt 0 ]; do
            jetons+=("$1")
            printf "    %d) %-12s %s\n" "$i" "$1" "$2" >&2
            shift 2; i=$((i+1))
        done
        printf "  choix [%s] : " "$def" >&2
        read -r rep </dev/tty || rep=""
        case "$rep" in
            ''|*[!0-9]*) rep="$def" ;;
            *) rep="${jetons[$((rep-1))]:-$def}" ;;
        esac
    fi
    printf "%s" "${rep:-$def}"
}

printf "  %smenu%s       --menu : les defauts proposes sont ceux du lancement automatique\n" \
    "${C_DIM:-}" "${C_Z:-}"

# ── 1. QUI FAIT LA COUCHE 1 ──────────────────────────────────────
# Le premier choix, parce qu il commande le second : le DSP n a que faire
# d une seconde station de base.
_L1="$(_menu "Couche 1" \
    "Qui demodule ?\n\n  DSP     (defaut) le C54x execute la mask-ROM TI et decode lui-meme :\n          le
     ↪  mobile campe (SI1-4) et fait sa mise a jour de localisation.\n  gr-gsm  la pile va jusqu a l appel
     ↪  voix. Le DSP ne tourne pas." \
    "$([ "${DSP_MODE:-0}" = 1 ] && echo c54x_exe || echo grgsm_exe)" \
    c54x_exe  "DSP C54x hors QEMU, mask-ROM TI (defaut)" \
    grgsm_exe "couche 1 gr-gsm dans qosmo (pile complete)")"
```

```
    case "$_L1" in c54x_exe) DSP_MODE=1 ;; *) DSP_MODE=0 ;; esac

    # ── 2. TOPOLOGIE : SEUL OU HYBRIDE ──────────────────────────────────
    # Meme axe que CALYPSO_PROFILE, dit dans les termes du banc :
    #   seul     = profil `qemu`          -> une BTS, un QEMU
    #   hybride  = profil `faketrx-qemu` -> + side-car fake_trx et un MS#2
    # Le defaut suit le choix ci-dessus : hybride en gr-gsm, seul en DSP.
    _topo_def=hybride
    [ "${DSP_MODE:-0}" = 1 ] && _topo_def=seul
    [ "$PROFILE" = qemu ] && _topo_def=seul
    _topo="$(_menu "Topologie" \
        "Combien de stations de base ?\n\n  seul     une BTS et un QEMU. C est ce qu il faut pour le DSP.\n
        ↪  hybride  + un side-car fake_trx et un second mobile, pour que\n           MS#2 appelle MS#1 sans
        ↪  sortir de la maquette." \
        "$_topo_def" \
        seul    "une BTS, un QEMU" \
        hybride "+ side-car fake_trx et MS#2")"
    case "$_topo" in
        seul)    PROFILE=qemu ;;
        hybride) PROFILE=faketrx-qemu ;;
    esac
    PROFILE_CHOISI=1
    unset _topo _topo_def

    # Les autres profils (faketrx seul, noproc) restent joignables, mais en
    # ligne de commande : les proposer ici alourdirait le choix courant.
    PROFILE="$(_ask "Profil" \
        "Profil de pile retenu (modifiable) :\n  faketrx-qemu = hybride   qemu = seul\n  faketrx  noproc  aussi
        ↪  acceptes" \
        "$PROFILE")"

    # Le noeud est deja resolu plus haut (--node, environnement, /etc/osmo-role,
    # table WAN) : on propose CE resultat, pas un 1 arbitraire.
    NODE_ID="$(_ask "Noeud SS7" \
        "Numero de ce noeud (1-9). Il fixe le point code 1.<noeud><op>.<role>,\nle routing context et les MSISDN
        ↪  <noeud>00<op><rang>." \
        "${NODE_ID:-1}")"
    NODE_OP="$(_ask "Operateur" "Operateur porte par ce noeud (1-9) :" "${NODE_OP:-1}")"
    HUB_IP="$(_ask "Inter-STP" \
        "Adresse du hub SS7 (vide = ne pas toucher au remote-ip en place) :" \
        "${HUB_IP:-$(awk -F= '/^OSMO_HUB_IP=/{gsub(/[ \r\t]/,"",$2);v=$2} END{print v}'
        ↪  "${ROLE_FILE:-/etc/osmo-role}" 2>/dev/null)}")"

    _yesno "WAN" "Monter le maillage WAN entre noeuds ?" "$([ "${WAN_MESH:-0}" -eq 1 ] && echo 0 || echo 1)" \
        && WAN_MESH=1 || WAN_MESH=0

    _yesno "Configurations" \
        "Regenerer /etc/osmocom et /etc/asterisk depuis les gabarits ?\n\nLa pile sera ARRETEE, les fichiers
        ↪  sauvegardes, puis reecrits\navec l'identite du noeud ci-dessus." \
        "$([ "${REGEN_GABARITS:-0}" -eq 1 ] && echo 0 || echo 1)" \
        && REGEN_GABARITS=1 || REGEN_GABARITS=0

    printf "  %smenu%s       profil=%s  noeud=%s  op=%s  hub=%s  wan=%s  regen=%s\n" \
        "${C_DIM:-}" "${C_Z:-}" "$PROFILE" "${NODE_ID:-?}" "$NODE_OP" \
        "${HUB_IP:-inchange}" "$WAN_MESH" "$REGEN_GABARITS"
}
menu_interactif

# Profil → mapping vers les profils attendus par run.sh
case "$PROFILE" in
    faketrx-qemu|hybrid) CALYPSO_PROFILE=hybrid;   MODE=faketrx-qemu ;;
    faketrx)             CALYPSO_PROFILE=faketrx;  MODE=faketrx ;;
    qemu)                CALYPSO_PROFILE=calypso;  MODE=qemu ;;
    noproc|core)         CALYPSO_PROFILE=core;     MODE=noproc; RUN_NO_PROCESS=1 ;;
    *)                   CALYPSO_PROFILE=hybrid;   MODE=faketrx-qemu; PROFILE=faketrx-qemu ;;
esac
```

```
export CALYPSO_PROFILE MODE
export RUN_NO_PROCESS="${RUN_NO_PROCESS:-0}"
export ENCRYPTION MS_COUNT HOST_IP
# RAND passe en prefixe (ex. `RAND=0 ./start-native.sh`) force le defi
# d'authentification, via le gate getenv("RAND") des patchs osmo-hlr + osmo-msc.
# Le RAND doit atteindre LES DEUX process : osmo-hlr le tire dans le vecteur
# (SRES/Kc calcules sur RAND=0 -> LU acceptee) et osmo-msc l'envoie. RAND pose
# sur osmo-msc SEUL -> mismatch -> LU reject. RAND absent -> reseau normal.
# core_svc_start (run.sh) lance ces demons de deux facons, on couvre les deux :
#   - enfant direct (setsid, pas de systemd) -> heritent de l'env : on exporte.
#   - service systemd -> `systemctl start` N'herite PAS de l'env : drop-in
#     transient /run/systemd/system/<unit>.service.d/rand.conf + daemon-reload.
[ -n "${RAND:-}" ] && export RAND
if command -v systemctl >/dev/null 2>&1; then
    _rand_changed=0
    for _u in osmo-hlr osmo-msc; do
        systemctl cat "$_u" >/dev/null 2>&1 || continue
        _dd="/run/systemd/system/$_u.service.d"
        [ -f "$_dd/rand.conf" ] && _rand_changed=1
        if [ -n "${RAND:-}" ]; then
            mkdir -p "$_dd" 2>/dev/null \
                && printf '[Service]\nEnvironment=RAND=%s\n' "$RAND" > "$_dd/rand.conf" \
                && _rand_changed=1
        else
            rm -f "$_dd/rand.conf" 2>/dev/null || true
        fi
    done
    [ "$_rand_changed" = 1 ] && systemctl daemon-reload 2>/dev/null || true
    # rebondir seulement si le forcage change, et HLR AVANT MSC pour que le MSC
    # reprenne des vecteurs a RAND=0 ; un run normal sans RAND ne touche a rien.
    if [ -n "${RAND:-}" ] || [ "$_rand_changed" = 1 ]; then
        for _u in osmo-hlr osmo-msc; do
            systemctl is-active --quiet "$_u" 2>/dev/null && systemctl restart "$_u" 2>/dev/null || true
        done
    fi
    unset _u _dd _rand_changed
fi
# Client de couche 2 : l'option de ligne de commande ECRASE tout, et elle est
# EXPORTEE -- une valeur posee sans export ne traverse pas jusqu'a run.sh,
# defaut deja paye sur ENCRYPTION (cf. generate_configs.sh).
if [ -n "${L2_CLIENT_CHOISI:-}" ]; then
    CALYPSO_L2_CLIENT="$L2_CLIENT_CHOISI"
fi
export CALYPSO_L2_CLIENT="${CALYPSO_L2_CLIENT:-mobile}"
# ── LE TAMPON DU GREFFON ALSA-PULSEAUDIO DU `mobile` ──────────────────────
# [2026-09-09] 240 ms, PAS 80. Le mecanisme est celui deja documente dans
# qosmo/run_modules/70-l2.sh (« DESCENDANT MUET ») : le `mobile` ecrit une
# trame GSM (160 echantillons, 20 ms) par `snd_pcm_writei`, et si le tampon du
# greffon est trop court chaque ecriture rend -EPIPE ; pq_alsa.c y repond par
# un `snd_pcm_prepare()` muet, qui DEMONTE ET REMONTE le flux PulseAudio.
# Le 80 ms retenu a l epoque ne tient plus des qu un consommateur lit
# gsm_audio.monitor (notre bouclage vers les haut-parleurs) : la latence
# minimale du sink monte a 26 ms et les remontages reviennent. Mesure du jour,
# ecrivain cadence a 20 ms comme le mobile, 10 s, sur ce banc :
#     80 ms -> 16 remontages   120 -> 19   160 -> 1   200 -> 0   240 -> 0
# Pendant un vrai appel c etait 50 remontages par seconde : flux detruit a
# chaque trame, « voix saturee et pas belle ». 240 ms = 12 trames GSM, le
# premier palier stable (200) avec de la marge pour la gigue sous charge.
# La variable est exportee ICI parce que update.sh re-fetch qosmo :
# son defaut interne peut repartir a 80, celui-ci gagne (`:=`).
# JUGE, pendant un appel etabli : l index du sink-input du mobile doit rester
# FIXE (`pactl list short sink-inputs`, deux releves a 1 s d intervalle).
export CALYPSO_PULSE_LATENCY_MSEC="${CALYPSO_PULSE_LATENCY_MSEC:-320}"
export LOG_DIR RUN_DIR
# --- 2c. firmware ARM : FIRMWARE_SRC ----------------------------------------
# Les deux lanceurs (qosmo/run.sh via bench.env, c54x_exe/run.sh) lisent
```

```
# FIRMWARE_ELF / FIRMWARE_BIN dans l environnement. FIRMWARE_SRC est le nom
# que les scripts de packaging emploient (build-debs.sh, 71-debs-banc.sh) ;
# on l accepte ici aussi pour qu un appel
#     FIRMWARE_SRC=/chemin/layer1.xxx.elf ./start-native.sh
# charge ce firmware-la dans QEMU. Un fichier = l ELF lui-meme ; un dossier =
# l arborescence firmware/ (FIRMWARE_DIR, board/<carte>/ en dessous).
# FIRMWARE_ELF explicite garde la main.
if [ -n "${FIRMWARE_SRC:-}" ]; then
    if [ -f "$FIRMWARE_SRC" ]; then
        : "${FIRMWARE_ELF:=$FIRMWARE_SRC}"
    elif [ -d "$FIRMWARE_SRC" ]; then
        : "${FIRMWARE_DIR:=$FIRMWARE_SRC}"
        export FIRMWARE_DIR
    else
        printf '  %sFIRMWARE_SRC introuvable : %s%s\n' "$C_SK" "$FIRMWARE_SRC" "$C_Z" >&2
    fi
fi
if [ -n "${FIRMWARE_ELF:-}" ]; then
    # Le .bin a cote de l ELF s il existe (qosmo --bin, c54x_exe) ; sinon on
    # laisse les lanceurs appliquer leur propre defaut.
    [ -n "${FIRMWARE_BIN:-}" ] || [ ! -r "${FIRMWARE_ELF%.elf}.bin" ] || FIRMWARE_BIN="${FIRMWARE_ELF%.elf}.bin"
    export FIRMWARE_ELF FIRMWARE_BIN
fi
# --- 3. validation des chemins critiques -------------------------------------
say_begin "Validation des chemins"
path_ok=1
_check() {
    local name="$1" val="${!1:-}"
    if [ -z "$val" ]; then
        printf '\n       %s%s non defini%s\n' "$C_DIM" "$name" "$C_Z"
        path_ok=0
    elif [ -e "$val" ]; then
        :
    else
        printf '\n       %s%s introuvable : %s%s\n' "$C_DIM" "$name" "$val" "$C_Z"
        path_ok=0
    fi
}
# Variables optionnelles selon le profil ; on ne bloque que si presentes et cassees
# (DSP_PROM0 retire : ROM plus generee depuis le merge `sans-dsp` de
#  qosmo — cf. run.sh --check-paths et run_modules/00-prereqs.sh.)
for v in QEMU_BIN FIRMWARE_ELF OSMOCON; do
    # `[ -n x ] && _check || true` avalait le verdict : _check pouvait poser
    # path_ok=0 sans que la boucle ne le laisse remonter. Forme explicite.
    if [ -n "${!v:-}" ]; then _check "$v"; fi
done
if [ $path_ok -eq 1 ]; then
    say_end " OK " "$C_OK" "Validation des chemins"
else
    say_end "WARN" "$C_SK" "Validation des chemins" "certains chemins manquent (run.sh verifiera)"
fi
# --- 4. generation des configs mobile ----------------------------------------
# MS#1 (QEMU / mobile principal) et MS#2 (faketrx side-car) pour le profil hybrid.
BB_DIR="/root/.osmocom/bb"
mkdir -p "$BB_DIR"
# Sources possibles pour le template mobile
MS_TEMPLATE=""
for t in \
    "${OQC_ROOT:-}/cfgs/mobile_group1.cfg" \
    "$RUN_ROOT/cfgs/mobile_group1.cfg" \
    "$HERE/configs/mobile.cfg.template" \
    "$HERE/configs/mobile.cfg" \
    "$BB_DIR/mobile_group1.cfg" \
    "$BB_DIR/mobile.cfg"
do
    if [ -f "$t" ]; then MS_TEMPLATE="$t"; break; fi
```

```
done
# La SIM du MS : « sap » pour le MS#1 quand --sap est demande (son socket est
# celui du serveur SAP, SAP_SOCK), « test » (la SIM de test du gabarit) sinon.
ms_sim_mode() {
    if [ "${SAP_MODE:-0}" = 1 ] && [ "$1" = "${SAP_SOCK:-/tmp/osmocom_sap}" ]; then echo sap; else echo test; fi
}
generate_mobile_cfg() {
    local dest="$1" vty_port="$2" l2sock="$3" sapsock="$4" arfcn="$5" imsi="$6" ki="$7"
    # [2026-08-08] Peripheriques ALSA PAR MOBILE. Le template porte
    # gsm_out/gsm_in pour tout le monde ; laisser ca remettrait les deux
    # mobiles sur le meme sink et la meme source, c'est-a-dire la boucle
    # audio qu'on vient d'ouvrir (cf /etc/asound.conf).
    local aout="${8:-gsm_out}" ain="${9:-gsm_in}"
    # [2026-08-08] GARDE-FOU : la liste de templates contient $BB_DIR/mobile.cfg,
    # qui est aussi une DESTINATION. Si c'est lui qui est retenu, `sed "$tpl" >
    # "$dest"` avec tpl == dest fait tronquer le fichier par le shell AVANT que
    # sed ne le lise : on produit un mobile.cfg VIDE et le mobile ne demarre plus.
    # On bascule alors sur le template de secours plutot que de se detruire.
    local tpl="$MS_TEMPLATE"
    if [ -n "$tpl" ] && [ "$tpl" -ef "$dest" ] 2>/dev/null; then
        printf '\n       %sMS_TEMPLATE == destination (%s) : template de secours%s\n' \
            "$C_DIM" "$dest" "$C_Z"
        tpl=""
    fi
    if [ -n "$tpl" ]; then
        # le gabarit porte « sim sap » : seul le MS#1 de --sap le garde
        # (delimiteur # : le | de s|..| casserait l alternance \| de sed)
        local _sim_sed=(-e "s#^\([[:space:]]*\)sim \(sap\|test\|reader\)\$#\1sim $(ms_sim_mode "$sapsock")#")
        sed \
            -e "s|bind 127.0.0.1 424[0-9]|bind 127.0.0.1 ${vty_port}|" \
            -e "s|layer2-socket /tmp/osmocom_l2[_0-9]*|layer2-socket ${l2sock}|" \
            -e "s|sap-socket /tmp/osmocom_sap[_0-9]*|sap-socket ${sapsock}|" \
            "${_sim_sed[@]}" \
            -e "s|stick [0-9]*|stick ${arfcn}|" \
            -e "s|^\([[:space:]]*\)imsi .*|\1imsi ${imsi}|" \
            -e "s|^\([[:space:]]*\)imei .*|\1imei $(rand_imei) 0|" \
            -e "s|^\([[:space:]]*\)ki .*|\1ki comp128 ${ki}|" \
            -e "s|alsa-output-dev .*|alsa-output-dev ${aout}|" \
            -e "s|alsa-input-dev .*|alsa-input-dev ${ain}|" \
            "$tpl" > "$dest"
    else
        # Template minimal de secours
        cat > "$dest" <<EOF
!
! mobile.cfg genere par start-native.sh (secours)
!
log stderr
 logging color 1
 logging print category 1
 logging timestamp 0
!
line vty
 no login
 bind 127.0.0.1 ${vty_port}
!
ms 1
 layer2-socket ${l2sock}
 sap-socket ${sapsock}
 sim $(ms_sim_mode "$sapsock")
 imsi ${imsi}
 ki comp128 ${ki}
 network-selection-mode auto
 stick ${arfcn}
!
EOF
    fi
```

```
    _ms_cfg_garantir_tch "$dest" "$aout" "$ain"
    _ms_cfg_garantir_log "$dest"
}

# ── LE BLOC tch-voice, GARANTI, QUEL QUE SOIT LE GABARIT ──────────────────
# [2026-09-09] MS#2 SORTAIT SANS « io-tch-format rtp ». Le gabarit retenu par
# la boucle ci-dessus est le PREMIER trouve, et c'est presque toujours
# qosmo/cfgs/mobile_group1.cfg - un fichier d'un autre depot, que
# update.sh refetche, et qui ne porte pas cette ligne. Le sed de generation ne
# fait que SUBSTITUER : ce qui manque au gabarit manque au resultat. MS#1
# l'avait (son fichier est ecrit ailleurs, 3192 octets), MS#2 non (2423).
#
# Sans cette ligne, le mobile lit et ecrit ses trames TCH dans la disposition
# Texas Instruments des vrais Calypso, alors que pont.py les code avec
# libosmocoding, c'est-a-dire en disposition RTP/RFC3551 (33 octets ouverts
# par la signature 0xD). Les deux bouts ne se comprennent alors DANS AUCUN
# SENS, aucun compteur ne bronche, et le son sort robotise (cf. le commentaire
# de configs/mobile.cfg, mesure du 27/08 : anneau descendant « d2 2c fd a2 »
# contre anneau montant « 83 00 05 e1 ... ff ff ff ff », une trame a champs).
#
# On ne corrige donc pas le gabarit - il ne nous appartient pas - on garantit
# le resultat : les quatre reglages de tch-voice sont poses ici, et le bloc
# entier est cree s'il manque (le gabarit de secours ci-dessus n'en a aucun).
# MS_TCH_FORMAT pour un banc qui voudrait autre chose que « rtp ».
_ms_cfg_garantir_tch() {
    local dest="$1" aout="${2:-gsm_out}" ain="${3:-gsm_in}" fmt="${MS_TCH_FORMAT:-rtp}"
    [ -f "$dest" ] || return 0
    awk -v aout="$aout" -v ain="$ain" -v fmt="$fmt" '
        BEGIN { vu = 0; dedans = 0 }
        # Le bloc existe : on pose nos quatre reglages en tete, une seule fois.
        /^[[:space:]]*tch-voice[[:space:]]*$/ && !vu {
            vu = 1; dedans = 1
            print " tch-voice"
            print "  io-handler gapk"
            print "  io-tch-format " fmt
            print "  alsa-output-dev " aout
            print "  alsa-input-dev " ain
            next
        }
        # Dans le bloc : on retire les anciennes copies de CES quatre lignes,
        # on garde tout le reste - les autres reglages comme les commentaires,
        # qui portent ici la mesure du 27/08 et valent mieux que le code.
        dedans {
            if ($0 ~
↪   /^[[:space:]][[:space:]]+(io-handler|io-tch-format|alsa-output-dev|alsa-input-dev)([[:space:]]|$)/) next
            if ($0 ~ /^[[:space:]][[:space:]]+[a-z]/ || $0 ~ /^[!#]/) { print; next }
            dedans = 0
        }
        # Pas de bloc du tout : on le pose juste avant « no shutdown ».
        /^[[:space:]]*no shutdown[[:space:]]*$/ && !vu {
            vu = 1
            print " tch-voice"
            print "  io-handler gapk"
            print "  io-tch-format " fmt
            print "  alsa-output-dev " aout
            print "  alsa-input-dev " ain
        }
        { print }
    ' "$dest" > "${dest}.tch" 2>/dev/null && mv -f "${dest}.tch" "$dest" || rm -f "${dest}.tch"
}
# ── LE BLOC log stderr, GARANTI, DE LA MEME FORME PARTOUT ─────────────────
# [2026-10-19] LE JOURNAL DU MOBILE N'AVAIT PAS LA MEME FORME SELON LE GABARIT.
# qosmo/cfgs/mobile_group1.cfg et c54x_exe/mobile_pont.cfg portent un « log
# stderr » (couleur, horodatage, categorie) ; configs/mobile.cfg n'en porte
# AUCUN - seul le « log gsmtap ». Le fichier pose dans /root/.osmocom/bb
# dependait donc du gabarit retenu en premier, et mobile.log sortait tantot
```

```
# avec ses couleurs et son horodatage a 17 chiffres (ce que l'encart
# osmo-fft-snap.py et rapport-run.sh savent lire), tantot avec les defauts de
# libosmocore plus le masque -d de qosmo. On pose ici LE bloc, une fois, quel
# que soit le gabarit : les sept categories que suit le banc en debug, couleur
# par sous-systeme, horodatage compact. ATTENTION aux deux horodatages de
# libosmocore 1.12 : « logging timestamp 1 » = ctime (« Mon Oct 19 10:10:03
# 2026 », 25 colonnes, constate le 19/10 sur le banc DSP) ; « logging print
# extended-timestamp 1 » = YYYYMMDDhhmmssnnn (17 chiffres, triable, et le seul
# que tail_lines() de l'encart sait retirer). C'est le second qu'on pose. Le
# bloc gsmtap, lui, n'est pas touche.
MS_LOG_STDERR_BLOC=' logging filter all 1
 logging color 1
 logging timestamp 0
 logging print extended-timestamp 1
 logging print category 1
 logging level mm debug
 logging level rr debug
 logging level cc debug
 logging level sms debug
 logging level plmn debug
 logging level sim debug
 logging level sap debug'
_ms_cfg_garantir_log() {
    local dest="$1" existe=0
    [ -f "$dest" ] || return 0
    # On sait AVANT de lire si le bloc existe : dans mobile_group1.cfg, « line
    # vty » precede « log stderr », et une seule passe inserait un second bloc
    # devant line vty avant de tomber sur le premier (constate au test).
    grep -q '^log stderr[[:space:]]*$' "$dest" && existe=1
    awk -v bloc="$MS_LOG_STDERR_BLOC" -v existe="$existe" '
        BEGIN { vu = 0; dedans = 0 }
        # Le bloc existe : on le remplace en entier, une seule fois.
        /^log stderr[[:space:]]*$/ && !vu {
            vu = 1; dedans = 1
            print "log stderr"; print bloc
            next
        }
        dedans {
            if ($0 ~ /^[[:space:]]+[a-z]/) next     # ancienne ligne du bloc
            dedans = 0                               # « ! » ou autre : fin du bloc
        }
        # Pas de bloc : on le pose devant le premier « log gsmtap », sinon
        # devant « line vty ».
        /^(log gsmtap|line vty)([[:space:]]|$)/ && !existe && !vu {
            vu = 1
            print "log stderr"; print bloc; print "!"
        }
        { print }
        END { if (!vu) { print "!"; print "log stderr"; print bloc; print "!" } }
    ' "$dest" > "${dest}.log" 2>/dev/null && mv -f "${dest}.log" "$dest" || rm -f "${dest}.log"
}
# ⚠ [2026-08-08, resolu le 26/08] LE RUN N'UTILISE PAS mobile.cfg POUR MS#1.
# Le mobile Calypso tourne avec
#     mobile -c /root/.osmocom/bb/mobile_group1.cfg
# alors qu'on ne generait ici que mobile.cfg. mobile_group1.cfg, lui, est COPIE
# tel quel par qosmo (run_modules/20-mobile-cfg.sh, depuis $QEMU_TREE/cfgs) :
# un fichier statique, jamais derive de l'operateur. Il porte donc l'identite de
# l'operateur 1 - IMSI 001010001000001, ARFCN 514 - quel que soit l'operateur.
#
# Constate sur le banc, operateur 2 : le HLR attend 002010002000001, la BTS
# emet sur 516, et le mobile se presentait avec 001010001000001 sur 514.
#     Mobile Subscriber of MS '1':
#      IMSI: 001010001000001
#      Status: U2_NOT_UPDATED  IMSI detached  LAI: invalid
#      Registered PLMN: MCC-MNC 001-01
# Tout le reste etait juste - clefs, MSISDN, point codes, ARFCN des BTS - et
```

```
# l'abonne restait detache, parce que la seule carte SIM qui compte n'etait
# generee nulle part.
#
# On ecrit donc MS#1 dans LES DEUX fichiers. mobile.cfg reste la reference
# (d'autres modules la lisent), mobile_group1.cfg est celui que le lanceur
# ouvre. Verifier lequel tourne, avant de croire un reglage pose ici :
#     pgrep -a mobile
# ── IMSI et Ki des MS : le plan du HLR, pas celui de l'operateur 1 ──────────
# Ces valeurs etaient ecrites EN DUR sur le PLMN de l'operateur 1 (001-01) et
# sur son numero d'abonne. Sur l'operateur 2, dont le reseau diffuse un autre
# MCC, le mobile presentait donc l'IMSI d'un abonne d'un autre pays : ni le MSC
# ni le HLR n'ont de fiche a ce numero, le Location Updating reste sans reponse,
# et l'on cherche la panne dans la BTS alors que la pile entiere est debout.
# Le plan est celui de start-docker.sh (section "HLR subscribers") :
#     IMSI = MCC MNC <operateur sur 4 chiffres> <index du MS sur 6 chiffres>
#     Ki   = 00112233445566778899aabbccdd <index du MS> <operateur>
# MCC et MNC sont relus dans la configuration REELLEMENT deployee - c'est elle
# que la BTS diffuse - et non redevines a partir d'une convention.
MS_OP_ID="${OPERATOR_ID:-1}"
MS_MCC="$(awk '$1=="network" && $2=="country" && $3=="code" {print $4; exit}' /etc/osmocom/osmo-bsc.cfg
 ↪  2>/dev/null || true)"
MS_MNC="$(awk '$1=="mobile" && $2=="network" && $3=="code" {print $4; exit}' /etc/osmocom/osmo-bsc.cfg
 ↪  2>/dev/null || true)"
# ── LES REPLIS SUIVENT LE PLAN, ILS NE L INVENTENT PAS ──────────────────────
# [2026-08-31] Ces deux lignes se declenchent quand osmo-bsc.cfg n est pas
# encore lisible (configs pas regenerees a cet instant), et elles etaient
# fausses toutes les deux :
#
#   MS_MNC="01"  en dur : l operateur 2 heritait du MNC de l operateur 1, et
#   ms_imsi() produisait 001|01|0002|000001 - MNC d un operateur, identifiant
#   d un autre. Ces abonnes-la arrivaient dans le HLR SANS MSISDN ni IMEI
#   ("none" / "-------------"), a cote des bons : le VLR ne pouvait les
#   rattacher a personne. Rien ne signalait l incoherence.
#
#   MS_MCC=<op sur 3 chiffres> : aurait donne MCC 002 pour l operateur 2,
#   alors que TOUT le banc est en 001 - c est le PLMN d essai, commun a tous
#   les operateurs, et c est le MNC qui les distingue.
#
# Le plan reel, verifie dans /var/tmp/osmo-annuaire.csv :
#   op 1 -> 001|01   op 2 -> 001|02   op 3 -> 001|03
# [2026-10-18] Au premier lancement d'une installation fraiche, osmo-bsc.cfg est
# encore le gabarit : on y lit les MARQUEURS __MCC__ / __MNC__, que la suite
# ecrivait tels quels dans les mobile_*.cfg (« imsi __MCC____MNC__0001000002 »,
# refuse par mobile : MS#2 ne demarrait pas) et exportait a run.sh. Seuls des
# chiffres comptent ; sinon, le repli ci-dessous.
[[ "$MS_MCC" =~ ^[0-9]{3}$ ]] || MS_MCC=""
[[ "$MS_MNC" =~ ^[0-9]{2,3}$ ]] || MS_MNC=""
[ -n "$MS_MCC" ] || MS_MCC="001"
[ -n "$MS_MNC" ] || MS_MNC="$(printf '%02d' "$MS_OP_ID")"
ms_imsi() { printf '%s%s%04d%06d' "$MS_MCC" "$MS_MNC" "$MS_OP_ID" "$1"; }

# ── LE QUATRIEME PROVISIONNEUR ──────────────────────────────────────────────
# run.sh embarque son propre module d'abonnes (run_modules/21-abonnes-hlr.sh,
# dans qosmo). Il calcule son IMSI comme MCC+MNC+operateur+rang - la meme
# formule que nous - mais lit MCC et MNC dans /etc/osmocom/osmo-msc.cfg via un
# chemin qui n'aboutit pas toujours ; il retombe alors sur ses defauts, 001 et
# 01. Sur l'operateur 2 il fabriquait donc 001010002000001 : un abonne que
# personne ne presente, avec en prime un MSISDN de l'ancien plan (20001), qui
# RETIENT ce numero et empeche le bon abonne de le recuperer.
#
# Il honore MCC et MNC depuis l'environnement. En les posant, son IMSI devient
# exactement celui que start-docker.sh a deja provisionne - et son propre controle
# d'etat (mod_abonnes_hlr_status) le trouve present, donc il ne fait rien.
# On neutralise ainsi un doublon par son idempotence, sans toucher a qosmo.
# [2026-10-24] SANS « ${MNC:-...} » : 03-affichage.sh a deja source globals.conf
# (MNC="01"), qui l emportait sur la config deployee - l op2 recreait
```

```
# 001010002000001 a chaque demarrage (sans auth, MSISDN refuse), visible en
# rouge dans les abonnes du conky. MS_MCC/MS_MNC SONT la config deployee.
export MCC="$MS_MCC"
export MNC="$MS_MNC"

# ── Le plan radio : on le LIT, on ne le recalcule pas ──────────────────
# generate_configs.sh l'ecrit dans /etc/osmocom/radio-plan.env en meme temps
# que les configurations. Le relire ici garantit que le mobile s'accorde sur
# ce que la BTS diffuse VRAIMENT - et non sur une formule recopiee qui finira
# par diverger, comme l'ont fait l'unit-id, l'ARFCN et l'IMSI avant elle.
# Les replis restent, pour un banc monte sans ce fichier.
if [ -r /etc/osmocom/radio-plan.env ]; then
    # shellcheck disable=SC1091
    . /etc/osmocom/radio-plan.env
    printf '  %splan radio%s  lu dans /etc/osmocom/radio-plan.env\n' "${C_DIM:-}" "${C_Z:-}"
fi

# ── Les deux canaux radio, et a qui chaque MS s'accroche ───────────────
# MS#1 et MS#2 avaient leur ARFCN ecrit EN DUR, 514 et 516 : les valeurs de
# l'operateur 1, et de lui seul. Sur l'operateur 2 la BTS principale passe a
# 516 et le side-car y restait aussi : deux cellules sur la meme frequence,
# leurs bursts melanges dans un milieu ou fake_trx apparie par frequence.
# Le mobile envoyait ses RACH, aucune Immediate Assignment ne revenait, et
# MS#1 restait accroche a 514 ou plus rien n'emettait.
#
# MS#1 suit la BTS principale, dont l'ARFCN est lu dans la configuration
# REELLEMENT deployee - c'est le BSC qui en decide et qui l'impose a la BTS
# par OML, donc c'est la qu'est la verite.
MS_ARFCN1="${PLAN_ARFCN:-}"
[ -n "$MS_ARFCN1" ] || MS_ARFCN1="$(awk '/^ bts 0$/{b=1} b && /^ *arfcn /{print $2; exit}'
 ↪  /etc/osmocom/osmo-bsc.cfg 2>/dev/null || true)"
# Meme garde que pour MCC/MNC : un gabarit pas encore regenere donne __ARFCN__.
[[ "$MS_ARFCN1" =~ ^[0-9]+$ ]] || MS_ARFCN1=""
[ -n "$MS_ARFCN1" ] || MS_ARFCN1=$(( 512 + MS_OP_ID * 2 ))

# MS#2 suit le side-car. Celui-la, c'est NOUS qui le decidons : le module
# 13-sidecar-cfg.sh de qosmo lit SC_ARFCN et SC_UNIT_ID dans
# l'environnement (run_modules/_lib/radio.sh : « : "${SC_ARFCN:=516}" »).
# Les poser ici evite de modifier qosmo, et garde une seule source par
# valeur - les memes formules que generate_configs.sh.
export SC_ARFCN="${SC_ARFCN:-${PLAN_ARFCN_BTS1:-$(( 612 + MS_OP_ID * 2 ))}}"
export SC_UNIT_ID="${SC_UNIT_ID:-${PLAN_UNIT_ID_BTS1:-$(( 6000 + MS_OP_ID * 10 + 2 ))}}"
MS_ARFCN2="$SC_ARFCN"
ms_ki()   { printf '00 11 22 33 44 55 66 77 88 99 aa bb cc dd %02x %02x' "$1" "$MS_OP_ID"; }

# IMEI tire au hasard, cle de Luhn calculee - un par MS, jamais deux identiques
# dans une meme trace.
rand_imei() {
    local body d i sum=0 dbl parity
    body="35892500$(printf '%06d' "$(( (RANDOM << 15 | RANDOM) % 1000000 ))")"
    for (( i = ${#body} - 1; i >= 0; i-- )); do
        d=${body:i:1}
        parity=$(( (${#body} - 1 - i) % 2 ))
        if [ "$parity" -eq 0 ]; then
            dbl=$(( d * 2 )); [ "$dbl" -gt 9 ] && dbl=$(( dbl - 9 ))
            sum=$(( sum + dbl ))
        else
            sum=$(( sum + d ))
        fi
    done
    printf '%s%d' "$body" "$(( (10 - sum % 10) % 10 ))"
}
```

3.376 /opt/GSM/osmo-operator/startup_scripts/07-plan-mobile.sh

511 octets, 7 lignes → 7 lignes

```
# 07-plan-mobile.sh - partie de start.sh (lignes 1512-1516 de start-native.sh d'origine)
# Sourcee par start.sh dans le shell principal : meme ordre, memes variables.
printf '  %sPLMN MS%s    MCC %s  MNC %s  operateur %s  ->  IMSI %s / %s\n' \
    "${C_DIM:-}" "${C_Z:-}" "$MS_MCC" "$MS_MNC" "$MS_OP_ID" "$(ms_imsi 1)" "$(ms_imsi 2)"
printf '  %sradio%s      MS#1 sur ARFCN %s (BTS principale)   MS#2 sur ARFCN %s (side-car, unit-id %s)\n' \
    "${C_DIM:-}" "${C_Z:-}" "$MS_ARFCN1" "$MS_ARFCN2" "$SC_UNIT_ID"
```

3.377 /opt/GSM/osmo-operator/startup_scripts/08-mobile-ms1.sh

1499 octets, 27 lignes → 27 lignes

```
# 08-mobile-ms1.sh - partie de start.sh (lignes 1517-1541 de start-native.sh d'origine)
# Sourcee par start.sh dans le shell principal : meme ordre, memes variables.
say_begin "Generation mobile MS#1"
MS1_CFG="${MOBILE_CFG_MS1_PATH:-$BB_DIR/mobile.cfg}"
generate_mobile_cfg "$MS1_CFG" \
    4247 \
    /tmp/osmocom_l2 \
    "${SAP_SOCK:-/tmp/osmocom_sap}" \
    "$MS_ARFCN1" \
    "$(ms_imsi 1)" \
    "$(ms_ki 1)"
# La copie que le lanceur ouvre reellement. On la REGENERE plutot que de la
# copier : generate_mobile_cfg fixe aussi le port VTY, les sockets et l'ARFCN,
# et un simple cp propagerait le fichier statique de qosmo.
# ── ET ON EMPECHE QU'IL SOIT ECRASE ──────────────────────────────
# Ecrire ce fichier ne suffit pas : run.sh le REECRIT juste apres. Son module
# 20-mobile-cfg.sh copie MOBILE_CFG_SRC par-dessus, et ce chemin vaut par
# defaut $QEMU_TREE/cfgs/mobile_group1.cfg - le fichier statique de qosmo,
# fige sur l'operateur 1. Mesure sur le banc :
#     mobile.cfg          18:22:22  imsi=002010002000001 arfcn=516
#     mobile_group1.cfg   18:22:27  imsi=001010001000001 arfcn=514
# Cinq secondes d'ecart, et c'est le second que le mobile ouvre.
# On designe donc NOTRE fichier comme source : le module recopie alors le bon
# contenu, et son controle d'idempotence (cmp -s src dest) passe puisque les
# deux sont identiques. Aucune modification de qosmo.
export MOBILE_CFG_SRC="$MS1_CFG"
```

3.378 /opt/GSM/osmo-operator/startup_scripts/09-mobile-cfg-tmux.sh

22838 octets, 384 lignes → 384 lignes

```
# 09-mobile-cfg-tmux.sh - partie de start.sh (lignes 1542-1911 de start-native.sh d'origine)
# Sourcee par start.sh dans le shell principal : meme ordre, memes variables.
MS1_GROUP_CFG="$BB_DIR/mobile_group1.cfg"
if [ "$MS1_GROUP_CFG" != "$MS1_CFG" ]; then
    generate_mobile_cfg "$MS1_GROUP_CFG" \
        4247 \
        /tmp/osmocom_l2 \
        "${SAP_SOCK:-/tmp/osmocom_sap}" \
        "$MS_ARFCN1" \
        "$(ms_imsi 1)" \
        "$(ms_ki 1)"
    printf '  %sMS#1%s       aussi ecrit dans %s (le fichier que run.sh ouvre)\n' \
        "${C_DIM:-}" "${C_Z:-}" "$MS1_GROUP_CFG"
fi
say_end " OK " "$C_OK" "Generation mobile MS#1" "$MS1_CFG"
```

```
say_begin "Generation mobile MS#2 (faketrx)"
MS2_CFG="$BB_DIR/mobile_faketrx_bts1.cfg"
generate_mobile_cfg "$MS2_CFG" \
    4248 \
    /tmp/ms2_l2 \
    /tmp/ms2_sap \
    "$MS_ARFCN2" \
    "$(ms_imsi 2)" \
    "$(ms_ki 2)" \
    gsm_out gsm_in
say_end " OK " "$C_OK" "Generation mobile MS#2 (faketrx)" "$MS2_CFG"
# ── LA SIM EMULEE LIT LE MEME FICHIER QUE LE MOBILE ──────────────────────
# CALYPSO_SIM_CFG est la SEULE variable d'environnement que lit le modele QEMU
# (hw/arm/calypso/calypso_sim.c) : il y prend la section « test-sim » - imsi et
# ki comp128 - au format de mobile.cfg, donc exactement le fichier MS#1 genere
# ci-dessus. Sans elle, la SIM emulee garde son IMSI par defaut
# (001010000000001), inconnu du HLR provisionne avec le plan. Elle n'etait
# exportee nulle part.
# (MOBILE_CFG_MS1/MS2 et CALYPSO_MS2_CFG, exportes ici jusqu'au 2026-09-02,
#  n'avaient aucun lecteur dans qosmo ; le side-car MS#2 lit
#  SC_MOBILE_CFG / le fichier lui-meme.)
export CALYPSO_SIM_CFG="$MS1_CFG"

# --- Mode PONT TRX (CALYPSO_BRIDGE=pont) : le pont maison est le transceiver --
# Le pont (pont/pont.py) se presente comme transceiver TRX-UDP a osmo-bts-trx
# (5700/5701/5702), decode les bursts DL en L2 -> GSMTAP 4730/4731 vers le
# modele QEMU, et encode l'UL depuis les sidebands /dev/shm/calypso_* -> TRXD
# -> BTS. Il a BESOIN d'osmo-bts-trx en face : c'est qosmo/60-bts.sh qui
# le lance, et la barriere plus bas attend son retour avant de lancer le pont.
if [ "${CALYPSO_BRIDGE:-}" = pont ]; then
    export CALYPSO_BRIDGE
    # ── LE PONT VIENT DU DEPOT, POINT ──────────────────────────────────────
    # Le defaut etait /opt/GSM/pont/pont.py, un chemin HORS DEPOT alimente par un
    # `COPY pont/pont.py` de Dockerfile.run et fige par un `ENV PONT_PY=`. Rien
    # ne le gardait en phase avec pont/pont.py, et rien ne signalait sa derive.
    #
    # CE QUE CA A COUTE, le 2026-08-27 : la copie executee etait restee a une
    # version anterieure de 198 lignes, SANS AUCUN plan SDCCH/8 (ni PLAN_SD8, ni
    # _dcch_plan, ni sacch_dl102). Elle raisonnait sur le SDCCH/4 combine du TS0
    # pendant que le mobile parlait sur SDCCH/8 SS0 du TS1. Pendant deux heures,
    # sans un seul message d erreur : 79 % de CRC fail, montant emis dans la
    # fenetre du /4 (donc hors de celle que la BTS ecoute, avec hors_fenetre=0
    # puisque coherent avec sa propre table), SABM sans reponse, T200 x6, LU en
    # echec, SMS rejete, appel impossible. Restaurer la copie du depot a suffi a
    # retablir le LU complet, chiffrement A5/1 compris, des le premier essai.
    #
    # La copie libre et son COPY sont supprimes ; PONT_PY reste disponible pour
    # qui veut essayer une variante, mais c est alors un choix explicite.
    #
    # [2026-08-30] NITB_ROOT RETIRE DE CE CHEMIN. Le defaut passait par lui, et
    # NITB_ROOT n appartient pas a ce depot : qosmo/environnement/paths.env
    # le pose a "$GSM_ROOT/osmo_egprs" - l ANCIEN nom de cet arbre. Le pont etait
    # donc cherche dans /opt/GSM/osmo_egprs/pont/pont.py, d ou
    # "pont introuvable (/opt/GSM/osmo_egprs/pont/pont.py)" sur un banc ou le
    # fichier est pourtant la, dans le depot. Pire quand ce vieux dossier existe
    # encore : on y trouve une COPIE, et on retombe mot pour mot dans la panne
    # decrite ci-dessus - un pont hors depot qui derive sans que rien ne le dise.
    # $HERE est le repertoire de CE script : le pont voyage avec lui, toujours.
    # [2026-09-23] Montage grgsm : son pont est pont.py (pont_dsp.py est
    # celui du montage DSP, lance par c54x_exe/run.sh). Voir pont/pont.py.
    _PONT="${PONT_PY:-$HERE/pont/pont.py}"

    # ── LE PONT DOIT SAVOIR SUR QUELLE CELLULE IL TRAVAILLE ────────────────
    # pont.py fait le codage/decodage de canal entre le Calypso de QEMU et la
    # BTS. Il lisait PONT_ARFCN et PONT_BSIC dans son environnement, avec pour
    # defauts 514 et 7 - les valeurs de l'operateur 1. Personne ne les posait.
```

```
#
# Le BSIC n'est pas decoratif : sa partie basse (BCC) EST la sequence
# d'apprentissage des canaux dedies. Avec un BSIC faux, la correlation
# echoue et les blocs ne se decodent pas. Constate sur l'operateur 2, dont
# la cellule est en ARFCN 516 / BSIC 14 :
#     dl_bursts=20660 dl_dec=1180 crc_fail=3984
#     TS0/22 : 0/295        <- la voie ou arrive l'Immediate Assignment
#     ul_sent=23 rach=23    <- le RACH partait bien
# Le mobile emettait son RACH, le reseau repondait, et la reponse n'etait
# jamais rendue au Calypso : "radio resource layer state: connection
# pending", indefiniment. L'operateur 1 s'en sortait parce que ses valeurs
# sont justement celles des defauts.
#
# On les prend dans le plan radio publie par generate_configs.sh, avec les
# memes replis que les mobiles.
export PONT_ARFCN="${PONT_ARFCN:-${PLAN_ARFCN:-$MS_ARFCN1}}"
export PONT_BSIC="${PONT_BSIC:-${PLAN_BSIC:-$(( MS_OP_ID * 7 % 64 ))}}"
printf '  %spont TRX%s : cellule ARFCN %s BSIC %s\n' \
    "${C_DIM:-}" "${C_Z:-}" "$PONT_ARFCN" "$PONT_BSIC"
# [2026-08-16] N'ARMER LE LANCEUR QUE SUR UN VRAI DEMARRAGE.
# Ce bloc s'execute AVANT le `case "$ACTION"` plus bas. Sur `--stop` (comme
# sur --list/--status/--check-paths) on armait donc quand meme le lanceur
# differe, que `run.sh --stop` tuait aussitot - d'ou le "line 450: Killed"
# crache par bash en plein arret propre. Un arret qui programme un demarrage
# n'a aucun sens, et le message faisait croire a une panne.
# Le pont DEJA VIVANT, lui, est bien arrete : le teardown le connait
# desormais (`f:pont/pont.py` dans _td_patterns) et mod_teardown_stop
# l'applique aussi.
if [ "$ACTION" != start ]; then
    printf '  %spont TRX%s : non arme (action=%s, pas un demarrage)\n' \
           "${C_DIM:-}" "${C_Z:-}" "$ACTION"
elif [ -r "$_PONT" ]; then
    # SINGLETON : tuer un pont precedent, sinon il tient 5700-5702 et le
    # teardown de run.sh echoue ("restes du run precedent : port:5700...").
    pkill -f "$_PONT" 2>/dev/null || true
    # [2026-08-16] KILLALL DEMANDE PAR L'OPERATEUR, et assume.
    # Le motif nominatif ci-dessus rate tout pont lance sous un autre chemin
    # (PONT_PY surcharge, une sauvegarde .bak), et c'est ce qui faisait
    # echouer le teardown sur "port:5700 port:5701 port:5702".
    # ⚠ CE QUE CA EMPORTE, en plus du pont : fake_trx.py (le transceiver de
    # MS#2), sms-interop-relay.py (routage SMS inter-operateurs, HORS pile
    # run.sh donc non relance automatiquement), gsm_sniff.py et les scripts
    # des panes tmux. C'est acceptable ICI parce qu'on est sur le chemin d'un
    # DEMARRAGE (ACTION=start) : le teardown de run.sh va de toute facon tout
    # arreter deux secondes plus tard. Ne PAS reprendre ce geste dans un
    # module de run.sh, ou il tuerait le filtre d'horodatage du run en cours.
    # CALYPSO_NO_KILLALL=1 pour s'en passer.
    if [ "${CALYPSO_NO_KILLALL:-0}" != "1" ]; then
        printf '  %spont TRX%s : python3 du banc termines (fake_trx, pont, airmesh, relais SMS, modem AT)\n'
        ↪  "${C_DIM:-}" "${C_Z:-}"
        killall_python -9 || true
    fi
    sleep 1
    printf '  %spont TRX%s : transceiver TRX-UDP d osmo-bts-trx (pont/pont.py)\n' "${C_DIM:-}" "${C_Z:-}"
    # LANCEMENT DIFFERE : run.sh fait d'abord son teardown (qui verifie que
    # 5700-5702 sont LIBRES). On ne binde donc qu'apres, sinon on se
    # bloque nous-memes. 25 s = teardown + demarrage des modules.
    # LOG HORS LOG_DIR : run.sh archive/efface LOG_DIR pendant son teardown ;
    # un pont.log ouvert avant se retrouve sur un inode SUPPRIME (sortie
    # invisible, "(deleted)" dans /proc/<pid>/fd/1). /dev/shm survit.
    # setsid : le pont doit survivre a la fin du shell de start-native.
    # [2026-08-16] LE `sleep 25` ETAIT UNE COURSE, PAS UNE BARRIERE.
    # 25 s etait un pari sur la duree du teardown de run.sh. Des que
    # celui-ci archive de gros journaux (mesure : 62 Mo -> ~40 s), le pont
    # se rebindait sur 5700-5702 EN PLEIN teardown, qui echouait alors sur
    # "restes du run precedent : port:5700 port:5701 port:5702" et
```

```
        # abandonnait toute la sequence. Le symptome se deplacait avec la
        # taille des journaux, ce qui le faisait passer pour intermittent.
        # On attend maintenant un EVENEMENT et non une duree : osmo-bts-trx,
        # qui est l'interlocuteur TRX-UDP du pont et ne demarre qu'APRES le
        # teardown. Sa presence prouve donc que les ports sont a nous.
        # [2026-08-27] LA BARRIERE ATTEND D'ABORD QU'IL DISPARAISSE.
        # Elle n'attendait que sa PRESENCE, en se fiant a « osmo-bts-trx ne
        # demarre qu'APRES le teardown ». Vrai seulement si l'on a fait --stop
        # avant. Sans --stop, l'osmo-bts-trx du run PRECEDENT est encore la :
        # pgrep reussit immediatement (_n=0), le pont binde 5700-5702 en plein
        # teardown, run.sh le tue derriere (f:pont/pont.py est dans
        # _td_patterns) et rien ne le relance -- « le pont n'a pas de bursts ».
        # On attend donc les DEUX fronts : la disparition (le teardown a bien
        # eu lieu), puis l'apparition (la pile est remontee). Avec --stop le
        # premier front est deja franchi et rien ne change.
        ( _n=0
          while [ "$_n" -lt 180 ] && pgrep -x osmo-bts-trx >/dev/null 2>&1; do
              _n=$((_n + 1)); sleep 1
          done
          _dispar=$_n
          while [ "$_n" -lt 180 ] && ! pgrep -x osmo-bts-trx >/dev/null 2>&1; do
              _n=$((_n + 1)); sleep 1
          done
          if [ "$_n" -ge 180 ]; then
              echo "[pont] osmo-bts-trx jamais (re)vu apres 180 s - demarrage quand meme"
          else
              echo "[pont] osmo-bts-trx parti a ${_dispar}s puis revu a ${_n}s - la pile est debout, on binde"
          fi
          sleep 1
          pkill -f "$_PONT" 2>/dev/null; sleep 1
          exec setsid python3 -u "$_PONT" ) >/dev/shm/pont.log 2>&1 &
        disown $! 2>/dev/null || true
        printf '  %spont: journal%s /dev/shm/pont.log (hors LOG_DIR, efface par le teardown)\n' "${C_DIM:-}"
         ↪  "${C_Z:-}"
    else
        printf '  pont introuvable (%s)\n' "$_PONT"
    fi
fi

# --- MAILLAGE DE BURSTS (--air-mesh) : le milieu radio devient commun ---------
# Jusqu'ici deux noeuds se parlaient tout en haut de la pile - un trunk SIP,
# un relais SMS, du M3UA vers un hub - et chacun gardait sa radio hermetique :
# le mobile de l'operateur 1 ne POUVAIT PAS entendre la BTS de l'operateur 2.
# Ici c'est le burst qui traverse : pont/airmesh.py prend ce qui passe sur le
# fake_trx local et l'echange avec les pairs. Voir l'en-tete d'airmesh.py pour
# le detail - notamment pourquoi il faut DEUX emplacements de transceiver, et
# comment on les obtient sans modifier une ligne de qosmo.
if [ "${AIR_MESH:-0}" = "1" ]; then
    _AIRMESH="${AIRMESH_PY:-${HERE}/pont/airmesh.py}"
    [ -f "$_AIRMESH" ] || _AIRMESH=/opt/GSM/osmo-operator/pont/airmesh.py
    if [ ! -f "$_AIRMESH" ]; then
        printf '  %sair-mesh%s : airmesh.py introuvable - maillage de bursts non lance\n' "${C_KO:-}" "${C_Z:-}"
    else
        # DEUX MOBILES DE PLUS QUE LE BANC N'EN A. fake_trx cree un transceiver
        # par mobile declare ; airmesh prend les deux derniers et les accorde
        # autrement (l'un comme un mobile, l'autre comme une BTS). Sans ce +2,
        # il n'y a aucun emplacement libre et airmesh se tait sans rien dire.
        _NMS_REEL="${N_MS:-2}"
        N_MS=$(( _NMS_REEL + 2 )); export N_MS
        _BBP="${BB_BASE_PORT:-6700}"; _BBS="${BB_PORT_STEP:-3}"
        _TRX_MS=$((  _BBP + (_NMS_REEL + 1 - 1) * _BBS ))   # emplacement N+1
        _TRX_BTS=$(( _BBP + (_NMS_REEL + 2 - 1) * _BBS ))   # emplacement N+2

        # Les pairs viennent de la table WAN : tout noeud qui n'est pas nous.
        _PAIRS=""
        for _tok in ${WAN_NODES:-}; do
```

```
            _pid="${_tok%%:*}"; _rest="${_tok#*:}"; _pip="${_rest%%:*}"
            [ -n "$_pid" ] && [ -n "$_pip" ] || continue
            [ "$_pid" = "${NODE_ID:-1}" ] && continue
            _PAIRS="$_PAIRS --pair ${_pid}:${_pip}:${AIRMESH_PORT:-4739}"
        done

        printf '  %sair-mesh%s   noeud %s  arfcn %s  emplacements TRXD %s et %s  pairs:%s\n' \
            "${C_DIM:-}" "${C_Z:-}" "${NODE_ID:-1}" "${ARFCN:-1}" \
            "$_TRX_MS" "$_TRX_BTS" "${_PAIRS:- aucun}"

        # On attend que fake_trx ait BINDE l'emplacement, pas une duree : il ne
        # demarre qu'apres le teardown de run.sh, dont la duree depend de la
        # taille des journaux archives. Une temporisation ferait de ce lancement
        # une course - c'est la lecon du pont, juste au-dessus.
        ( _n=0
          while [ "$_n" -lt 180 ] && ! ss -uln 2>/dev/null | grep -q ":$(( _TRX_MS + 1 ))\b"; do
              _n=$((_n + 1)); sleep 1
          done
          if [ "$_n" -ge 180 ]; then
              echo "[airmesh] emplacement $(( _TRX_MS + 1 )) jamais ouvert par fake_trx apres 180 s"
              echo "[airmesh] verifiez que le banc tourne bien avec N_MS=${N_MS} (deux de plus que ${_NMS_REEL})"
              exit 1
          fi
          echo "[airmesh] fake_trx a ouvert nos emplacements apres ${_n} s"
          sleep 1
          # shellcheck disable=SC2086
          exec setsid python3 -u "$_AIRMESH" \
              --node "${NODE_ID:-1}" --arfcn "${ARFCN:-1}" \
              --port "${AIRMESH_PORT:-4739}" \
              --trx-ms "$_TRX_MS" --trx-bts "$_TRX_BTS" \
              --topologie "${HERE}/data/air-mesh.txt" \
              $_PAIRS ) >/dev/shm/airmesh.log 2>&1 &
        printf '  %sair-mesh: journal%s /dev/shm/airmesh.log\n' "${C_DIM:-}" "${C_Z:-}"
    fi
fi

# --- 6. resume ---------------------------------------------------------------
banner
printf '  %sprofil%s     %s (%s)\n' "$C_DIM" "$C_Z" "$CALYPSO_PROFILE" "$MODE"
printf '  %senviron.%s   %s\n' "$C_DIM" "$C_Z" \
    "$(case "$RUNTIME_ENV" in docker) echo "conteneur docker" ;; vm) echo "machine virtuelle" ;; *) echo "machine
    ↪  physique" ;; esac)"
if [ -n "$NODE_ID" ]; then
    printf '  %snoeud%s      %s  %s(%s)%s\n' "$C_DIM" "$C_Z" "$NODE_ID" "$C_DIM" "${NODE_ID_SRC:-?}" "$C_Z"
else
    printf '  %snoeud%s      %s\n' "$C_DIM" "$C_Z" "aucun - identite SS7 inchangee"
fi
printf '  %srun.sh%s     %s\n' "$C_DIM" "$C_Z" "$RUN_SH"
[ -z "${FIRMWARE_ELF:-}" ] || printf '  %sfirmware%s   %s%s\n' "$C_DIM" "$C_Z" "$FIRMWARE_ELF" \
    "${FIRMWARE_SRC:+ (FIRMWARE_SRC)}"
# [2026-09-22] EN --dsp, MS#1 N'EST PAS LE FICHIER GENERE.
# Le module `l2` du fork est retire du plan : /root/.osmocom/bb/mobile.cfg
# n'est ouvert par personne, et le mobile qui tourne est celui du banc DSP
# (mobile_pont.cfg, VTY 4347). Le resume annoncait le mauvais fichier ET le
# mauvais port -- on allait se connecter a une VTY qui n'existe pas.
if [ "$DSP_BANC" = 1 ] && [ "${OSMO_TRX:-0}" = 1 ]; then
    printf '  %sMS#1%s       non lance : le Calypso de QEMU est le TRX de la BTS#0 (--osmo-trx : trx.highram.elf +
    ↪  transceiver, %s)\n' \
        "$C_DIM" "$C_Z" "$(trx_ts_desc)"
elif [ "$DSP_BANC" = 1 ]; then
    _ms1_cfg="${MOBILE_CFG:-$(dirname "$BANC_DSP")/mobile_pont.cfg}"
    _ms1_vty="$(sed -n 's/^ *bind 127.0.0.1 \([0-9]*\).*/\1/p' "$_ms1_cfg" 2>/dev/null | head -1)"
    printf '  %sMS#1%s       %s  IMSI %s  ARFCN %s  VTY %s  %s(banc DSP)%s\n' \
        "$C_DIM" "$C_Z" "$_ms1_cfg" "$(ms_imsi 1)" "$MS_ARFCN1" "${_ms1_vty:-?}" "$C_DIM" "$C_Z"
    printf '  %s           %s non lance (module l2 retire du plan)%s\n' \
        "$C_DIM" "$MS1_CFG" "$C_Z"
```

```
else
    printf '  %sMS#1%s       %s  IMSI %s  ARFCN %s  VTY 4247\n' "$C_DIM" "$C_Z" "$MS1_CFG" "$(ms_imsi 1)"
     ↪  "$MS_ARFCN1"
fi
# [2026-10-12] LE PORT VTY DU MOBILE EST PUBLIE, PAS DEVINE. En --dsp le
# mobile ecoute sur 4347 (mobile_pont.cfg), sinon sur 4247 - et tout ce qui
# ouvre « la console du mobile » (launch.sh --vty, l encart osmo-panel.py, le
# tableau de bord osmo-egprs-web pour le Kc et la VTY bb1) tapait 4247 en
# dur : en DSP, connexion refusee. Le port est ecrit dans
# /run/osmo-banc/ms1-vty (lu par les outils de session) et pousse dans
# l environnement de systemd (OSMO_BB1_VTY_PORT, lu par server.js au
# redemarrage du tableau de bord, plus bas). Efface a l arret.
MS1_VTY_PORT="${_ms1_vty:-4247}"
# [2026-10-19] LE JOURNAL DU MOBILE EST PUBLIE DE LA MEME FACON. Il vit sous
# LOG_DIR (qosmo, run_modules/70-l2.sh : « mobile ... >>LOG_DIR/mobile.log »),
# et en --dsp sous le RUNDIR de c54x_exe (run.sh : RUNDIR/mobile.log, /tmp/
# c54x-pont par defaut). L encart osmo-fft-snap.py tapait /run/user/0/... en
# dur : LOG_DIR suit l uid qui lance (XDG_RUNTIME_DIR), et le cadre restait sur
# « pas encore de journal ». Le chemin reel est ecrit dans
# /run/osmo-banc/mobile-log et pousse dans l environnement de systemd
# (OSMO_MOBILE_LOG). Efface a l arret, avec ms1-vty.
# [2026-10-23] --bts --ms / --ms-magnetite : le telephone est l INSTANCE 2 de run.sh (14-demarrage.sh,
# INSTANCE=2 AIR=1), son RUNDIR est /tmp/c54x-pont2 ; /tmp/c54x-pont est celui du TRX, sans mobile.
_ms1_rundir="${RUNDIR:-/tmp/c54x-pont}"
[ "${OSMO_TRX:-0}" = 1 ] && [ "${MS_AIR:-0}" = 1 ] && _ms1_rundir="${RUNDIR:-/tmp/c54x-pont2}"
MS1_LOG_NOM="mobile.log"
if [ "$DSP_BANC" = 1 ] && [ -n "${MS_MAGNETITE:-}" ]; then
    # [2026-10-22] Magnetite : pas de mobile L23, le journal du telephone est la trace L1/pile TI de rvinterf
    MS1_LOG="$_ms1_rundir/rvinterf.log"
    MS1_LOG_NOM="rvinterf"
elif [ "$DSP_BANC" = 1 ]; then
    MS1_LOG="$_ms1_rundir/mobile.log"
else
    MS1_LOG="$LOG_DIR/mobile.log"
fi
# [2026-10-24] exporte : la fenetre ms1 de tmux (qosmo tmux_modules/bench.sh) suit le journal du telephone
# (rvinterf en Magnetite) au lieu d'un mobile.log vide
export MS1_LOG MS1_LOG_NOM
if [ "$ACTION" = "start" ] && [ "$DRY" -ne 1 ]; then
    mkdir -p /run/osmo-banc 2>/dev/null && chmod 755 /run/osmo-banc 2>/dev/null
    printf '%s\n' "$MS1_VTY_PORT" > /run/osmo-banc/ms1-vty 2>/dev/null && chmod 644 /run/osmo-banc/ms1-vty
     ↪  2>/dev/null
    printf '%s\n' "$MS1_LOG" > /run/osmo-banc/mobile-log 2>/dev/null && chmod 644 /run/osmo-banc/mobile-log
     ↪  2>/dev/null
    command -v systemctl >/dev/null 2>&1 && systemctl set-environment "OSMO_BB1_VTY_PORT=$MS1_VTY_PORT"
     ↪  "OSMO_MOBILE_LOG=$MS1_LOG" 2>/dev/null || true
fi
printf '  %sMS#2%s       %s  IMSI %s  ARFCN %s  VTY 4248\n' "$C_DIM" "$C_Z" "$MS2_CFG" "$(ms_imsi 2)" "$MS_ARFCN2"
printf '  %sjournaux%s   %s\n' "$C_DIM" "$C_Z" "$LOG_DIR"
printf '  %s%-10s%s %s\n' "$C_DIM" "$MS1_LOG_NOM" "$C_Z" "$MS1_LOG"
printf '  %schiffrement%s %s\n' "$C_DIM" "$C_Z" "$ENCRYPTION"
printf '\n'
# --- 7. actions deleguees a run.sh ------------------------------------------
RUN_ARGS=()
[ "$DRY" -eq 1 ]     && RUN_ARGS+=(--dry-run)
[ "$FORCE" -eq 1 ]   && RUN_ARGS+=(--force)
[ "$VERBOSE" -eq 1 ] && RUN_ARGS+=(--verbose)
RUN_ARGS+=(--profile "$CALYPSO_PROFILE")
# [2026-08-08] --restart etait ajoute DIRECTEMENT sur la ligne exec, hors de
# RUN_ARGS. Consequence : `--dry-run` affichait une commande SANS --restart puis
# le vrai lancement en ajoutait un. Une simulation qui ne decrit pas ce qui va
# se passer est pire qu'absente. Il entre donc dans RUN_ARGS, ou il est visible.
# CALYPSO_NO_RESTART=1 pour demarrer sans reinitialiser (l'ancien comportement
# n'etait pas atteignable du tout).
[ "${CALYPSO_NO_RESTART:-0}" = "1" ] || RUN_ARGS+=(--restart)
# run.sh lit deja CALYPSO_NO_ATTACH (run.sh:8), mais une variable ne se voit
```

```
# pas dans --dry-run ni dans la ligne affichee juste avant l exec : l option
# porte le meme reglage la ou on le cherche.
[ "${CALYPSO_NO_ATTACH:-0}" = "1" ] && RUN_ARGS+=(--no-attach)
# [2026-10-19] --icount et --assembly-logs vers run.sh, donc vers 40-qemu.sh
# (chemin --grgsm ; en --dsp le module qemu est retire, c54x_exe/run.sh lit
# QEMU_ICOUNT et ASSEMBLY_LOGS lui-meme). Memes -d / -dfilter que le banc DSP.
case "${QEMU_ICOUNT:-}" in ''|0|off) ;; *) RUN_ARGS+=("--icount=$QEMU_ICOUNT") ;; esac
if [ "${ASSEMBLY_LOGS:-0}" = 1 ]; then
    RUN_ARGS+=(--asm)
    # [2026-10-19] in_asm seul par defaut : exec,nochain (une ligne par bloc
    # execute, 494 Mo en 90 s) rendait l'ARM 150 fois plus lent et cassait
    # l'attach, voir l'aide de --assembly-logs.
    export QEMU_ASM_FLAGS="${ASSEMBLY_LOGS_FLAGS:-in_asm}"
    [ -n "${ASSEMBLY_LOGS_FILTRE:-}" ] && export QEMU_ASM_FILTRE="$ASSEMBLY_LOGS_FILTRE"
fi
# ── Les sessions tmux d'un run precedent ──────────────────────────────
# Le teardown de run.sh arrete les processus mais laisse la session tmux
# debout. Au demarrage suivant, son module de panes RECREE ses fenetres dans la
# session survivante : il y a alors DEUX fenetres nommees « all ».
#
# Et deux fenetres de meme nom ne sont pas seulement inesthetiques : tmux ne
# sait plus resoudre la cible. Verifie sur le banc, tmux 3.2a :
#     $ tmux split-window -t calypso:all
#     can't find window: all
# Les trois splits echouent, le module les compte comme "pane refuse par tmux
# (place insuffisante)" - un diagnostic FAUX qui envoie chercher un probleme de
# taille de terminal - et la barriere conclut « la fenetre "all" n'a pas garde
# ses panes ». La cause etait deux fenetres plus haut.
#
# On efface donc les sessions du run precedent avec lui. Trois conventions
# cohabitent selon le lanceur, on les couvre toutes ; aucune n'a besoin
# d'exister.
purge_sessions_tmux() {
    local n=0 s
    for s in calypso gapk osmo; do
        if tmux has-session -t "$s" 2>/dev/null; then
            tmux kill-session -t "$s" 2>/dev/null && n=$((n + 1))
        fi
    done
    if tmux -S /tmp/osmocom_tmux has-session -t osmocom 2>/dev/null; then
        tmux -S /tmp/osmocom_tmux kill-session -t osmocom 2>/dev/null && n=$((n + 1))
    fi
    [ "$n" -gt 0 ] && printf '  %ssessions tmux%s   %s session(s) du run precedent effacee(s)\n' \
        "${C_DIM:-}" "${C_Z:-}" "$n"
    return 0
}
```

3.379 /opt/GSM/osmo-operator/startup_scripts/10-identite-noeud.sh

12751 octets, 217 lignes → 217 lignes

```
# 10-identite-noeud.sh - partie de start.sh (lignes 1912-2122 de start-native.sh d'origine)
# Sourcee par start.sh dans le shell principal : meme ordre, memes variables.
case "$ACTION" in
    list)
        if [ "$DSP_BANC" = 1 ]; then
            printf '  %sbanc DSP%s   %s retire du plan : %s ; %s les reprend (5 etapes)\n' \
                "$C_DIM" "$C_Z" "$RUN_SH" "$DSP_MODULES_RETIRES" "$BANC_DSP"
        fi
        # [2026-10-18] Sans --restart : run.sh garde la DERNIERE action lue, et
        # RUN_ARGS finit par --restart. `--list` arretait donc la pile en cours
        # et la relancait au lieu d'afficher le plan.
        _list_args=()
        for _a in "${RUN_ARGS[@]}"; do [ "$_a" = "--restart" ] || _list_args+=("$_a"); done
```

```
    exec env CALYPSO_PROFILE="$CALYPSO_PROFILE" bash "$RUN_SH" --list "${_list_args[@]}"
    ;;
genconf)
    # ── LA FICHE DU NOEUD, SANS LANCER ──────────────────────────────
    # [2026-09-04] La meme que celle ecrite par le run (juste avant run.sh) :
    # identite resolue comme au lancement (--node, environnement,
    # /etc/osmo-role, table WAN), table WAN relue si elle existe.
    . "$HERE/network/wan-nodes.sh"; . "$HERE/network/node-conf.sh"
    _gc_node="$NODE_ID"
    if [ -z "$_gc_node" ]; then
        _gc="$(detect_node_id)"; _gc_node="${_gc%%|*}"
    fi
    _gc_node="${_gc_node:-1}"
    wan_nodes_load "${WAN_CONF_FILE:-/etc/osmo-wan.conf}" 2>/dev/null || true
    _gc_hub="${HUB_IP:-${OSMO_HUB_IP:-}}"
    [ -n "$_gc_hub" ] || _gc_hub="$(awk -F= '/^OSMO_HUB_IP=/{print $2}' /etc/osmo-role 2>/dev/null | tr -d '
    ↪  \r')"
    _gc_out="${GEN_CONF_OUT:-$(node_conf_path "$_gc_node")}"
    if node_conf_write "$_gc_out" "$_gc_node" "${NODE_OP:-1}" "$_gc_hub"; then
        say_end " OK " "$C_OK" "Fiche du noeud $_gc_node" "$_gc_out"
        printf '  %ssur un autre noeud :%s  ./start-native.sh --wan %s\n' "${C_DIM:-}" "${C_Z:-}" "$(basename
        ↪  "$_gc_out")"
        exit 0
    fi
    say_end " KO " "$C_KO" "Fiche du noeud" "ecriture impossible : $_gc_out"; exit 1
    ;;
stop)
    rm -f /run/osmo-banc/ms1-vty /run/osmo-banc/mobile-log 2>/dev/null
    command -v systemctl >/dev/null 2>&1 && systemctl unset-environment OSMO_BB1_VTY_PORT OSMO_MOBILE_LOG
    ↪  2>/dev/null || true
    say_begin "Arret de la pile via run.sh"
    banc_dsp_arreter          # QEMU, osmocon, mobile, pont : a lui
    banc_dsp_arreter_instances
    bash "$RUN_SH" --stop --profile "$CALYPSO_PROFILE"
    # les serveurs SAP de softSIM (--sap) et leur socket
    for _p in $(ps -eo pid,args | awk '/[d]emo_server\.rb/ {print $1}'); do kill -TERM "$_p" 2>/dev/null ||
    ↪  true; done
    rm -f "${SAP_SOCK:-/tmp/osmocom_sap}" 2>/dev/null; unset _p
    say_end " OK " "$C_OK" "Arret de la pile via run.sh"
    purge_sessions_tmux
    # ── LE TABLEAU DE BORD S ARRETE AVEC LE BANC ────────────────────
    # [2026-09-04] Il survivait a tous les arrets : ni run.sh --stop ni la
    # purge tmux ne le connaissent, son unite est WantedBy=multi-user.target
    # et killall_python ne voit pas un serveur node. On se retrouvait avec
    # :8080 qui repond DEVANT UNE PILE MORTE - l etat le plus trompeur qui
    # soit, puisqu il se lit comme "le banc tourne".
    # PartOf=osmo-banc.service couvre `systemctl stop|restart osmo-banc` ;
    # ceci couvre le geste direct, par lequel passent le bouton « Arreter »
    # du bureau, launch.sh --stop et start-multi.sh --stop.
    # --no-block : on n attend pas systemd, et surtout on ne peut pas se
    # bloquer quand cet arret est lui-meme l ExecStop de osmo-banc.
    # [2026-10-22] PLUS PAR DEFAUT : le tableau de bord (:8080) reste en ligne banc arrete (demande
    # explicite) ; CALYPSO_DASHBOARD_SUIT_BANC=1 retablit l'arret avec le banc.
    if [ "${CALYPSO_DASHBOARD_SUIT_BANC:-0}" != 1 ]; then
        :
    elif [ -d /run/systemd/system ] && command -v systemctl >/dev/null 2>&1; then
        systemctl --no-block stop osmo-egprs-web.service 2>/dev/null || true
    else
        pkill -f "osmo-egprs-web/server.js" 2>/dev/null || true
    fi
    # Filet identique a celui de run.sh, pose ici aussi : on peut arreter par
    # ce script sans passer par l'autre, et un fake_trx.py orphelin suffit a
    # faire echouer le run suivant sur « port UDP 5720 deja pris ».
    # run.sh le fait deja de son cote ; le repeter ne coute rien et couvre le
    # cas ou RUN_SH est introuvable ou sort en erreur.
    # PORTEE : les python3 DU BANC seulement (voir killall_python).
```

```
        # CALYPSO_STOP_KILL_PYTHON=0 le desactive.
        if [ "${CALYPSO_STOP_KILL_PYTHON:-1}" != 0 ]; then
            if killall_python -9; then
                say_end " OK " "$C_OK" "python3 du banc termines"
            fi
        fi
        exit 0
        ;;
    status)
        if [ "$DSP_BANC" = 1 ]; then
            printf '\n  %sbanc DSP%s  %s\n' "$C_DIM" "$C_Z" "$BANC_DSP"
            banc_dsp --status
            printf '\n'
        fi
        exec env CALYPSO_PROFILE="$CALYPSO_PROFILE" bash "$RUN_SH" --status --profile "$CALYPSO_PROFILE"
        ;;
    checkpaths)
        # Le banc DSP n'a pas de --check-paths : ses cinq etapes verifient
        # chacune son binaire et sortent en nommant ce qui manque. On montre
        # donc ce qu'elles vont chercher, plutot que d'inventer une reponse.
        if [ "$DSP_BANC" = 1 ]; then
            printf '  %sbanc DSP%s   %s\n' "$C_DIM" "$C_Z" "$BANC_DSP"
            for f in "$(dirname "$BANC_DSP")/c54x_exe" "$GSM_ROOT/qosmo/build/qemu-system-arm" \
                     "${FIRMWARE_ELF:-$GSM_ROOT/firmware/board/compal_e88/$([ "${OSMO_TRX:-0}" = 1 ] && echo trx
                     ↪  || echo layer1).highram.elf}" \
                     "$GSM_ROOT/osmocom-bb/src/host/osmocon/osmocon" \
                     "$([ "${OSMO_TRX:-0}" = 1 ] && echo
                     ↪  "$GSM_ROOT/osmocom-bb-transceiver/src/host/layer23/src/transceiver/transceiver" || echo
                     ↪  "$HERE/pont/pont_dsp.py")"; do
                [ -e "$f" ] && printf '   OK   %s\n' "$f" || printf '   KO   %s (absent)\n' "$f"
            done
            printf '\n'
        fi
        exec env CALYPSO_PROFILE="$CALYPSO_PROFILE" bash "$RUN_SH" --check-paths
        ;;
esac
# --- 7bis-menu. Ce qui manque, demande UNE fois, avec le cache en defaut ------
# Les trois valeurs qui decident de l'identite SS7 - numero de noeud, operateur,
# adresse du hub - se deduisent la plupart du temps : environnement transmis par
# start-docker.sh, /etc/osmo-role fige dans l'image, table WAN. Quand la deduction ne
# donne rien, la pile demarrait quand meme et gardait les point codes du
# gabarit : 1.1.2 partout, ASP vers 127.0.0.1, interco morte sans un mot.
#
# On demande donc - mais seulement ce qui manque, et en proposant ce que le
# cache sait deja. Sans terminal, rien n'est demande : les defauts s'appliquent,
# une ISO qui demarre seule ne peut pas rester bloquee sur une question.
ask_node_identity() {
    # NO_MENU=1 est ce que start-docker.sh pose devant le lanceur quand les choix ont
    # deja ete faits sur l'hote. Aujourd'hui la garde tty juste en dessous
    # suffit (start-docker.sh lance par docker exec -d, sans terminal) : ce test ne
    # repare donc aucune panne observable, il aligne l'identite SS7 sur la
    # meme convention que les menus de pile de start-docker.sh, pour le jour ou le
    # lanceur automatique aura un terminal.
    [ "${NO_MENU:-0}" = "1" ] && return 0
    [ -t 0 ] || return 0
    [ "$DRY" -eq 1 ] && return 0

    local cached_node cached_src cached_hub
    IFS='|' read -r cached_node cached_src <<< "$(detect_node_id)"

    # L'ENVIRONNEMENT NE SE DEMANDE PAS. C'est ainsi que start-docker.sh transmet le
    # noeud a un conteneur : la reponse est deja connue, et la question n'a pas
    # de sens - pire, dans un conteneur ou personne ne lit le terminal, elle
    # repart vide. L'identite n'etait alors PAS appliquee, la pile regenerait
    # ses configs par defaut (point-code 1.1.2, ASP vers le hub docker, en
    # shutdown) et ecrasait celle que start-docker.sh venait d'ecrire. Le fichier
```

```
    # portait les bonnes valeurs juste avant, les mauvaises juste apres.
    # Le rattrapage qui se trouvait ici ne servait plus a rien : il exigeait un
    # NODE_ID vide, alors que la resolution du noeud, bien plus haut, l'a deja
    # rempli depuis cette meme source. NODE_ID et HUB_IP sont donc repris de
    # l'environnement la-bas, une fois, et pas ici.
    cached_hub="$(awk -F= '/^OSMO_HUB_IP=/{gsub(/[ \r\t]/,"",$2);v=$2} END{print v}' /etc/osmo-role 2>/dev/null)"
    [ -n "$cached_hub" ] || cached_hub="172.20.0.10"

    if command -v whiptail >/dev/null 2>&1; then
        [ -n "$NODE_ID" ] || NODE_ID="$(whiptail --title "Identite du noeud" \
            --inputbox "Numero de ce noeud (1-9)${cached_src:+   [$cached_src]}\nVide = ne pas toucher a
            ↪  l'identite SS7." \
            11 66 "${cached_node:-}" 3>&1 1>&2 2>&3)" || NODE_ID=""
        if [ -n "$NODE_ID" ] && [ -z "$HUB_IP" ]; then
            HUB_IP="$(whiptail --title "Inter-STP" \
                --inputbox "Adresse du hub SS7 (PC 0.0.0)\nVide = laisser le remote-ip deja present." \
                11 66 "$cached_hub" 3>&1 1>&2 2>&3)" || HUB_IP=""
        fi
    else
        if [ -z "$NODE_ID" ]; then
            printf '  Numero de ce noeud (1-9)%s [%s] : ' \
                "${cached_src:+ - $cached_src}" "${cached_node:-aucun}"
            read -r NODE_ID || NODE_ID=""
            [ -n "$NODE_ID" ] || NODE_ID="${cached_node:-}"
        fi
        if [ -n "$NODE_ID" ] && [ -z "$HUB_IP" ]; then
            printf '  Adresse de l inter-STP [%s] : ' "$cached_hub"
            read -r HUB_IP || HUB_IP=""
            [ -n "$HUB_IP" ] || HUB_IP="$cached_hub"
        fi
    fi
}
ask_node_identity

# == La regeneration des configurations, POUR DE VRAI ========================
# CE QUE --regen FAISAIT JUSQU'ICI : RIEN.
# Il levait OSMO_NO_REGEN et comptait sur run_modules/08-gabarits.sh pour
# rejouer apply_config_templates au demarrage. Ce module N'EXISTE PAS - il n'a
# jamais existe dans run_modules/ - et run.sh ne genere aucune configuration.
# Le message « regeneration demandee » s'affichait, la pile s'arretait, et pas
# un octet de /etc/osmocom ne changeait. Le seul effet reel de l'option etait
# d'arreter le banc.
#
# CE QU'IL FAIT MAINTENANT : la meme chose que start-docker.sh cote docker, et que
# build-iso.sh a la construction - c'est-a-dire la sequence complete :
#
#   1. apply_config_templates      substitue configs/*.cfg et *.conf dans un
#                                  repertoire NEUF (jamais en place : une
#                                  substitution a moitie faite sur /etc/osmocom
#                                  laisse un banc qui ne demarre plus) ;
#   2. apply_native_post_patches   la recette NATIVE de l'ISO - sms-routing par
#                                  MS reel, SGSN/MSC sur l'adresse locale du
#                                  coeur, run.sh degraisse ;
#   3. install_configs_native      pose le resultat dans /etc/osmocom et
#                                  /etc/asterisk.
#
# EN SOUS-SHELL. lib/gabarits.sh charge globals.conf et pose une trentaine de
# variables - MCC, ARFCN, HOST_IP, N_OPERATORS, KI... - qui portent ici un tout
# autre sens (HOST_IP est l'adresse du banc pour le lanceur, l'adresse interne
# du coeur pour les gabarits). Les laisser fuir, c'est changer le plan radio
# d'un run parce qu'on a demande une regeneration.
#
#   $1 numero de noeud   $2 numero d'operateur   $3 inter-STP (peut etre vide)
# -- LE POINT CODE AFFICHE SUIT LE PLAN REELLEMENT APPLIQUE ------------------
# [2026-08-31] Ces bandeaux calculaient toujours 1.<noeud><op>.x, la formule du
# plan WAN, alors que set-node-id.sh recoit --local hors WAN et ecrit
```

```
# 1.<op>.x. On lisait donc, dans la meme sortie :
#     [ OK ] Identite SS7 (noeud 1 · PC 1.13.x · mode docker)
#       SS7  point-code 1.3.2
# deux valeurs contradictoires pour la meme chose - la premiere fausse. Sur un
# banc ou le desaccord de plan a justement coute une demi-journee, un bandeau
# qui annonce le mauvais plan est exactement ce qu il ne faut pas.
_pc_affiche() {   # $1 = noeud, $2 = operateur
    if [ "${WAN_MESH:-0}" != "1" ] && [ -z "${WAN_NODES:-}" ]; then
        printf '1.%s.x' "$2"          # plan local, celui qu on applique
    else
        printf '1.%s%s.x' "$1" "$2"   # plan WAN, entre machines
    fi
}
```

3.380 /opt/GSM/osmo-operator/startup_scripts/11-regen-configs.sh

15185 octets, 270 lignes → 270 lignes

```
# 11-regen-configs.sh - partie de start.sh (lignes 2123-2390 de start-native.sh d'origine)
# Sourcee par start.sh dans le shell principal : meme ordre, memes variables.
regen_configs_natif() {
    local node="$1" op="$2" hub="$3"
    local tmp rc=0

    # ── Le plan NATIF, celui de build-iso.sh, et non celui de docker ─────────
    # 127.0.0.2 (__CONTAINER_IP__) porte le coeur : GGSN, SGSN, UPF, la nsvc du
    # BSC. Une adresse de boucle locale existe TOUJOURS et est prete avant tout
    # service ; une adresse de carte peut manquer au moment ou osmo-ggsn se lie,
    # et le bind echoue sans rapport visible avec le reseau.
    # 127.0.0.1 (__GATEWAY_IP__) recoit le gsmtap, la ou tshark capte deja.
    local core_ip="${OSMO_NATIVE_CORE_IP:-127.0.0.2}"
    local gw_ip="${OSMO_NATIVE_GW_IP:-127.0.0.1}"

    # ── L'identite SUIT LE NOEUD ─────────────────────────────────────────────
    # 1.<noeud><op>.<role>, le plan de network/set-node-id.sh et de build-iso.sh
    # --node. Sans le noeud dans le point code, trois machines attachees au meme
    # hub y presentent trois fois 1.11.2 : un point code est une ADRESSE, et
    # deux equipements qui partagent la leur donnent du routage faux, silencieux.
    # Le routing context suit la meme regle : c'est lui que le hub lit pour
    # distinguer les AS, et la formule locale (op*100+50) rend la meme valeur
    # sur tous les noeuds.
    # Plan WAN seulement s'il y a un WAN (--wan, --wan-nodes). Un hote seul garde
    # le plan LOCAL 1.<op>.<role>, comme set-node-id.sh --local au demarrage :
    # sinon la regeneration reecrit 1.<noeud><op> a chaque relance, meme sans WAN.
    local mid="${node}${op}" wan_plan=0
    if [ "${WAN_MESH:-0}" = "1" ] || [ -n "${WAN_NODES:-}" ]; then wan_plan=1; else mid="$op"; fi
    local pc_msc="1.${mid}.1" pc_stp="1.${mid}.2" pc_bsc="1.${mid}.3"
    local shut="shutdown"
    [ -n "$hub" ] && shut="no shutdown"

    # ── Le PLMN : celui de la machine, pas celui de la table ─────────────────
    # Relu dans osmo-msc.cfg installe. Reprendre les defauts de globals.conf
    # renommerait le reseau a chaque regeneration - OsmoDirect redevenait
    # OsmoQEMU - et un mobile deja provisionne ne reconnait plus sa cellule.
    local cfgdir="${OSMOCOM_CFG:-/etc/osmocom}"
    local mcc mnc opname
    mcc="$(awk '/^ *network country code /{print $4; exit}' "$cfgdir/osmo-msc.cfg" 2>/dev/null)"
    mnc="$(awk '/^ *mobile network code /{print $4; exit}'  "$cfgdir/osmo-msc.cfg" 2>/dev/null)"
    opname="$(awk '/^ *short name /{print $3; exit}'        "$cfgdir/osmo-msc.cfg" 2>/dev/null)"
    [ -n "$mcc" ]    || mcc="${MCC:-001}"
    [ -n "$mnc" ]    || mnc="${MNC:-01}"
    [ -n "$opname" ] || opname="${OP_NAME:-OsmoGSM}"

    # Le nombre de MS vient de coeur.env, ecrit par build-iso.sh : c'est lui qui
```

```
# dit combien d'abonnes le HLR provisionne, donc combien de routes SMS il
# faut. Au-dela, le relais rend "No route for destination".
local n_ms="${N_MS:-}"
if [ -z "$n_ms" ] && [ -r "$cfgdir/coeur.env" ]; then
    n_ms="$(awk -F'[:=}]' '/N_MS:=/{print $3; exit}' "$cfgdir/coeur.env" 2>/dev/null)"
fi
[ -n "$n_ms" ] || n_ms=2

printf '  %splan natif%s coeur %s  gsmtap %s  PC %s  rctx %s  PLMN %s-%s (%s)  %s MS\n' \
    "${C_DIM:-}" "${C_Z:-}" "$core_ip" "$gw_ip" "$pc_stp" \
    "$(( node * 1000 + op * 100 + 50 ))" "$mcc" "$mnc" "$opname" "$n_ms"

tmp="$(mktemp -d)" || return 1

# ── La sauvegarde, AVANT d'ecrire ──────────────────────────────────
# Une regeneration ecrase des fichiers que quelqu'un a pu retoucher a la
# main sur le banc. Qu'elle soit demandee ne veut pas dire qu'on accepte de
# perdre l'etat d'avant sans trace.
local sav="${LOG_DIR:-/tmp}/osmocom-avant-regen-$(date +%Y%m%d-%H%M%S).tar.gz"
tar czf "$sav" -C / etc/osmocom etc/asterisk 2>/dev/null && \
    printf '  %ssauvegarde%s %s\n' "${C_DIM:-}" "${C_Z:-}" "$sav"

(
    cd "$HERE" || exit 1
    # AVANT de sourcer. lib/gabarits.sh fige ENCRYPTION a "a5 0" et HOST_IP
    # a 127.0.0.1 par ":=" des qu'ils sont vides, et gc_load tient ensuite
    # ces valeurs pour un choix de l'appelant : le banc repartait en clair
    # apres un --regen, sans un mot.
    : "${ENCRYPTION:=a5 1}"
    HOST_IP="$core_ip"
    # Le routing context et l'adresse source de l'ASP : les deux surcharges
    # que build-iso.sh pose pour un noeud de WAN. 0.0.0.0 parce que
    # l'adresse du noeud vient du DHCP et n'est pas forcement montee quand
    # osmo-stp demarre - se lier a une adresse absente echoue au lancement.
    if [ "$wan_plan" = "1" ]; then
        export RCTX_INTER_OVERRIDE=$(( node * 1000 + op * 100 + 50 ))
    else
        export RCTX_INTER_OVERRIDE=$(( op * 100 + 50 ))
    fi
    export INTER_LOCAL_IP_OVERRIDE="0.0.0.0"
    # _apply_node_ss7_addressing (appele en fin de apply_config_templates)
    # ne rejoue set-node-id.sh que s'il sait QUI il sert, et il ne le lit
    # que la.
    export OSMO_ROLE=operator OSMO_WAN_NODE="$node"
    [ -n "$hub" ] && export OSMO_HUB_IP="$hub"
    unset OSMO_NO_REGEN

    # shellcheck source=lib/gabarits.sh
    . "$HERE/lib/gabarits.sh" || exit 1

    apply_config_templates "$tmp" \
        "$core_ip" "$gw_ip" \
        "$op" "$pc_msc" "$pc_stp" "$pc_bsc" \
        "$mcc" "$mnc" "$opname" \
        "${hub:-127.0.0.1}" "$shut" "1" || exit 1

    # Le noeud et la table WAN : c'est de la table que sms-routing.conf
    # tire l'adresse de chaque noeud, et du noeud qu'il tire les MSISDN
    # (<noeud>00<operateur><rang>).
    apply_native_post_patches "$tmp" "$op" "$n_ms" "$core_ip" \
        "$node" "${WAN_CONF_FILE:-/etc/osmo-wan.conf}" || exit 1

    # ── bb/ RETIRE, et ce n'est pas un oubli ──────────────────────
    # install_configs_native poserait $tmp/bb/mobile.cfg par-dessus
    # /root/.osmocom/bb/mobile.cfg. Or ce fichier a deja ete ECRIT plus haut
    # par generate_mobile_cfg, qui seul fixe le port VTY, les sockets L1CTL
```

```
        # et l'ARFCN de CHAQUE mobile. Le gabarit, lui, est fige sur l'operateur
        # 1 : le remettre ici rendait les deux mobiles au meme ARFCN et a la
        # meme VTY, et le second restait muet pour toujours.
        rm -rf "$tmp/bb"

        install_configs_native "$tmp" "" || exit 1
    ) || rc=1

    rm -rf "$tmp"
    return "$rc"
}

# ── Les gabarits : conserves, sauf --regen ──────────────────────────
# Par defaut on ne touche a rien : dans un conteneur, ce que start-docker.sh vient
# d'ecrire - point codes du noeud, inter-STP, ASP actif - ne doit pas etre
# remplace par les valeurs des gabarits, sans quoi l'identite SS7 est juste
# puis fausse 80 secondes plus tard, sans que rien ne l'annonce.
if [ "${REGEN_GABARITS:-0}" -eq 1 ]; then
    unset OSMO_NO_REGEN
    printf '  %sgabarits%s   regeneration demandee (--regen)\n' "${C_DIM:-}" "${C_Z:-}"

    # ── REGEN : ON ARRETE D ABORD ──────────────────────────────────
    # Regenerer /etc/osmocom sous une pile VIVANTE ne sert a rien et trompe :
    # les demons ont deja lu leur configuration, ils gardent l'ancienne en
    # memoire, et les fichiers neufs ne prennent effet qu'au demarrage suivant.
    # On lisait donc une conf juste sur le disque pendant que la pile tournait
    # sur la precedente -- le genre d'ecart qui se paie dix minutes plus tard.
    # Un --regen arrete donc la pile, puis regenere : l'etat est sans ambiguite.
    if [ -x "$RUN_SH" ] || [ -r "$RUN_SH" ]; then
        say_begin "Arret de la pile avant regeneration"
        banc_dsp_arreter >/dev/null 2>&1 || true
        bash "$RUN_SH" --stop --profile "$CALYPSO_PROFILE" >/dev/null 2>&1 || true
        say_end " OK " "$C_OK" "Arret de la pile avant regeneration"
        declare -F purge_sessions_tmux >/dev/null && purge_sessions_tmux
        # Meme filet qu'au --stop : un python3 orphelin (pont, fake_trx, trxcon)
        # garde son port et fait echouer le demarrage suivant sur « port UDP
        # 5720 deja pris ». PORTEE : les python3 du banc seulement.
        if [ "${CALYPSO_STOP_KILL_PYTHON:-1}" != 0 ]; then
            killall_python -TERM && \
                say_end " OK " "$C_OK" "python3 du banc termines"
        fi
    fi

    # ── LE NOEUD, PAR DEFAUT 1 ─────────────────────────────────────
    # Une regeneration produit une identite COMPLETE, jamais un plan « sans
    # noeud ». Sans numero, apply_config_templates rendait le plan du gabarit -
    # point-code 1.1.2 pour toute machine - et deux noeuds du meme banc se
    # retrouvaient a la meme adresse SS7. On numerote donc toujours : le noeud
    # resolu plus haut (--node, l'environnement, /etc/osmo-role, la table WAN),
    # et 1 quand il n'y en a aucun. Les configurations s'incrementent alors
    # avec lui : PC 1.<noeud><op>.<role>, routing context <noeud>*1000+..., et
    # le segment prive 192.168.<noeud+1>.x.
    REGEN_NODE="${NODE_ID:-1}"
    if ! [[ "$REGEN_NODE" =~ ^[1-9]$ ]]; then
        say_end " KO " "$C_KO" "--regen" "numero de noeud invalide : $REGEN_NODE (1 a 9)"
        exit 2
    fi
    REGEN_OP="${NODE_OP:-1}"
    if ! [[ "$REGEN_OP" =~ ^[1-9]$ ]]; then
        say_end " KO " "$C_KO" "--regen" "numero d'operateur invalide : $REGEN_OP (1 a 9)"
        exit 2
    fi
    # cached_hub est LOCAL a ask_node_identity : le lire ici rendait le vide et,
    # sans inter-STP, l'ASP repartait en shutdown apres chaque regeneration.
    REGEN_HUB="${HUB_IP:-${OSMO_HUB_IP:-}}"
    [ -n "$REGEN_HUB" ] || REGEN_HUB="$(awk -F= '/^OSMO_HUB_IP=/{gsub(/[ \r\t]/,"",$2);v=$2} END{print v}' \
```

```
        "${ROLE_FILE:-/etc/osmo-role}" 2>/dev/null)"

    say_begin "Regeneration des configurations (noeud $REGEN_NODE, operateur $REGEN_OP)"
    if [ "$DRY" -eq 1 ]; then
        say_end " -- " "$C_DIM" "Regeneration des configurations" "dry-run"
    elif regen_configs_natif "$REGEN_NODE" "$REGEN_OP" "$REGEN_HUB"; then
        say_end " OK " "$C_OK" "Regeneration des configurations" \
            "/etc/osmocom + /etc/asterisk  ·  PC $(_pc_affiche "$REGEN_NODE" "$REGEN_OP")"
    else
        say_end " KO " "$C_KO" "Regeneration des configurations" \
            "echec - les fichiers d'avant sont dans ${LOG_DIR:-/tmp}"
        exit 1
    fi

    # ── LE HLR SUIT LA NUMEROTATION ──────────────────────────────────────
    # Une regeneration change les MSISDN (<noeud>00<operateur><rang>) : ceux du
    # HLR datent du plan precedent. L'IMSI, lui, ne bouge pas - il ne porte pas
    # le noeud, precisement pour que le Ki et l'authentification restent
    # valables. L'etape d'abonnes ne peut donc pas se contenter de constater
    # que l'IMSI existe : elle relit desormais AUSSI le MSISDN
    # (run_modules/21-abonnes-hlr.sh, mod_abonnes_hlr_status) et rejoue
    # "subscriber ... update msisdn" quand il a change.
    #
    # On la force en plus ici : la pile vient d'etre arretee, le HLR repartira
    # avec run.sh, et l'etape doit s'executer meme si un cache d'etat la croit
    # deja faite. Sans ca, l'abonne s'attachait (IMSI connu) mais restait
    # injoignable, sans qu'une ligne ne dise que son numero avait change.
    FORCE=1
    printf '  %sHLR%s        abonnes a reapprovisionner : MSISDN %s..%s\n' \
        "${C_DIM:-}" "${C_Z:-}" \
        "$(( REGEN_NODE * 100000 + REGEN_OP * 100 + 1 ))" \
        "$(( REGEN_NODE * 100000 + REGEN_OP * 100 + ${N_MS:-2} ))"

    # Le noeud retenu vaut pour la suite du lancement : c'est celui qui vient
    # d'etre ecrit dans les fichiers. Le laisser vide ici, c'est laisser la
    # section « Identite SS7 » plus bas repartir sur une autre resolution et
    # reecrire par-dessus une identite differente de celle qu'on vient de poser.
    NODE_ID="$REGEN_NODE"
    NODE_OP="$REGEN_OP"
    [ -n "$REGEN_HUB" ] && HUB_IP="${HUB_IP:-$REGEN_HUB}"
    export OSMO_ROLE=operator
    export OSMO_WAN_NODE="$NODE_ID"
    [ -n "$HUB_IP" ] && export OSMO_HUB_IP="$HUB_IP"
elif [ "${RUNTIME_ENV:-native}" = "docker" ]; then
    # DANS UN CONTENEUR seulement. Les configs y sont deja ecrites par start-docker.sh,
    # avec l'identite du noeud : les regenerer les remplace par les valeurs des
    # gabarits, et l'interco meurt.
    export OSMO_NO_REGEN=1
else
    # SUR UNE VM sans --regen : on CONSERVE ce qui est en place. C'est l'ISO qui
    # a genere ces fichiers a la construction, avec la meme recette que
    # regen_configs_natif ; les refaire a chaque demarrage ne changerait rien et
    # ecraserait toute retouche faite sur le banc. --regen est la pour ca, et il
    # le dit.
    if [ -n "$NODE_ID" ]; then
        export OSMO_ROLE=operator
        export OSMO_WAN_NODE="$NODE_ID"
        [ -n "$HUB_IP" ] && export OSMO_HUB_IP="$HUB_IP"
    fi
fi

# --- 7ter. Identite de noeud (--node) ----------------------------------------
# AVANT le WAN et avant run.sh : les point codes sont lus par osmo-stp, osmo-msc
# et osmo-bsc a leur demarrage. Les changer apres ne servirait a rien.
#
# UNE SEULE SOURCE DE VERITE. start-docker.sh calcule deja l'identite d'un conteneur et
```

```
# la lui passe par l'environnement ; la recalculer ici en donnerait deux, et
# c'est la mauvaise qui gagnait. On reprend donc la sienne quand elle existe.
if [ -z "$NODE_ID" ] && [ -n "${OSMO_WAN_NODE:-${WAN_NODE_ID:-}}" ]; then
    NODE_ID="${OSMO_WAN_NODE:-$WAN_NODE_ID}"
    [ -n "$HUB_IP" ] || HUB_IP="${OSMO_HUB_IP:-}"
fi

# Sans identite alors que le WAN est demande, on ne part PAS en silence : la
# pile demarrerait avec les point codes du gabarit (1.1.2 pour tout le monde),
# deux noeuds porteraient la meme adresse SS7, et aucun ASP ne s'attacherait -
# sans qu'une seule ligne ne le signale.
# --wan FICHIER sans --node : ce noeud prend le premier numero que les fiches
# ne portent pas (les fiches disent qui est deja la, on dit qu'on l'a fait).
if [ -z "$NODE_ID" ] && [ "${#WAN_IMPORT_FILES[@]}" -gt 0 ]; then
    _used=" $(sed -n 's/^OSMO_NODE_ID=//p' "${WAN_IMPORT_FILES[@]}" 2>/dev/null | tr -d '" ' | tr '\n' ' ') "
    for _cand in 1 2 3 4 5 6 7 8 9; do
        case "$_used" in *" $_cand "*) ;; *) NODE_ID="$_cand"; break ;; esac
    done
    NODE_ID_SRC="premier numero libre d'apres les fiches"
    printf '  %sIdentite%s   noeud %s (%s ; --node N pour choisir)\n' "${C_DIM:-}" "${C_Z:-}" "$NODE_ID"
     ↪  "$NODE_ID_SRC"
fi
if [ -z "$NODE_ID" ] && [ "${WAN_MESH:-0}" -eq 1 ]; then
    say_end " KO " "$C_KO" "Identite SS7" "WAN demande mais aucun noeud : --node N, ou WAN_NODE_ID"
    exit 1
fi
```

3.381 /opt/GSM/osmo-operator/startup_scripts/12-pcap-softsim.sh

31519 octets, 555 lignes → 555 lignes

```
# 12-pcap-softsim.sh - partie de start.sh (lignes 2391-2943 de start-native.sh d'origine)
# Sourcee par start.sh dans le shell principal : meme ordre, memes variables.
if [ -n "$NODE_ID" ]; then
    if ! [[ "$NODE_ID" =~ ^[1-9]$ ]]; then
        say_end " KO " "$C_KO" "--node" "un chiffre de 1 a 9"; exit 2
    fi
    SETID="$HERE/network/set-node-id.sh"
    if [ ! -r "$SETID" ]; then
        say_end " KO " "$C_KO" "--node" "network/set-node-id.sh absent"; exit 1
    fi
    say_begin "Identite SS7 du noeud $NODE_ID"
    setid_args=(--node "$NODE_ID" --op "$NODE_OP" --mode "$NODE_MODE")
    # ── PLAN LOCAL HORS WAN ──────────────────────────────────────────
    # [2026-08-31] --local n etait JAMAIS passe. set-node-id.sh prenait donc
    # toujours le plan WAN, MID=<noeud><op> :
    #     noeud 1 / op 2 -> 1.12.2      noeud 1 / op 3 -> 1.13.2
    # alors que le hub et la topologie du banc attendent le plan LOCAL,
    # MID=<op> :
    #     op 2 -> 1.2.2                 op 3 -> 1.3.2
    # Les conteneurs s enregistraient donc en 1.12.2 / 1.13.2 (visible cote hub
    # sous as-rkm-1250 / as-rkm-1350, ACTIFS), pendant que les AS statiques
    # as-op2 / as-op3 attendaient en 1.2.2 / 1.3.2 et restaient AS_DOWN.
    # Consequence : tout le monde attache, personne routable - les routes vers
    # 1.2.2 et 1.3.2 sortaient PROHIB chez le natif. Rien ne signalait le
    # desaccord de plan : les deux cotes avaient raison, chacun dans le sien.
    #
    # Le plan WAN n a de sens qu ENTRE MACHINES. Sans WAN, une seule machine
    # porte tout : c est le plan local qui vaut, et c est deja celui du natif.
    if [ "${WAN_MESH:-0}" != "1" ] && [ -z "${WAN_NODES:-}" ]; then
        setid_args+=(--local)
    fi
    [ -n "$HUB_IP" ] && setid_args+=(--hub-ip "$HUB_IP")
    if [ "$DRY" -eq 1 ]; then
```

```
        say_end " -- " "$C_DIM" "Identite SS7 du noeud $NODE_ID" "dry-run"
        bash "$SETID" "${setid_args[@]}" --dry-run 2>&1 | sed 's/^/  /'
    elif bash "$SETID" "${setid_args[@]}" </dev/null > "${LOG_DIR:-/tmp}/set-node-id.log" 2>&1; then
        say_end " OK " "$C_OK" "Identite SS7" "noeud $NODE_ID · PC $(_pc_affiche "$NODE_ID" "$NODE_OP") · mode
        ↪  $NODE_MODE"
        # Ce qui a REELLEMENT ete ecrit, relu dans le fichier. Jusqu'ici il
        # fallait deduire l'identite d'un routing-key pour savoir si elle avait
        # pris - et une regeneration ulterieure pouvait l'effacer sans bruit.
        _pc_now="$(awk "/^cs7 instance/{c=1} c && /^ *point-code /{print \$2; exit}" \
                   "${OSMOCOM_CFG:-/etc/osmocom}/osmo-stp.cfg" 2>/dev/null)"
        _hub_now="$(awk "/asp asp-to-inter/{f=1} f&&/remote-ip/{print \$2; exit}" \
                   "${OSMOCOM_CFG:-/etc/osmocom}/osmo-stp.cfg" 2>/dev/null)"
        printf "  %sSS7%s        point-code %s   inter-STP %s\n" \
            "${C_DIM:-}" "${C_Z:-}" "${_pc_now:-?}" "${_hub_now:-?}"
    else
        say_end " KO " "$C_KO" "Identite SS7" "voir ${LOG_DIR:-/tmp}/set-node-id.log"
        exit 1
    fi
    # Le numero de noeud vaut aussi pour la voix et les SMS : c'est le meme noeud.
    WAN_NODE_ID="$NODE_ID"
fi

# --- 7bis. WAN a N noeuds ----------------------------------------------------
# ICI et pas apres : ce processus va devenir run.sh (exec), il n'y a pas d'apres.
# Asterisk n'est pas encore lance - setup-wan-mesh.sh le voit, ecrit la conf et
# ne redemarre rien ; run.sh demarrera Asterisk avec le WAN deja en place.
#
# Sur vis-a-vis des gabarits : en natif rien ne reinstalle /etc/asterisk au
# demarrage (install_configs_native n'a aucun appelant, run_modules/08-gabarits.sh
# n'existe pas), donc le bloc WAN ecrit ici survit a run.sh. Si un module de
# gabarits revient un jour, il faudra rejouer le WAN APRES lui.
if [ -r "${WAN_CONF_FILE:-/etc/osmo-wan.conf}" ] && [ "$WAN_MESH" -eq 0 ]; then
    # Table figee dans l'image (ISO construite avec --wan) : WAN_AUTO=1 dit
    # "ce systeme EST un noeud", on n'oblige pas a retaper l'option.
    if grep -q '^WAN_AUTO=1' "${WAN_CONF_FILE:-/etc/osmo-wan.conf}" 2>/dev/null; then
        WAN_MESH=1
        WAN_AUTO_KEEP=1
        printf '  %sWAN%s        table figee dans %s (WAN_AUTO=1)\n' \
            "$C_DIM" "$C_Z" "${WAN_CONF_FILE:-/etc/osmo-wan.conf}"
    fi
fi

# ── PAYLOAD ISM : le texte des SMS par radio, le reste par IP ──────────────
# [2026-10-14] N'intervient qu'au demarrage. Ecrit (si un backend est donne) la
# section [payload] des sms-routing.conf via network/setup-payload-ism.sh, puis
# exporte PAYLOAD_ISM : le relais SMS (scripts/sms-interop-relay.py) met alors un
# jeton dans le TCP inter-noeud et fait voyager le texte sur la radio ISM. Sans
# backend, on reutilise la section deja ecrite par setup-payload-ism.sh.
if [ "${PAYLOAD_ISM:-0}" -eq 1 ] && [ "$ACTION" = "start" ]; then
    PISM_SH="$HERE/network/setup-payload-ism.sh"
    export PAYLOAD_ISM=1
    if [ "$DRY" -eq 1 ]; then
        say_begin "Payload ISM"
        if [ -n "$PAYLOAD_ISM_LINK" ]; then
            printf '  [dry-run] %s --%s%s\n' "$PISM_SH" "$PAYLOAD_ISM_LINK" \
                "$([ "$PAYLOAD_ISM_LINK" = serial ] && echo " ${ISM_SERIAL:-/dev/ttyUSB0}")"
        else
            printf '  [dry-run] section [payload] deja en place (sms-routing.conf)\n'
        fi
        say_end " -- " "$C_DIM" "Payload ISM" "dry-run"
    elif [ -n "$PAYLOAD_ISM_LINK" ]; then
        say_begin "Payload ISM ($PAYLOAD_ISM_LINK)"
        pism_args=("--$PAYLOAD_ISM_LINK")
        case "$PAYLOAD_ISM_LINK" in
            serial) pism_args=(--serial "${ISM_SERIAL:-/dev/ttyUSB0}" --baud "${ISM_SERIAL_BAUD:-9600}") ;;
            fifo)   pism_args=(--fifo --tx "${ISM_TX:-/tmp/osmo_ism_tx}" --rx "${ISM_RX:-/tmp/osmo_ism_rx}") ;;
```

```
            exec)   pism_args=(--exec ${ISM_TX_CMD:+--tx-cmd "$ISM_TX_CMD"} ${ISM_RX_CMD:+--rx-cmd "$ISM_RX_CMD"})
                ;;
        esac
        if [ -r "$PISM_SH" ] && bash "$PISM_SH" "${pism_args[@]}" \
                > "${LOG_DIR:-/tmp}/payload-ism.log" 2>&1; then
            say_end " OK " "$C_OK" "Payload ISM" "texte des SMS par radio ($PAYLOAD_ISM_LINK), signalisation par IP"
        else
            say_end " KO " "$C_KO" "Payload ISM" "voir ${LOG_DIR:-/tmp}/payload-ism.log ; le relais retombera sur
                IP"
        fi
    else
        say_begin "Payload ISM"
        if grep -rqs '^\[payload\]' /etc/osmocom/sms-routing.conf 2>/dev/null; then
            say_end " OK " "$C_OK" "Payload ISM" "section [payload] en place (sms-routing.conf)"
        else
            say_end " -- " "$C_DIM" "Payload ISM" "aucune section [payload] : lance d'abord $PISM_SH
                --serial|--exec|..."
        fi
    fi
fi

# [2026-10-15, revue] --interstp ne vaut que dans le WAN : sans --wan, il
# etait ignore sans un mot.
if [ "$INTERSTP" = "1" ] && [ "$WAN_MESH" -ne 1 ] && [ "$ACTION" = "start" ]; then
    printf '  %s!%s --interstp sans --wan : pas de WAN, pas de hub lance (./start-native.sh --wan --interstp)\n'
        "${C_KO:-}" "${C_Z:-}" >&2
fi
if [ "$WAN_MESH" -eq 1 ] && [ "$ACTION" = "start" ]; then
    WAN_LIB="$HERE/network/wan-nodes.sh"
    WAN_MESH_SH="$HERE/network/setup-wan-mesh.sh"
    if [ ! -r "$WAN_LIB" ] || [ ! -r "$WAN_MESH_SH" ]; then
        say_end " KO " "$C_KO" "WAN" "network/wan-nodes.sh ou setup-wan-mesh.sh absent"
        exit 1
    fi
    # shellcheck source=network/wan-nodes.sh
    . "$WAN_LIB"
    export WAN_AUTO_SAVE="$WAN_AUTO_KEEP"
    # --regen-conf : on repart de zero. La table en place est mise de cote
    # (jamais perdue en silence) puis effacee, AVANT le moindre wan_nodes_load :
    # la suite du bloc la reconstruit depuis --wan-nodes, les fiches ou la
    # question, sans trainer les pairs eteints d'hier (routes PROHIB au STP).
    if [ "$REGEN_WAN_CONF" -eq 1 ]; then
        _wc="${WAN_CONF_FILE:-/etc/osmo-wan.conf}"
        # La sauvegarde reste A COTE de la table : /tmp (LOG_DIR par defaut)
        # ne survit pas a un redemarrage, encore moins sur l'ISO en RAM.
        _sav="${_wc}.avant-regen-$(date +%Y%m%d-%H%M%S)"
        if [ "$DRY" -eq 1 ]; then
            [ -f "$_wc" ] && printf '  [dry-run] cp -p %s %s && rm -f %s\n' "$_wc" "$_sav" "$_wc"
        elif [ -f "$_wc" ]; then
            # Pas de sauvegarde, pas d'effacement : la table n'est jamais perdue en silence.
            if cp -p "$_wc" "$_sav" 2>/dev/null; then
                printf '  %sWAN%s        table mise de cote : %s\n' "$C_DIM" "$C_Z" "$_sav"
                rm -f "$_wc"
            else
                say_end " KO " "$C_KO" "WAN" "--regen-conf : sauvegarde impossible ($_sav), table laissee en place"
                exit 1
            fi
        fi
        printf '  %sWAN%s        table reconstruite de zero (--regen-conf)\n' "$C_DIM" "$C_Z"
        unset _wc _sav
    fi
    # --virtualbox : segment host-only + VM + table, avant tout le reste.
    if [ "$VBOX_INTERCO" -eq 1 ] && [ -z "${WAN_NODES:-}" ]; then
        VB="$HERE/network/setup-vbox-interco.sh"
        [ -r "$VB" ] || { say_end " KO " "$C_KO" "WAN" "network/setup-vbox-interco.sh absent"; exit 1; }
        vb_args=(--host-node "$VBOX_HOST_NODE" --conf "${WAN_CONF_FILE:-/etc/osmo-wan.conf}")
```

```
    [ -n "$VBOX_NODES" ] && vb_args+=(--nodes "$VBOX_NODES")
    bash "$VB" "${vb_args[@]}" || exit 1
    wan_nodes_load "${WAN_CONF_FILE:-/etc/osmo-wan.conf}" || exit 1
    WAN_NODES="$(wan_nodes_spec)"
    WAN_NODE_ID="$VBOX_HOST_NODE"
fi

if [ "${#WAN_IMPORT_FILES[@]}" -gt 0 ]; then
    # ── --wan mynode1.conf : LA TABLE VIENT DES FICHES ──────────────────
    # [2026-09-04] Chaque ISO ecrit sa fiche au run (network/node-conf.sh) ;
    # on les importe par-dessus la table en place, on s'y ajoute avec
    # l'adresse locale, et le maillage habituel fait le reste. Pas de
    # question : tout ce qu'il fallait demander est dans les fiches.
    . "$HERE/network/node-conf.sh"
    wan_nodes_load "${WAN_CONF_FILE:-/etc/osmo-wan.conf}" 2>/dev/null || true
    printf '  %sWAN%s        fiches de noeuds :\n' "$C_DIM" "$C_Z"
    for _nf in "${WAN_IMPORT_FILES[@]}"; do node_conf_import "$_nf" || exit 1; done
    [ -n "$NODE_ID" ] || { say_end " KO " "$C_KO" "WAN" "--wan FICHIER exige de connaitre ce noeud (--node
     ↪  N)"; exit 1; }
    WAN_NODE_ID="$NODE_ID"
    node_conf_self "$NODE_ID" || exit 1
    WAN_NODES="$(wan_nodes_spec)"
    wan_nodes_parse "$WAN_NODES" || exit 1
elif [ -n "${WAN_NODES:-}" ]; then
    wan_nodes_parse "$WAN_NODES" || exit 1
    # --node DIT DEJA quel noeud est cette machine : c'est la meme notion
    # que --wan-id, vue depuis le SS7. S'en servir evite d'echouer sur une
    # detection par IP quand la table embarquee est plus ancienne que la
    # machine - le cas d'une ISO gravee avant le dernier plan d'adressage.
    if [ "${WAN_NODE_ID:-0}" = "0" ] && [ -n "$NODE_ID" ]; then
        WAN_NODE_ID="$NODE_ID"
        printf '  %sWAN%s        noeud %s (repris de --node)\n' "$C_DIM" "$C_Z" "$WAN_NODE_ID"
    fi
    [ "${WAN_NODE_ID:-0}" != "0" ] || wan_nodes_detect_self || {
        printf '  %sWAN%s : aucune IP locale dans la table - passez --wan-id N\n' "$C_KO" "$C_Z"
        printf '  %s      (ou --node N : le numero de noeud vaut pour les deux)%s\n' "$C_DIM" "$C_Z"
        exit 1; }
else
    wan_nodes_load "${WAN_CONF_FILE:-/etc/osmo-wan.conf}" 2>/dev/null || true
    # Une ISO diffusee sur N machines porte la MEME table : chaque machine
    # se reconnait a son IP plutot que d'exiger une image par noeud.
    wan_nodes_detect_self 2>/dev/null || true
    if [ "${WAN_NODE_COUNT:-0}" -lt 1 ] || [ "${WAN_NODE_ID:-0}" = "0" ]; then
        wan_nodes_prompt || exit 1
    fi
    # --node N : cette machine EST le noeud N. Sa ligne dans la table est
    # remise a l'adresse reelle, et la table reecrite - sans question.
    if [ -n "$NODE_ID" ] && [ "$WAN_MESH" -eq 1 ]; then
        wan_self_refresh "$NODE_ID" "${HUB_IP:-${OSMO_HUB_IP:-}}" || exit 1
    fi
fi
wan_nodes_validate || exit 1
printf '\n'
wan_nodes_summary
printf '\n'
if [ "$DRY" -eq 1 ]; then
    printf '  [dry-run] %s --native --id %s\n' "$WAN_MESH_SH" "$WAN_NODE_ID"
else
    say_begin "WAN - application de la table"
    # --native / --docker : meme distinction que pour les point codes.
    if bash "$WAN_MESH_SH" "--$NODE_MODE" --id "$WAN_NODE_ID" \
            --nodes "$(wan_nodes_spec)" --ops 1 \
            --config "${WAN_CONF_FILE:-/etc/osmo-wan.conf}" > "${LOG_DIR:-/tmp}/wan-mesh.log" 2>&1; then
        say_end " OK " "$C_OK" "WAN - noeud $WAN_NODE_ID, indicatif $(wan_local_ind)"
    else
        say_end " KO " "$C_KO" "WAN" "voir ${LOG_DIR:-/tmp}/wan-mesh.log"
```

```
        fi
    fi

    # --- 7quater. Inter-STP standalone (hub SS7 du WAN) -----------------------
    # Le hub est le seul point ou les noeuds se rejoignent en SS7 ; sans lui,
    # l'ASP de chaque STP reste en shutdown. Standalone : un osmo-stp a part,
    # avec sa propre config (start-interstp.sh), pas une pile du banc.
    # [2026-10-15, revue] Qui le lance, sans en lancer deux :
    #   --interstp       ce noeud, toujours ;
    #   auto (defaut)    seulement si l'adresse du hub est DEJA locale, ou si ce
    #                    noeud est le noeud 1 et que personne ne repond en M3UA
    #                    sur HUB_IP:2908. Deux noeuds qui demarrent ensemble ne
    #                    posent plus chacun l'adresse du hub (IP en double).
    #   --no-interstp    jamais.
    # « Present » se lit sur le port M3UA (SCTP 2908), pas au ping : un hote
    # quelconque a cette adresse, ou un hub arrete, repondaient au ping.
    INTERSTP_SH="$HERE/launch/start-interstp.sh"
    if [ -z "${HUB_IP:-}" ]; then
        HUB_IP="$(awk -F= '/^OSMO_HUB_IP=/{gsub(/[ \r\t]/,"",$2);v=$2} END{print v}' /etc/osmo-role 2>/dev/null)"
        # Le defaut LAN ne vaut que pour un noeud natif ; en conteneur le hub est
        # sur le backbone docker (/etc/osmo-role, ou start-interstp.sh).
        [ -n "$HUB_IP" ] || { [ "$NODE_MODE" = native ] && HUB_IP="192.168.1.49"; }
    fi
    _m3ua_ecoute() {   # $1 IP : un hub repond-il en M3UA (SCTP) sur le port 2908 ?
        python3 -c 'import socket,sys
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM,132); s.settimeout(2)
try: s.connect((sys.argv[1],2908)); sys.exit(0)
except Exception: sys.exit(1)' "$1" 2>/dev/null
    }
    if [ "$INTERSTP" != "0" ] && [ -n "${HUB_IP:-}" ] && [ -r "$INTERSTP_SH" ]; then
        hub_local=0; hub_up=0; hub_owned=0
        ip -4 -o addr show 2>/dev/null | awk '{split($4,a,"/"); print a[1]}' | grep -qxF "$HUB_IP" && hub_local=1
        _m3ua_ecoute "$HUB_IP" && hub_up=1
        # hub_owned : l'adresse repond au ping mais pas en M3UA, et n'est pas a nous -- une AUTRE
        # machine la tient ; la poser ici ferait un conflit d'IP sur le LAN (repris d'origin/main).
       if [ "$hub_up" = 0 ] && [ "$hub_local" = 0 ] && ping -c1 -W1 "$HUB_IP" >/dev/null 2>&1; then hub_owned=1; fi
        lancer=0
        if [ "$INTERSTP" = "1" ]; then lancer=1
        elif [ "$hub_up" = 1 ] && [ "$hub_local" = 0 ]; then lancer=0
        elif [ "$hub_local" = 1 ]; then lancer=1
        elif [ "${WAN_NODE_ID:-0}" = "1" ] && [ "$hub_owned" = 0 ]; then lancer=1
        fi
        if [ "$lancer" = 1 ]; then
            # Les noeuds avec LEUR numero (id:ip) : une table 1,3 reste 1,3 sur le hub.
            ips=""
            for id in "${WAN_NODE_LIST[@]}"; do ips="${ips}${id}:${WAN_IP[$id]} "; done
            if [ "$DRY" -eq 1 ]; then
                printf '  [dry-run] %s --no-save --ops 1 --ip %s --ips "%s"\n' "$INTERSTP_SH" "$HUB_IP" "$ips"
            else
                say_begin "Inter-STP standalone ($HUB_IP)"
                # --no-save : la table WAN de CE noeud (WAN_AUTO=1, son numero,
                # ses indicatifs) vient d'etre ecrite par setup-wan-mesh.sh ;
                # le hub ne la reecrit pas.
                if bash "$INTERSTP_SH" --no-save --ops 1 \
                        --ip "$HUB_IP" --ips "$ips" > "${LOG_DIR:-/tmp}/interstp.log" 2>&1; then
                    say_end " OK " "$C_OK" "Inter-STP" "hub $HUB_IP:2908, $WAN_NODE_COUNT noeud(s)"
                else
                    say_end " KO " "$C_KO" "Inter-STP" "voir ${LOG_DIR:-/tmp}/interstp.log"
                fi
            fi
        elif [ "$hub_up" = 1 ]; then
            say_end " -- " "$C_DIM" "Inter-STP" "hub $HUB_IP deja en service (M3UA 2908) - non relance"
        elif [ "$hub_owned" = 1 ]; then
            say_end " KO " "$C_KO" "Inter-STP" "$HUB_IP repond au ping mais pas en SCTP 2908 : lancer
             ↪  start-interstp.sh SUR cette machine-la"
        else
```

```
            say_end " -- " "$C_DIM" "Inter-STP" "hub $HUB_IP muet, non lance ici (noeud ${WAN_NODE_ID:-?}, pas le
             ↪  1) - --interstp pour le forcer"
        fi
    fi

    # --- 7quinquies. Adressage SS7 du noeud (set_stp_ip.sh) -------------------
    # setup-wan-mesh.sh monte la VOIX et les SMS entre noeuds ; il ne touche pas
    # au SS7. Sans ce qui suit, --wan donnait un WAN ou les appels passaient mais
    # ou l'ASP inter-STP visait encore 127.0.0.1, en shutdown : l'interco SS7 ne
    # montait jamais, et rien dans la sortie ne disait pourquoi.
    #
    # set_stp_ip.sh est le seul a ecrire d'un coup les trois configs, la table
    # WAN et /etc/osmo-role. On lui passe --no-ip : l'adresse locale est deja
    # posee (DHCP ou bloc WAN ci-dessus), et --restart serait absurde ici
    # puisque run.sh va demarrer la pile juste apres, avec les fichiers corriges.
    STP_IP_SH="$HERE/set_stp_ip.sh"
    if [ -r "$STP_IP_SH" ] && [ "${WAN_NODE_ID:-0}" != "0" ]; then
        self_ip="${WAN_IP[$WAN_NODE_ID]:-}"
        hub_ip="$HUB_IP"
        [ -n "$hub_ip" ] || hub_ip="$(awk -F= '/^OSMO_HUB_IP=/{gsub(/[ \r\t]/,"",$2);v=$2} END{print v}'
         ↪  /etc/osmo-role 2>/dev/null)"

        stp_args=(--operator --node "$WAN_NODE_ID" --no-ip --conf-dir "${OSMOCOM_CFG:-/etc/osmocom}")
        [ -n "$self_ip" ] && stp_args+=(--ip "$self_ip")
        [ -n "$hub_ip" ]  && stp_args+=(--hub-ip "$hub_ip")

        if [ "$DRY" -eq 1 ]; then
            printf '  [dry-run] %s %s\n' "$STP_IP_SH" "${stp_args[*]}"
        else
            say_begin "Adressage SS7 - inter-STP ${hub_ip:-?}"
            # </dev/null : appel automatique, jamais interactif (set_stp_ip.sh
            # demande l'adresse du hub quand elle manque - ici, personne ne
            # repondrait et le lancement resterait fige).
            if bash "$STP_IP_SH" "${stp_args[@]}" </dev/null > "${LOG_DIR:-/tmp}/set-stp-ip.log" 2>&1; then
                say_end " OK " "$C_OK" "Adressage SS7" "noeud $WAN_NODE_ID -> inter-STP ${hub_ip:-?}"
            else
                # Non fatal : la pile locale tourne tres bien sans interco. On le
                # DIT, plutot que de laisser croire a un WAN complet.
                say_end " ~~ " "$C_SK" "Adressage SS7" "echec - voir ${LOG_DIR:-/tmp}/set-stp-ip.log"
            fi
        fi
    fi
fi

# --- 7quater-bis. Hub SS7 local (banc autonome) ----------------------------
# [2026-10-15] LE RESTART POSE LE SS7. Hors WAN, le STP local garde un ASP
# « asp-to-inter » vers un inter-STP que personne ne lancait : as-inter
# AS_DOWN, barreau ss7 en echec a chaque restart. Si l'adresse de ce hub
# (remote-ip de asp-to-inter) est portee par CETTE machine et que rien n'y
# repond en M3UA (SCTP 2908), on le lance ici, avant run.sh : l'ASP du STP
# s'y attache des son demarrage. Noeud et operateur viennent de la
# routing-key de as-inter (rctx = noeud*1000 + op*100 + 50). Un hub distant n'est
# jamais touche ; un hub local deja en service ne l'est que s'il porte encore
# les noeuds d'un WAN precedent (voir plus bas) ; --no-interstp coupe, --no-save garde
# la table WAN. Meme logique que tests/modules/33-ss7.sh.
if [ "$ACTION" = "start" ] && [ "$WAN_MESH" -ne 1 ] && [ "$INTERSTP" != "0" ]; then
    _stp_cfg="${OSMOCOM_CFG:-/etc/osmocom}/osmo-stp.cfg"
    _hub="$(awk '/^ *asp asp-to-inter /{f=1;next} f&&/^ *remote-ip /{print $2; exit} f&&/^ (asp|as) /{exit}'
     ↪  "$_stp_cfg" 2>/dev/null)"
    _rctx="$(awk '/^ *as as-inter /{f=1;next} f&&/^ *routing-key /{print $2; exit} f&&/^ (asp|as) /{exit}'
     ↪  "$_stp_cfg" 2>/dev/null)"
    if [ -n "$_hub" ] && [[ "$_rctx" =~ ^[0-9]+$ ]] && [ -r "$HERE/launch/start-interstp.sh" ] \
       && ip -4 -o addr show 2>/dev/null | awk '{split($4,a,"/"); print a[1]}' | grep -qxF "$_hub"; then
        _noeud=$(( _rctx / 1000 )); _op=$(( (_rctx / 100) % 10 ))
        # [2026-10-18] Un hub DEJA en service n'etait jamais touche - y compris
        # quand il tournait encore avec la table d'un WAN precedent : il declarait
```

```
        # les AS des autres noeuds (as-n2op1, rctx 2150...), sans ASP puisque ces
        # noeuds ne s'y attachent plus, et les annoncait PROHIB a notre STP
        # (route « 1.21.2 ... PROHIB dyn », global_check en echec). Vider les
        # tables du STP n'y peut rien : la route est dynamique, le hub la
        # re-annonce. Hors WAN, un hub local qui porte d'autres routing-keys que
        # la notre est donc relance avec notre seul noeud.
        _hub_cfg="${OSMOCOM_CFG:-/etc/osmocom}/osmo-stp-interop.cfg"
        _hub_autres="$(awk -v k="$_rctx" '/^ *routing-key /{ if ($2 != k) print $2 }' "$_hub_cfg" 2>/dev/null | tr
         ↪  '\n' ' ')"
        if python3 -c 'import socket,sys
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM,132); s.settimeout(2)
s.connect((sys.argv[1],2908))' "$_hub" 2>/dev/null && [ -z "${_hub_autres// }" ]; then
            say_end " -- " "$C_DIM" "Hub SS7 local" "$_hub:2908 deja en service"
        elif [ "$DRY" -eq 1 ]; then
            printf '  [dry-run] start-interstp.sh --ip %s --ops %s --ips "%s:%s" --no-save\n' "$_hub" "$_op"
             ↪  "$_noeud" "$_hub"
        elif [ "$_noeud" -ge 1 ] && [ "$_noeud" -le 9 ] && [ "$_op" -ge 1 ]; then
            if [ -n "${_hub_autres// }" ]; then
                say_begin "Hub SS7 local ($_hub) - table perimee (rctx ${_hub_autres% }), relance avec le noeud
                 ↪  $_noeud seul"
            else
                say_begin "Hub SS7 local ($_hub)"
            fi
            if bash "$HERE/launch/start-interstp.sh" --ip "$_hub" --ops "$_op" --ips "$_noeud:$_hub" --no-save \
                    > "${LOG_DIR:-/tmp}/interstp.log" 2>&1 < /dev/null; then
                say_end " OK " "$C_OK" "Hub SS7 local" "$_hub:2908, noeud $_noeud op $_op (rctx $_rctx)"
            else
                say_end " KO " "$C_KO" "Hub SS7 local" "voir ${LOG_DIR:-/tmp}/interstp.log"
            fi
        fi
    fi
fi

# --- 7quater-ter. La local-ip de l'ASP suit l'adresse du jour ---------------
# [2026-10-16] L'ASP asp-to-inter part d'une local-ip ecrite en dur dans
# osmo-stp.cfg. Un bail DHCP qui change l'adresse du noeud, et ce bind echoue
# en silence : as-inter reste AS_DOWN, l'interco SS7 ne monte pas et rien ne le
# dit. A CHAQUE start on recalcule l'adresse locale (STP_LOCAL_IP) et on la
# reecrit si elle a bouge, avant que run.sh ne demarre la pile. En WAN c'est
# set_stp_ip.sh qui s'en charge (7quinquies) ; ici, le cas autonome.
#   STP_LOCAL_IP = la patte locale qui route vers le hub (remote-ip de l'ASP) ;
#   au besoin (hub injoignable) l'OSMO_NODE_IP du mynode<N>.conf (N = noeud de
#   la routing-key, rctx = noeud*1000 + op*100 + 50).
if [ "$ACTION" = "start" ] && [ "$WAN_MESH" -ne 1 ]; then
    _stp_cfg="${OSMOCOM_CFG:-/etc/osmocom}/osmo-stp.cfg"
    if [ -w "$_stp_cfg" ]; then
        _hub="$(awk '/^ *asp asp-to-inter /{f=1;next} f&&/^ *remote-ip /{print $2; exit} f&&/^ (asp|as) /{exit}'
         ↪  "$_stp_cfg" 2>/dev/null)"
        _lo_now="$(awk '/^ *asp asp-to-inter /{f=1;next} f&&/^ *local-ip /{print $2; exit} f&&/^ (asp|as) /{exit}'
         ↪  "$_stp_cfg" 2>/dev/null)"
        _rctx="$(awk '/^ *as as-inter /{f=1;next} f&&/^ *routing-key /{print $2; exit} f&&/^ (asp|as) /{exit}'
         ↪  "$_stp_cfg" 2>/dev/null)"
        STP_LOCAL_IP=""
        [ -n "$_hub" ] && STP_LOCAL_IP="$(ip route get "$_hub" 2>/dev/null | sed -n 's/.*src \([0-9.]*\).*/\1/p' |
         ↪  head -1)"
        if [ -z "$STP_LOCAL_IP" ] && [[ "$_rctx" =~ ^[0-9]+$ ]]; then
            _nconf="${OSMOCOM_CFG:-/etc/osmocom}/mynode$(( _rctx / 1000 )).conf"
            [ -r "$_nconf" ] && STP_LOCAL_IP="$(awk -F= '/^OSMO_NODE_IP=/{gsub(/[ \r\t]/,"",$2);v=$2} END{print
             ↪  v}' "$_nconf" 2>/dev/null)"
        fi
        if [ -n "$STP_LOCAL_IP" ] && [ "$STP_LOCAL_IP" != "$_lo_now" ]; then
            if [ "$DRY" -eq 1 ]; then
                printf '  [dry-run] osmo-stp.cfg local-ip %s -> %s\n' "${_lo_now:-?}" "$STP_LOCAL_IP"
            else
                say_begin "Adresse SS7 locale"
                # Borne au bloc asp-to-inter : les local-ip 127.0.0.x du listener
```

```
                # local ne sont pas touchees.
                sed -i "/^ *asp asp-to-inter /,/^ *as /{s/^\( *local-ip *\).*/\1$STP_LOCAL_IP/}" "$_stp_cfg"
                say_end " OK " "$C_OK" "Adresse SS7 locale" "local-ip ${_lo_now:-?} -> $STP_LOCAL_IP"
            fi
        fi
    fi
fi

# --- 7sexies. Le wrapper tcpdump ---------------------------------------------
# La capture GSMTAP est lancee par run_modules/75-gsmtap.sh, dans qosmo : un
# autre depot, qu'on ne modifie pas. Elle ecrit un pcap NON BORNE, et la racine
# du live etant un tmpfs, une capture un peu chargee finit par remplir la RAM.
#
# Le wrapper y substitue un anneau. Il est aussi pose par build-iso.sh, mais une
# ISO deja gravee ne le connait pas, et un live sans persistance le perd a chaque
# reboot : le reposer ICI, a chaque demarrage de la pile, est ce qui le rend
# effectif sans reconstruire ni toucher a qosmo.
#
# DEUX PIEGES, DEUX PARADES - c'est tout l'objet de ce script :
#   -Z root : avec -C/-W, tcpdump cree les membres suivants de l'anneau APRES
#             avoir lache ses privileges. Sans lui : "Permission denied", et
#             AUCUNE capture - alors que sans -C il ouvrait le fichier avant.
#   le lien : avec -C/-W, tcpdump numerote (capture.pcap0, .pcap1...) ; le nom
#             exact demande n'existe jamais, et la barriere du module - un
#             "test -s <nom exact>" - conclut a un echec sur une capture qui
#             tourne. On maintient <nom exact> -> membre courant.
#
# OSMO_NO_PCAP_RING=1 saute l'installation.
install_pcap_wrapper() {
    local dst=/usr/local/bin/tcpdump tmp
    [ "${OSMO_NO_PCAP_RING:-0}" = "1" ] && return 0
    [ "$(id -u)" -eq 0 ] || return 0
    command -v tcpdump >/dev/null 2>&1 || return 0
    # Le vrai binaire doit exister sous /usr/bin ou /usr/sbin ET etre un ELF :
    # un script a sa place (voir le garde-fou dans le wrapper) rendrait le
    # wrapper recursif. Dans ce cas on ne pose rien et on le dit.
    local real=/usr/bin/tcpdump; [ -x "$real" ] || real=/usr/sbin/tcpdump
    if [ ! -x "$real" ] || [ "$(head -c 2 "$real" 2>/dev/null)" = "#!" ]; then
        printf '  %s!%s pcap : %s n est pas le binaire tcpdump - anneau non arme (apt-get install --reinstall
        ↪  tcpdump)\n' \
            "${C_WARN:-}" "$C_Z" "$real" >&2
        return 0
    fi

    tmp="$(mktemp)" || return 0
    cat > "$tmp" <<'PCAPWRAP'
#!/bin/sh
# tcpdump - wrapper osmo-operator : impose un anneau aux captures sur fichier.
# Pose par start-native.sh et par build-iso.sh. Voir leurs commentaires.
# Reglable : OSMO_PCAP_RING_MB (32), OSMO_PCAP_RING_FILES (5).
REAL=/usr/bin/tcpdump
[ -x "$REAL" ] || REAL=/usr/sbin/tcpdump
# [2026-09-04] Si REAL est lui-meme un script (ce wrapper copie par-dessus le
# binaire dpkg, cf. iso_modules/52-qemu.sh), exec "$REAL" boucle a l'infini :
# aucune capture, et un `tcpdump --version` qui ne rend jamais la main.
if [ "$(head -c 2 "$REAL" 2>/dev/null)" = "#!" ]; then
    echo "tcpdump (wrapper) : $REAL n'est pas le vrai binaire - reinstallez le paquet tcpdump" >&2
    exit 127
fi

has_w=0; has_ring=0; has_z=0; wfile=""; next_is_w=0
for a in "$@"; do
    if [ "$next_is_w" = 1 ]; then wfile="$a"; next_is_w=0; continue; fi
    case "$a" in
        -w)            has_w=1; next_is_w=1 ;;
        -w?*)          has_w=1; wfile="${a#-w}" ;;
```

```
        -C|-C*|-W|-W*) has_ring=1 ;;
        -Z|-Z*)        has_z=1 ;;
    esac
done

# Rien a borner, ou l'appelant a deja choisi son anneau : on s'efface.
if [ "$has_w" != 1 ] || [ "$has_ring" = 1 ] || [ -z "$wfile" ]; then
    exec "$REAL" "$@"
fi

# Le veilleur du lien. Lance AVANT l'exec : apres, ce processus EST tcpdump.
# $$ survit a l'exec, donc il suit exactement sa vie et s'arrete avec lui.
( ppid=$$
  n=0
  while [ "$n" -lt 300 ]; do
      [ -e "${wfile}0" ] && break
      kill -0 "$ppid" 2>/dev/null || exit 0
      sleep 0.2; n=$((n + 1))
  done
  while kill -0 "$ppid" 2>/dev/null; do
      newest=$(ls -t "${wfile}"[0-9]* 2>/dev/null | head -1)
      if [ -n "$newest" ] && [ "$(readlink "$wfile" 2>/dev/null)" != "$newest" ]; then
          ln -sfn "$newest" "$wfile"
      fi
      sleep 2
  done ) >/dev/null 2>&1 &

if [ "$has_z" = 1 ]; then
    exec "$REAL" -C "${OSMO_PCAP_RING_MB:-32}" -W "${OSMO_PCAP_RING_FILES:-5}" "$@"
fi
exec "$REAL" -Z root -C "${OSMO_PCAP_RING_MB:-32}" -W "${OSMO_PCAP_RING_FILES:-5}" "$@"
PCAPWRAP

    # Le wrapper s'appelle "tcpdump" et vit dans /usr/local/bin, qui precede
    # /usr/bin : il ne doit donc JAMAIS s'installer par-dessus le vrai binaire,
    # sous peine de recursion infinie. On verifie ou pointe le vrai.
    case "$(command -v tcpdump)" in
        "$dst") ;;                      # deja le wrapper, on remplace
        /usr/bin/tcpdump|/usr/sbin/tcpdump|/bin/tcpdump) ;;
        *) rm -f "$tmp"; return 0 ;;    # emplacement inattendu : on s'abstient
    esac

    if cmp -s "$tmp" "$dst" 2>/dev/null; then
        rm -f "$tmp"                    # deja a jour, rien a dire
    else
        install -m 0755 "$tmp" "$dst" 2>/dev/null \
            && printf '  %spcap%s       anneau de capture arme (%s)\n' \
                 "$C_DIM" "$C_Z" "$dst"
        rm -f "$tmp"
    fi
}
[ "$DRY" -eq 1 ] || install_pcap_wrapper

# --- 7ter. ARRET SYSTEMATIQUE AVANT DEMARRAGE ---------------------------------
# Tout demarrage arrete d'abord la pile. Ce n'est pas de la prudence : sans cela
# le lanceur differe du pont part sur un etat ambigu.
#
# CE QUE CA CORRIGE. La barriere du pont attend osmo-bts-trx pour savoir que les
# ports 5700-5702 sont a lui. Si un run precedent tourne encore, son osmo-bts-trx
# est deja la : la barriere est franchie a l'instant meme, le pont binde EN PLEIN
# teardown de run.sh, qui le tue derriere (f:pont/pont.py est dans _td_patterns)
# -- et rien ne le relance. Symptome : « le pont n'a pas de bursts », sans un
# message pour le dire. La barriere attend desormais les deux fronts (disparition
# puis apparition), mais partir d'une pile arretee rend la question sans objet.
#
# Et cela libere les ports : un fake_trx ou un trxcon orphelin suffit a faire
```

```
# echouer le demarrage suivant sur « port UDP 5720 deja pris » -- constate le
# 27/08. Le killall couvre les processus Python que le registre ne connait pas.
#
# CALYPSO_NO_AUTOSTOP=1 desactive ce comportement.
# SAP : arrete les serveurs lances par start_sap (processus demo_server.rb de
# softSIM) et supprime le socket SAP. Le motif [d] evite de se trouver soi-meme.
kill_softsim() {
    local p sock="${SAP_SOCK:-/tmp/osmocom_sap}"
    for p in $(ps -eo pid,args | awk '/[d]emo_server\.rb/ {print $1}'); do
        kill -TERM "$p" 2>/dev/null || true
    done
    rm -f "$sock" 2>/dev/null || true
}
```

3.382 /opt/GSM/osmo-operator/startup_scripts/13-adresses.sh

21250 octets, 372 lignes → 372 lignes

```
# 13-adresses.sh - partie de start.sh (lignes 2944-3312 de start-native.sh d'origine)
# Sourcee par start.sh dans le shell principal : meme ordre, memes variables.
if [ "$DRY" -eq 0 ] && [ "${CALYPSO_NO_AUTOSTOP:-0}" != 1 ]; then
    say_begin "Arret de la pile avant demarrage"
    banc_dsp_arreter >/dev/null 2>&1 || true
    banc_dsp_arreter_instances
    bash "$RUN_SH" --stop --profile "$CALYPSO_PROFILE" >/dev/null 2>&1 || true
    declare -F purge_sessions_tmux >/dev/null && purge_sessions_tmux
    if [ "${CALYPSO_STOP_KILL_PYTHON:-1}" != 0 ]; then
        killall_python -TERM || true
    fi
    # SAP : les serveurs softSIM (demo_server.rb) et leur socket. CALYPSO_STOP_KILL_SOFTSIM=0 le desactive.
    if [ "${CALYPSO_STOP_KILL_SOFTSIM:-1}" != 0 ]; then
        kill_softsim || true
    fi
    say_end " OK " "$C_OK" "Arret de la pile avant demarrage"
fi

# --- 7sexies. Socket PCU (BTS <-> osmo-pcu) -----------------------------------
# Les sockets d'un run precedent restent sur le disque ; le PCU, lui, doit
# trouver le socket du BTS principal (/tmp/pcu_bts). Le harnais supprime les
# orphelins avant le lancement et affiche l'etat, pour qu'un GPRS muet se lise.
PCU_SOCK_SH="$HERE/network/pcu-socket.sh"
if [ "$DRY" -eq 0 ] && [ -r "$PCU_SOCK_SH" ]; then
    say_begin "Socket PCU"
    if bash "$PCU_SOCK_SH" --fix > "${LOG_DIR:-/tmp}/pcu-socket.log" 2>&1; then
        say_end " OK " "$C_OK" "Socket PCU" "voir ${LOG_DIR:-/tmp}/pcu-socket.log"
    else
        say_end " KO " "$C_KO" "Socket PCU" "suppression d'un orphelin impossible - voir
        ↪  ${LOG_DIR:-/tmp}/pcu-socket.log"
    fi
fi

# --- 7septies. Sortie Internet de l'APN (NAT apn0 -> carte qui route) ----------
APN_NAT_SH="$HERE/network/apn-nat.sh"
if [ "$DRY" -eq 0 ] && [ -r "$APN_NAT_SH" ]; then
    say_begin "Sortie Internet APN (NAT)"
    if bash "$APN_NAT_SH" > "${LOG_DIR:-/tmp}/apn-nat.log" 2>&1; then
        say_end " OK " "$C_OK" "Sortie Internet APN" "voir ${LOG_DIR:-/tmp}/apn-nat.log"
    else
        say_end " KO " "$C_KO" "Sortie Internet APN" "voir ${LOG_DIR:-/tmp}/apn-nat.log"
    fi
fi

# --- 8. lancement : exec run.sh -----------------------------------------------
# ── LES ADRESSES QUE LES CONFIGS RECLAMENT DOIVENT EXISTER ──────────────────
```

```
# Un demon qui se lie a une adresse absente ne demarre pas, et il le dit dans
# SON journal, pas dans le notre :
#     osmo-ggsn : « gtp bind-ip 192.168.2.10 » -> adresse introuvable localement
#     osmo-sgsn : « listen 192.168.2.10 23000 » -> idem
# Ces adresses viennent du generateur de gabarits (op_private_ip), pas du
# reseau : rien ne les pose. Elles etaient figees dans 20-dhcp.network cote ISO,
# donc absentes de toute machine montee autrement - et la panne se lisait comme
# un defaut de coeur paquet.
#
# osmo-ip-plan.sh les EXTRAIT des configs plutot que de les recalculer (le plan
# peut venir d'un autre generateur que le notre), choisit la carte qui fournit
# reellement Internet, et retombe sur la boucle locale quand aucune ne mene
# nulle part. Il ne fait rien dans une image de ce depot (OSMO_IMAGE=1) : la,
# c'est start-docker.sh / compose qui adresse. Un conteneur quelconque est traite en natif.
if [ "$ACTION" = "start" ] && [ "$DRY" -ne 1 ]; then
    IPPLAN="$HERE/network/osmo-ip-plan.sh"
    if [ -r "$IPPLAN" ]; then
        say_begin "Adresses privees du noeud"
        _ipout="$(OSMOCOM_CFG="${OSMOCOM_CFG:-/etc/osmocom}" \
                  OSMO_WAN_NODE="${NODE_ID:-${OSMO_WAN_NODE:-}}" \
                  bash "$IPPLAN" --apply 2>&1)" || true
        say_end " OK " "$C_OK" "Adresses privees du noeud" \
            "$(printf '%s' "$_ipout" | tail -1 | sed 's/^osmo-ip-plan : //')"
    fi
fi

# ── LE TABLEAU DE BORD REDEMARRE AVEC LE BANC ──────────────────────────
# [2026-09-09] GESTE SYMETRIQUE DE CELUI DE `stop)` (voir plus haut, ~l.1455).
# Depuis le 04/09 le chemin d arret coupe osmo-egprs-web.service, pour ne plus
# laisser un :8080 qui repond devant une pile morte. Mais RIEN ne le rallumait :
# l unite est PartOf=osmo-banc.service, or le lancement direct ne passe pas par
# osmo-banc. Consequence mesuree le 09/09 : apres un --stop suivi d un
# demarrage direct, le dashboard restait eteint pour de bon.
# Ce n est pas qu un tableau de bord qui manque : c est LUI qui sert /psd, donc
# tools/osmo-fft-snap.py interrogeait en boucle un port muet et la FFT du fond
# d ecran ne se mettait plus a jour. Le symptome ne designait pas sa cause.
# (fft-web/fft_web.py, lui, sert le 8081 : c est un autre serveur, il n est pas
# concerne par ce bloc.)
# --no-block, comme a l arret : on ne se bloque pas sur systemd. `restart` et
# pas `start` : si une instance survit d un run precedent, on la reprend au lieu
# de la laisser servir l etat d avant.
# CALYPSO_NO_DASHBOARD=1 desactive ce demarrage.
if [ "$ACTION" = "start" ] && [ "$DRY" -ne 1 ] && [ "${CALYPSO_NO_DASHBOARD:-0}" != 1 ]; then
    say_begin "Tableau de bord (:8080)"
    if [ -d /run/systemd/system ] && command -v systemctl >/dev/null 2>&1; then
        if systemctl --no-block restart osmo-egprs-web.service 2>/dev/null; then
            say_end " OK " "$C_OK" "Tableau de bord (:8080)" "osmo-egprs-web.service"
        else
            say_end " -- " "$C_KO" "Tableau de bord (:8080)" "unite absente ou refusee"
        fi
    elif [ -r /opt/GSM/osmo-egprs-web/server.js ] && command -v node >/dev/null 2>&1; then
        # Hors systemd (conteneur) : lancement direct, comme le fait osmo-launch.sh.
        pkill -f "osmo-egprs-web/server.js" 2>/dev/null || true
        ( cd /opt/GSM/osmo-egprs-web && setsid node server.js \
            >>"${LOG_DIR:-/tmp}/web-dashboard.log" 2>&1 </dev/null & )
        say_end " OK " "$C_OK" "Tableau de bord (:8080)" "node server.js"
    else
        say_end " -- " "$C_KO" "Tableau de bord (:8080)" "ni systemd ni node/server.js"
    fi
fi

# ── LA FICHE DU NOEUD : LE RESULTAT DU RUN ─────────────────────────────
# [2026-09-04] /etc/osmocom/mynode<N>.conf (network/node-conf.sh) : ce qu'un
# autre noeud doit savoir pour joindre celui-ci (--wan mynode<N>.conf). Ecrite
# ici, une fois l'identite SS7, l'adresse et l'indicatif definitivement poses.
# start-multi.sh la reecrit quand les conteneurs sont montes (--gen-conf).
```

```
if [ "$ACTION" = "start" ] && [ "$DRY" -ne 1 ]; then
    . "$HERE/network/wan-nodes.sh" 2>/dev/null
    . "$HERE/network/node-conf.sh" 2>/dev/null
    _nc_node="${NODE_ID:-${WAN_NODE_ID:-}}"; [[ "${_nc_node:-0}" =~ ^[1-9]$ ]] || _nc_node=1
    [ "${WAN_NODE_COUNT:-0}" -gt 0 ] || wan_nodes_load "${WAN_CONF_FILE:-/etc/osmo-wan.conf}" 2>/dev/null || true
    _nc_out="$(node_conf_path "$_nc_node")"
    if node_conf_write "$_nc_out" "$_nc_node" "${NODE_OP:-1}" "${HUB_IP:-${OSMO_HUB_IP:-}}" 2>/dev/null; then
        say_end " OK " "$C_OK" "Fiche du noeud $_nc_node" "$_nc_out  (pour un autre noeud : --wan $(basename
        ↪  "$_nc_out"))"
    else
        say_end " -- " "$C_DIM" "Fiche du noeud" "non ecrite ($_nc_out)"
    fi
fi

# ── LE RACCORD MOBILE : oFONO = LE MODEM DU BANC ──────────────────────────
# [2026-09-07] Le telephone du banc - la VM postmarketOS/Phosh - est le MEME
# abonne que le mobile osmocom-bb : 100101. Son ModemManager ne parle qu a un
# modem AT ; ce banc n a pas de dongle, le modem est donc ecrit ici. La VM se
# branche directement sur osmo-phonesim-banc.py (mode --connect, cf.
# tools/osmo-pmos.sh) ; oFono, lui, reste utile a l hote - il donne osmo-call
# et osmo-sms-send, et un second modem AT pour la mise au point.
#
#   tools/osmo-phonesim-banc.py   le modem AT du banc, cote oFono (TCP 12345,
#                                 plugin phonesim). Il lit le reseau sur le VTY
#                                 d osmo-bsc, envoie les SMS par le VTY
#                                 d osmo-msc, passe les appels par Asterisk, et
#                                 recoit les SMS en ESME SMPP (port 2775).
#   tools/osmo-ril-atmodem.py     le modem AT que rild consommera, adosse a
#                                 oFono. Il publie son pty dans
#                                 /run/osmo-ril/at-pty.
#
# Rien de tout ca n est indispensable au banc : si oFono manque, on le dit et on
# continue. Le banc doit demarrer meme sans le telephone Android.
if [ "$ACTION" = "start" ] && [ "$DRY" -ne 1 ] && [ "${OSMO_RACCORD_MOBILE:-1}" = "1" ]; then
    say_begin "Raccord mobile (oFono)"
    _rm_detail=""
    if ! command -v ofonod >/dev/null 2>&1; then
        say_end " -- " "$C_DIM" "Raccord mobile (oFono)" "ofono absent - ignore"
    else
        # 1. oFono doit savoir ou trouver le modem du banc.
        if ! grep -q '^\[osmo\]' /etc/ofono/phonesim.conf 2>/dev/null; then
            printf '\n[osmo]\nAddress=127.0.0.1\nPort=12345\n' >> /etc/ofono/phonesim.conf
            _rm_detail="phonesim.conf pose"
        fi
        # 2. le compte AMI : c est par lui que le modem fait naitre les appels
        #    (la CLI d Asterisk n a pas « channel originate » ici, le module
        #    res_clioriginate n est pas charge - modules.conf est en autoload=no).
        if [ -d /etc/asterisk/manager.d ] && [ ! -f /etc/asterisk/manager.d/osmo.conf ]; then
            install -m640 "$HERE/configs/manager-osmo.conf" \
                /etc/asterisk/manager.d/osmo.conf 2>/dev/null && \
                { chown asterisk:asterisk /etc/asterisk/manager.d/osmo.conf 2>/dev/null || true; }
            asterisk -rx "manager reload" >/dev/null 2>&1 || true
            _rm_detail="${_rm_detail:+$_rm_detail, }compte AMI pose"
        fi
        # 3. le modem du banc, PUIS oFono (l ordre compte : ofonod cherche le
        #    modem au demarrage et n y revient pas de lui-meme).
        pgrep -f "$HERE/tools/osmo-phonesim-banc.py" >/dev/null 2>&1 || \
        # [2026-09-22] LANCEMENT HORS TABLE DES JOBS.
        # `setsid` detache la SESSION, pas le job : bash gardait ces aides dans
        # sa table et annoncait leur mort par un
        #     ./start-native.sh: line NNNN: <pid> Killed setsid ...
        # asynchrone, qui tombait par-dessus le prompt apres un --stop
        # (killall_python les tue). `disown` n'y suffit pas de facon fiable en
        # shell non interactif. Le sous-shell, lui, est sur : bash ne voit que
        # LUI comme job, il rend la main tout de suite avec 0, et le petit-fils
        # est adopte par init -- plus personne ne le surveille ni ne le signale.
```

```
        # `</dev/null` en prime : aucune de ces aides ne lit le terminal, et
        # sans cela elles restent rattachees a son entree.
        # [2026-09-22] LA VTY DU MOBILE N'EST PAS TOUJOURS 4247.
        # osmo-phonesim-banc.py retombe sur OSMO_MOB_VTY_PORT=4247, la valeur de
        # /root/.osmocom/bb/mobile.cfg. En --dsp le mobile qui tourne est celui
        # du banc (mobile_pont.cfg, VTY 4347) : le modem parlait a un port que
        # personne n'ecoute, donc pas de reseau, pas d'appel, pas de SMS par
        # oFono -- sans la moindre erreur, il se contentait de ne jamais voir le
        # mobile. On lui passe le port reellement lie, lu dans le fichier.
            ( setsid env OSMO_MOB_VTY_PORT="${_ms1_vty:-4247}" \
                  "$HERE/tools/osmo-phonesim-banc.py" \
                  </dev/null >>/tmp/osmo-phonesim.log 2>&1 & )
        sleep 1
        systemctl restart ofono >/dev/null 2>&1 || true
        sleep 3
        # 4. allumer le modem : oFono le laisse eteint tant qu on ne demande rien.
        /usr/bin/python3 - <<'PYON' >/dev/null 2>&1 || true
from gi.repository import Gio, GLib
import time
c = Gio.bus_get_sync(Gio.BusType.SYSTEM, None)
for prop in ("Powered", "Online"):
    try:
        c.call_sync("org.ofono", "/osmo", "org.ofono.Modem", "SetProperty",
                    GLib.Variant("(sv)", (prop, GLib.Variant("b", True))),
                    None, Gio.DBusCallFlags.NONE, 25000, None)
    except GLib.Error:
        pass
    time.sleep(2)
PYON
        # 5. le modem AT virtuel (le pty que consomme un RIL, ou un modem de
        #    mise au point pour tools/at-cmd.py).
        #    /usr/bin/python3 EN DUR, et non le shebang « env python3 ».
        #    [2026-09-07] Releve sur le banc : ce programme ne demarrait plus
        #    du tout depuis des heures -
        #        File ".../osmo-ril-atmodem.py", line 43, in <module>
        #          from gi.repository import Gio, GLib
        #        ModuleNotFoundError: No module named 'gi'
        #    - et le seul endroit ou ca se voyait etait /tmp/osmo-ril.log, que
        #    personne ne lit. La cause n est pas le paquet : PyGObject est bien
        #    la, cote SYSTEME. C est le PATH du banc, qui place le venv
        #    (/root/.env/bin) en tete : « env python3 » y trouve un interprete
        #    qui n a pas les liaisons GObject, et jamais celui du systeme. Meme
        #    raison que le bloc PYON ci-dessus, qui appelle deja /usr/bin/python3.
        pgrep -f "$HERE/tools/osmo-ril-atmodem.py" >/dev/null 2>&1 || \
        # [2026-09-22] LANCEMENT HORS TABLE DES JOBS.
        # `setsid` detache la SESSION, pas le job : bash gardait ces aides dans
        # sa table et annoncait leur mort par un
        #     ./start-native.sh: line NNNN: <pid> Killed setsid ...
        # asynchrone, qui tombait par-dessus le prompt apres un --stop
        # (killall_python les tue). `disown` n'y suffit pas de facon fiable en
        # shell non interactif. Le sous-shell, lui, est sur : bash ne voit que
        # LUI comme job, il rend la main tout de suite avec 0, et le petit-fils
        # est adopte par init -- plus personne ne le surveille ni ne le signale.
        # `</dev/null` en prime : aucune de ces aides ne lit le terminal, et
        # sans cela elles restent rattachees a son entree.
            ( setsid /usr/bin/python3 "$HERE/tools/osmo-ril-atmodem.py" \
                  </dev/null >>/tmp/osmo-ril.log 2>&1 & )
        sleep 1
        if pgrep -f "$HERE/tools/osmo-phonesim-banc.py" >/dev/null 2>&1; then
            say_end " OK " "$C_OK" "Raccord mobile (oFono)" \
                "modem /osmo${_rm_detail:+ - $_rm_detail} ; pty RIL : /run/osmo-ril/at-pty"
        else
            say_end " -- " "$C_DIM" "Raccord mobile (oFono)" "modem du banc non demarre (cf.
             ↪  /tmp/osmo-phonesim.log)"
        fi
    fi
```

```
fi

# ── LE BANC DSP : REGLAGES MESURES, PUIS PASSAGE DE MAIN ──────────────────
# Les valeurs viennent du banc, pas d'un choix de style (c54x_exe/run_real.sh
# les porte depuis le 2026-09-21, run.sh les documente une a une) :
#   INSNS=200000             plafond d'instructions par trame (voir plus bas)
#   LOCKSTEP=1               QEMU n'avance la trame que quand le DSP a fini la
#                            precedente ; sans lui QEMU saute 3 trames sur 4
#   IQ=none                  aucune cellule synthetique : les bursts viennent
#                            de la vraie BTS, par pont.py
#   CALYPSO_BSP_STREAM=1     un burst TS0 par trame, dans l'ordre des FN
#   CALYPSO_RHEA_DMA_XFER=1  sans lui la page API n'est jamais remplie : pas de
#                            SB, pas de BCCH, pas de SI
# Tout reste surchargeable : on ne pose que ce que l'operateur n'a pas dit.

# ── SAP : LA SIM SERVIE AU MOBILE SUR LE SOCKET SAP (--sap) ───────────────
# Serveur demo_server.rb du depot softSIM (/opt/GSM/softsim, shim Ruby 3.2+ :
# File.exists?, Fixnum), lance AVANT la pile : le mobile (sim sap) s y connecte
# au demarrage et y reste (patch 0002 : le serveur sert le client suivant apres
# une deconnexion, donc un redemarrage de la baseband ne demande rien).
#   type pcsc : la carte du lecteur PC/SC (HID Omnikey 6121, sysmoISIM-SJA5 ;
#               gem smartcard). La carte doit porter l IMSI/Ki du MS#1 (ou son
#               abonne doit etre dans le HLR) : tools/osmo-sim.sh verifier.
#   type sim  : la SIM logicielle sim-op1.xml, regeneree a chaque lancement
#               (IMSI du MS#1 ; COMP128v1 avec SOFTSIM_KI = Ki du MS#1, patch 0001).
# En DSP, le MS#1 qui tourne est celui de c54x_exe (mobile_pont.cfg, sim test) :
# on en derive $BB_DIR/mobile_pont_sap.cfg (sim sap, sap-socket) et le banc le
# recoit par MOBILE_CFG (SAP_DSP_MOBILE_CFG, a lui seul). Variables :
# SOFTSIM_DIR, SAP_SOCK, SAP_TYPE, SAP_FILE, SAP_LOG, SAP_VERBOSITY, SOFTSIM_KI.
# le python qui a pyscard : celui du venv pySim (le PATH de root commence par
# /root/.env, le venv du pont, sans pyscard), sinon celui du systeme
sap_python() {
    local p="${PYSIM_VENV:-/opt/GSM/pysim-venv}/bin/python3"
    [ -x "$p" ] || p=/usr/bin/python3
    printf '%s' "$p"
}
sap_carte_presente() {
    "$(sap_python)" - <<'PYSC' 2>/dev/null
import sys
try:
    from smartcard.System import readers
except Exception:
    sys.exit(2)
for r in readers():
    try:
        c = r.createConnection(); c.connect(); c.disconnect(); sys.exit(0)
    except Exception:
        pass
sys.exit(1)
PYSC
}
sap_lecteur() {
    "$(sap_python)" -c 'from smartcard.System import readers; print(str(readers()[0]))' 2>/dev/null | sed "s/
    ↪  \[.*//"
}
SAP_DSP_MOBILE_CFG=""
start_sap() {
    [ "${SAP_MODE:-0}" = 1 ] || return 0
    local sdir="${SOFTSIM_DIR:-/opt/GSM/softsim}" sock="${SAP_SOCK:-/tmp/osmocom_sap}"
    local log="${SAP_LOG:-/tmp/osmo-sap.log}" type="${SAP_TYPE:-auto}" file="" detail=""
    local shim=/usr/local/share/softsim/ruby_compat.rb
    [ -f "$shim" ] || shim="$HERE/tools/softsim/ruby_compat.rb"
    say_begin "SIM via SAP"
    if [ ! -f "$sdir/src/demo_server.rb" ] || ! command -v ruby >/dev/null 2>&1; then
       say_end " KO " "$C_KO" "SIM via SAP" "softSIM ou ruby absent ($sdir) : sudo $HERE/tools/osmo-sim-install.sh"
        return 1
```

```
fi
if [ "$type" = auto ]; then
    if sap_carte_presente; then type=pcsc; else type=sim; fi
fi
case "$type" in
    pcsc)
        if ! sap_carte_presente; then
            say_end " KO " "$C_KO" "SIM via SAP" "aucune carte dans un lecteur PC/SC (pcscd actif ? Omnikey
             ↪  branche ? tools/osmo-sim.sh lecteur)"
            return 1
        fi
        if ! ruby -e 'require "smartcard"' >/dev/null 2>&1; then
            say_end " KO " "$C_KO" "SIM via SAP" "gem smartcard absent : sudo $HERE/tools/osmo-sim-install.sh"
            return 1
        fi
        detail="carte du lecteur $(sap_lecteur)" ;;
    sim)
        file="${SAP_FILE:-$sdir/src/sim-op1.xml}"
        # le Ki du MS#1, celui que 21-abonnes-hlr met dans le HLR (COMP128v1)
        export SOFTSIM_KI="${SOFTSIM_KI:-$(ms_ki 1 | tr -d ' ')}"
        detail="SIM logicielle $file (IMSI $(ms_imsi 1))" ;;
    *)
        say_end " KO " "$C_KO" "SIM via SAP" "SAP_TYPE=$type inconnu (pcsc, sim, auto)"
        return 1 ;;
esac
# En DSP : la copie de mobile_pont.cfg en « sim sap », pour le banc seul.
if [ "${DSP_BANC:-0}" = 1 ]; then
    local _src="${MOBILE_CFG:-$(dirname "$BANC_DSP")/mobile_pont.cfg}" _dst="$BB_DIR/mobile_pont_sap.cfg"
    if [ -r "$_src" ]; then
        sed -e "s#^\([[:space:]]*\)sim \(test\|reader\)\$#\1sim sap#" \
            -e "s|^\([[:space:]]*\)sap-socket .*|\1sap-socket $sock|" \
            -e "s|^\([[:space:]]*\)stick [0-9]*\$|\1stick ${PLAN_ARFCN:-${ARFCN:-1}}|" "$_src" > "$_dst.tmp" \
            && mv -f "$_dst.tmp" "$_dst"
        grep -q "^[[:space:]]*sap-socket " "$_dst" \
            || sed -i "s|^\([[:space:]]*\)layer2-socket .*|&\n\1sap-socket $sock|" "$_dst"
        SAP_DSP_MOBILE_CFG="$_dst"
        detail="$detail ; MS#1 $_dst"
    else
        say_end " KO " "$C_KO" "SIM via SAP" "config du MS#1 du banc DSP illisible : $_src"
        return 1
    fi
fi
if [ $DRY -eq 1 ]; then
    say_end " -- " "$C_DIM" "SIM via SAP" "dry-run : type $type, ruby -r$shim demo_server.rb --unix
     ↪  $sock${file:+ --file $file} ($detail)"
    return 0
fi
# sim-op1.xml est regenere a chaque lancement : le serveur le reecrit a chaque
# deconnexion du client, et un fichier ancien garderait l ICCID / le PLMN d avant.
if [ "$type" = sim ] && [ -z "${SAP_FILE:-}" ] && [ -f "$sdir/src/sim.xml" ] && [ -f
 ↪  "$HERE/tools/softsim/make_sim_op1.py" ]; then
    python3 "$HERE/tools/softsim/make_sim_op1.py" "$sdir/src/sim.xml" "$sdir/src/sim-op1.xml" \
        --imsi "$(ms_imsi 1)" >/dev/null 2>&1 || true
    [ -f "$file" ] || file="$sdir/src/sim.xml"
fi
rm -f "$sock"
local args=(--type "$type" --socket unix --unix "$sock" --verbosity "${SAP_VERBOSITY:-3}")
[ "$type" = sim ] && args+=(--file "$file")
( cd "$sdir/src" && nohup ruby -r"$shim" demo_server.rb "${args[@]}" >"$log" 2>&1 & )
for _ in $(seq 1 50); do [ -S "$sock" ] && break; sleep 0.2; done
if [ -S "$sock" ]; then
    say_end " OK " "$C_OK" "SIM via SAP" "type $type : $detail ; socket $sock, journal $log"
else
    say_end " KO " "$C_KO" "SIM via SAP" "pas de socket $sock : voir $log"
    return 1
fi
```

```
}
start_sap || { echo "SAP demande (--sap) mais non lance" >&2; exit 1; }
```

3.383 /opt/GSM/osmo-operator/startup_scripts/14-demarrage.sh

11607 octets, 167 lignes → 167 lignes

```
# 14-demarrage.sh - partie de start.sh (lignes 3313-3442 de start-native.sh d'origine)
# Sourcee par start.sh dans le shell principal : meme ordre, memes variables.
if [ "$DSP_BANC" = 1 ]; then
    # ── LE FORK MONTE TOUT, MOINS LA CHAINE CALYPSO ──────────────────────
    # [2026-09-22] Deux versions ratees avant celle-ci, toutes deux pour la
    # meme raison -- avoir voulu remplacer le pipeline au lieu de s'y inserer :
    #   1. RUN_SH = c54x_exe/run.sh : la pile osmocom ne montait plus du tout
    #      (coeur, BTS, side-car, tmux sont des run_modules du fork) ;
    #   2. delegation a c54x_exe/run_real.sh : meme probleme, plus un verdict
    #      « ECHEC : aucun SI lu » imprime apres zero seconde d'observation.
    # Ici le fork joue son plan habituel, on lui retire seulement les quatre
    # modules que c54x_exe assure (qemu, pty, osmocon, l2), et le banc DSP est
    # lance ensuite avec ses cinq etapes. --no-attach est indispensable : sans
    # lui le fork s'attache a tmux et on ne reprend jamais la main.
    #
    # Les reglages du banc viennent du banc, pas d'un choix de style
    # (c54x_exe/run_real.sh les porte depuis le 2026-09-21) :
    #   INSNS=200000             plafond d'instructions par trame (voir plus bas)
    #   LOCKSTEP=1               QEMU n'avance la trame que quand le DSP a fini
    #                            la precedente ; sans lui il saute 3 trames sur 4
    #   IQ=none                  pas de cellule synthetique : les bursts
    #                            viennent de la vraie BTS, par pont.py
    #   CALYPSO_BSP_STREAM=1     un burst TS0 par trame, dans l'ordre des FN
    #   CALYPSO_RHEA_DMA_XFER=1  sans lui la page API n'est jamais remplie :
    #                            pas de SB, pas de BCCH, pas de SI
    # Tout reste surchargeable : on ne pose que ce que l'operateur n'a pas dit.
    export MODE=dsp PONT=1
    # [2026-10-19] --osmo-trx : transceiver a la place de mobile, pas de pont
    # (c54x_exe/run.sh force PONT=0 lui-meme ; on le dit ici pour le dry-run).
    if [ "${OSMO_TRX:-0}" = 1 ]; then export OSMO_TRX PONT=0; fi
    # [2026-09-23] Trace asm de l'ARM, lue par c54x_exe/run.sh (etape 2, QEMU -d).
    export ASSEMBLY_LOGS="${ASSEMBLY_LOGS:-0}" ASSEMBLY_LOGS_FLAGS ASSEMBLY_LOGS_FILTRE
    : "${IQ:=none}";        export IQ
    # [2026-09-23] 60000 -> 200000, puis 120000 (TCH mesure jusqu a 87000 insn/trame). Le budget n'est qu'un
     ↪  plafond (le DSP
    # s'arrete a son IDLE, c54x_exe/src/pont.c jouer_trame) ; a 60000 le
    # decodage TCH debordait sur la trame suivante (« insn=58524 occupe »), le
    # Viterbi de la parole (0x9a7c, ST TRN,*AR6+ jusqu'a 0x2d23) passait sur
    # data[0x2d0d] apres sa reinitialisation par 0xe68a, et le CALLD 0xea0c
    # ecrasait SP (« SP-CORRUPT pc=0xeac9 »). Le vrai C54x a ~480000 cycles par
    # trame. Capture gdb du 2026-09-23 15:07.
    # [2026-10-22] Magnetite (--ms-magnetite) : 480000. Le firmware TI en TCH demande bien plus que
    # l'osmocom : a 120000 la file des taches du DSP debordait (d_error_status 20h/28h), la L1 voyait
    # ses resultats en retard (« > COM »), relancait le DSP et l'appel au 600 tombait (Assignment
    # Failure). Rejeu tools/rejeu_banc de l'appel de 10:58 : 120000 -> ticks non finis des fn%26=10,
    # 240000 -> 5, 480000 -> 0. Le vrai C54x a ~480000 cycles par trame.
    if [ -n "${MS_MAGNETITE:-}" ]; then : "${INSNS:=480000}"; fi
    : "${INSNS:=120000}";   export INSNS
    : "${LOCKSTEP:=1}";     export LOCKSTEP
    : "${CALYPSO_BSP_STREAM:=1}";    export CALYPSO_BSP_STREAM
    : "${CALYPSO_RHEA_DMA_XFER:=1}"; export CALYPSO_RHEA_DMA_XFER
    # [2026-10-18] TCH-VOICE EN l1phy, PLUS EN gapk. Le profil ROM de
    # /etc/default/osmo-banc (scripts/dsp-rom-max.sh : BANC_VOCODEUR=rom,
    # CALYPSO_VSP=1 et les drapeaux d'emulateur du vocodeur) n'etait lu que par
    # osmo-banc.service. Lance a la main, start-native.sh passait run.sh sans
    # BANC_VOCODEUR, et le mobile tournait sur mobile_pont.cfg brut, en gapk.
    # On charge donc le profil ici - l'environnement gagne toujours - puis
```

```
# l1phy par defaut : run.sh en derive mobile_pont_rom.cfg (config_vocodeur_rom).
# BANC_VOCODEUR=hote pour revenir a gapk.
if [ -r /etc/default/osmo-banc ]; then
    while IFS='=' read -r _k _v; do
        case "$_k" in ''|\#*|*[!A-Za-z0-9_]*) continue ;; esac
        [ -n "${!_k+x}" ] || export "$_k=$_v"
    done < /etc/default/osmo-banc
    unset _k _v
fi
: "${BANC_VOCODEUR:=rom}"; export BANC_VOCODEUR
# [2026-10-22] --gapk avec --ms-magnetite : pas de mobile L23 pour GAPK, la voix est codee par le pont
# (pont/voix_hote.py, libgsm) ; le DSP ne fait plus que le canal (CALYPSO_VSP=0).
if [ "$BANC_VOCODEUR" = hote ] && [ -n "${MS_MAGNETITE:-}" ]; then
    : "${PONT_VOIX_HOTE:=1}"; export PONT_VOIX_HOTE
fi
: "${CALYPSO_VSP:=1}";     export CALYPSO_VSP
# [2026-10-18] ET LE RESTE DU PROFIL --vocodeur-rom, MEME SANS
# /etc/default/osmo-banc. Sur acer (10.99.0.2) le fichier n'existait pas :
# BANC_VOCODEUR=rom et CALYPSO_VSP=1 venaient d'ici, mais pas les drapeaux
# d'emulateur du codec FR ni PONT_UL_RAW_TCH. La ROM recevait bien le micro
# (crete UL 2200 sur le port VSP) et codait... la trame de silence
# (pont : « TCH ul rom P=0 »), l'echo test du 600 restait muet. Memes
# valeurs que scripts/dsp-rom-max.sh profile --vocodeur-rom (le defaut).
if [ "$BANC_VOCODEUR" = rom ]; then
    for _kv in CALYPSO_NORM_TC=0 CALYPSO_C54X_BCTC_NORM=1 PONT_UL_RAW_TCH=1 \
               CALYPSO_C54X_BCD_CARRY=1 CALYPSO_C54X_OVM_LOGIQUE=1 CALYPSO_MAC=1 \
               CALYPSO_C54X_BCD_TC=1 CALYPSO_C54X_BCTC_COMBO=1; do
        _k="${_kv%%=*}"; [ -n "${!_k+x}" ] || export "$_kv"
    done
    unset _kv _k
fi
RUN_ARGS+=(--skip "$DSP_MODULES_RETIRES" --no-attach)
if [ $DRY -eq 1 ]; then
    say_begin "Transmission a run.sh puis au banc DSP"
    say_end " -- " "$C_DIM" "Transmission a run.sh puis au banc DSP" "dry-run"
    printf '  1. CALYPSO_PROFILE=%s bash %s %s\n' "$CALYPSO_PROFILE" "$RUN_SH" "${RUN_ARGS[*]}"
    printf '  2. MODE=dsp PONT=%s IQ=%s INSNS=%s LOCKSTEP=%s BANC_VOCODEUR=%s%s%s bash %s\n' \
        "$PONT" "$IQ" "$INSNS" "$LOCKSTEP" "$BANC_VOCODEUR" \
        "${QEMU_ICOUNT:+ QEMU_ICOUNT=$QEMU_ICOUNT}" \
        "$([ "${ASSEMBLY_LOGS:-0}" = 1 ] && echo " ASSEMBLY_LOGS=1")" "$BANC_DSP"
    [ "${OSMO_TRX:-0}" = 1 ] && printf '     OSMO_TRX=1 : trx.highram.elf + transceiver --no-sync (pas de
     ↪  mobile, pas de pont) ; %s\n' "$(trx_ts_desc)"
    [ "${OSMO_TRX_TS1:-0}" = 1 ] && printf '     OSMO_TRX_TS1=1 : INSTANCE=3 OSMO_TRX_PHONE=2 bash %s   (le 2e
     ↪  Calypso TRX, TS1, avant le transceiver)\n' "$BANC_DSP"
    # [2026-10-21] BTS classique (--bts-classic ou defaut) : le mobile est sur le pont, Magnetite par
     ↪  FIRMWARE_FLASH
    [ "${OSMO_TRX:-0}" != 1 ] && printf '     BTS classique : osmo-bts-trx <-> pont_dsp.py ; mobile %s\n' \
        "$([ -n "${MS_MAGNETITE:-}" ] && echo "Magnetite (FIRMWARE_FLASH=$MAGNETITE_FLASH)" || echo
         ↪  "osmocom-bb (mobile + layer1)")"
    [ "${OSMO_TRX:-0}" = 1 ] && [ "${MS_AIR:-0}" = 1 ] && printf '  3. INSTANCE=2 AIR=1 MODE=dsp %sbash %s
     ↪  (le mobile par l air, quand le firmware trx est en mode BTS)\n'
     ↪  "${MS_MAGNETITE:+FIRMWARE_FLASH=$MAGNETITE_FLASH }" "$BANC_DSP"
    exit 0
fi
# [2026-10-24] Le mobile de l'instance 1 lit mobile_pont.cfg tel quel (run.sh ne
# reecrit le stick que pour les instances 2+) : gabarit a 514, cellule du plan a 1
# dans les conteneurs du multi -> FBSB result=255 en boucle. On l'aligne ici.
_cfg_pont="$(dirname "$BANC_DSP")/mobile_pont.cfg"
_arfcn_pont="${PLAN_ARFCN:-${MS_ARFCN1:-${ARFCN:-1}}}"
if [ -w "$_cfg_pont" ] && [[ "$_arfcn_pont" =~ ^[0-9]+$ ]]; then
    sed -i "s|^\([[:space:]]*\)stick [0-9]*\$|\1stick ${_arfcn_pont}|" "$_cfg_pont"
fi
# [2026-10-24] Meme gabarit, meme trou pour la SIM : imsi/ki/rplmn restaient
# ceux de l'op1 (001010001000001, ki ..01 01). Dans le conteneur de l'op2,
# la HLR ne connait cette IMSI que sans donnees d'auth -> LU reject cause 2.
# On reprend la carte de mobile.cfg, celle que start-docker genere par operateur.
```

```
    _sim_src="/root/.osmocom/bb/mobile.cfg"
    if [ -w "$_cfg_pont" ] && [ -r "$_sim_src" ]; then
        for _k in imsi "ki comp128" rplmn; do
            _l="$(grep -m1 -E "^[[:space:]]+${_k} " "$_sim_src")" || continue
            _l="${_l//|/\\|}"
            sed -i "s|^[[:space:]]\+${_k} .*\$|${_l}|" "$_cfg_pont"
        done
    fi
    unset _cfg_pont _arfcn_pont _sim_src _k _l
    audio_start
    say_begin "Transmission a run.sh (sans $DSP_MODULES_RETIRES)"
    if ! env CALYPSO_PROFILE="$CALYPSO_PROFILE" bash "$RUN_SH" "${RUN_ARGS[@]}"; then
        say_end " KO " "$C_KO" "Transmission a run.sh" "le plan du fork a echoue, banc DSP non lance"
        exit 1
    fi
    say_end " OK " "$C_OK" "Transmission a run.sh" "profil=$CALYPSO_PROFILE, chaine Calypso laissee au banc DSP"
    say_begin "Banc DSP C54x"
    say_end " OK " "$C_OK" "Banc DSP C54x" "$BANC_DSP  (INSNS=$INSNS LOCKSTEP=$LOCKSTEP IQ=$IQ$([ "${OSMO_TRX:-0}"
     ↪  = 1 ] && echo " OSMO_TRX=1 : trx.highram.elf + transceiver, $(trx_ts_desc)"))"
    # Hand-off total : ce processus devient c54x_exe/run.sh (cinq etapes, puis
    # il rend la main ; les journaux restent dans /tmp/c54x-pont). MODE=dsp est
    # deja exporte juste au-dessus, il ecrase celui du profil.
    # [2026-10-14] --sap : la copie « sim sap » de mobile_pont.cfg, au banc seul.
    # [2026-10-20] --bts --ms : l'instance 1 (la radio de la BTS) d'abord, puis, quand le
    # firmware trx est passe en mode BTS (osmo-bts-trx a envoye POWERON : « BTS MODE: 7 514 »
    # dans osmocon.log), l'instance 2 (le mobile + l'air), qui prend la main.
    if [ "${OSMO_TRX:-0}" = 1 ] && [ "${MS_AIR:-0}" = 1 ]; then
        bash "$BANC_DSP" || exit 1
        say_begin "Mobile par l'air (instance 2)"
        _ok=0
        for _i in $(seq 1 "${OSMO_TRX_BTS_ATTENTE:-150}"); do
            grep -aq 'BTS MODE: [0-9]* [1-9]' /tmp/c54x-pont/osmocon.log 2>/dev/null && { _ok=1; break; }
            sleep 1
        done
        if [ "$_ok" = 1 ]; then
            say_end " OK " "$C_OK" "Mobile par l'air (instance 2)" "firmware trx en mode BTS apres ${_i}s ;
             ↪  INSTANCE=2 AIR=1 $BANC_DSP"
        else
            say_end " !! " "$C_KO" "Mobile par l'air (instance 2)" "le firmware trx n'est pas en mode BTS
             ↪  (osmo-bts-trx absente ?) ; on lance quand meme"
        fi
        sleep 10
        # [2026-10-20 21:20] OSMO_TRX=0 explicite : OSMO_TRX=1 est exporte (le TRX, c'est l'instance 1) et
        # l'instance 2 en heritait -> firmware trx + second transceiver, « port UDP 5700 deja pris ».
        # Le mobile par l'air est le layer1 d'origine (FIRMWARE_ELF du profil retire : c'est le
        # layer1.highram.elf qui a fait le LU et l'appel du 20/10 21:08 ; FIRMWARE_SRC explicite garde la main).
        # [2026-10-21] --ms-magnetite : FIRMWARE_FLASH (exporte au parsing) remplace le layer1 pour l'instance 2
        exec env -u FIRMWARE_ELF -u FIRMWARE_BIN ${FIRMWARE_SRC:+FIRMWARE_ELF="$FIRMWARE_ELF"} \
            ${MS_MAGNETITE:+FIRMWARE_FLASH="$MAGNETITE_FLASH"} \
            INSTANCE=2 AIR=1 OSMO_TRX=0 ${SAP_DSP_MOBILE_CFG:+MOBILE_CFG="$SAP_DSP_MOBILE_CFG"} bash "$BANC_DSP"
    fi
    exec env ${MS_MAGNETITE:+FIRMWARE_FLASH="$MAGNETITE_FLASH"}
     ↪  ${SAP_DSP_MOBILE_CFG:+MOBILE_CFG="$SAP_DSP_MOBILE_CFG"} bash "$BANC_DSP"
fi
```

3.384 /opt/GSM/osmo-operator/startup_scripts/15-fin.sh

594 octets, 13 lignes → 13 lignes

```
# 15-fin.sh - partie de start.sh (lignes 3443-3454 de start-native.sh d'origine)
# Sourcee par start.sh dans le shell principal : meme ordre, memes variables.
[ $DRY -eq 1 ] || audio_start
say_begin "Transmission a run.sh"
if [ $DRY -eq 1 ]; then
```

```
    say_end " -- " "$C_DIM" "Transmission a run.sh" "dry-run"
    printf '  commande : CALYPSO_PROFILE=%s bash %s %s\n' \
        "$CALYPSO_PROFILE" "$RUN_SH" "${RUN_ARGS[*]}"
    exit 0
fi
say_end " OK " "$C_OK" "Transmission a run.sh" "profil=$CALYPSO_PROFILE"
# Hand-off total : ce processus devient run.sh
exec bash "$RUN_SH" "${RUN_ARGS[@]}"
```

3.385 /opt/GSM/osmo-operator/tools/at-cmd.py

5175 octets, 141 lignes → 141 lignes

```
#!/usr/bin/env python3
# at-cmd.py - PARLER A LA MAIN AU MODEM AT.
#
# Le pendant de tools/osmo-ril-atmodem.py : celui-ci TIENT le role du modem
# (adosse a oFono), celui-la tient le role du RIL. C est l outil pour verifier,
# sans Android et sans rild, que le modem virtuel repond ce qu il faut - et
# c est aussi ce qu on branche sur un VRAI modem (un EC25 sur /dev/ttyUSB2)
# pour comparer les reponses.
#
#   at-cmd.py                      ouvre une session interactive
#   at-cmd.py AT+CREG?             une commande, la reponse, on sort
#   at-cmd.py -d /dev/ttyUSB2 AT+CSQ   sur un modem physique
#   at-cmd.py --sms 100102 "salut"     le dialogue AT+CMGS complet (avec le ^Z)
#   at-cmd.py --call 100102            ATD100102;
#
# Par defaut on parle au modem virtuel, dont le lien est publie dans
# /run/osmo-ril/at-pty.
import argparse
import os
import sys
import termios
import time

DEV = os.environ.get("OSMO_AT_DEV", "/run/osmo-ril/at-pty")


def open_port(dev, baud=115200):
    fd = os.open(dev, os.O_RDWR | os.O_NOCTTY)
    try:                                  # un pty n a pas de vitesse, un tty si
        attrs = termios.tcgetattr(fd)
        attrs[0] = attrs[1] = attrs[3] = 0          # iflag, oflag, lflag : brut
        attrs[2] = termios.CS8 | termios.CREAD | termios.CLOCAL
        speed = getattr(termios, "B%d" % baud, termios.B115200)
        attrs[4] = attrs[5] = speed
        termios.tcsetattr(fd, termios.TCSANOW, attrs)
    except termios.error:
        pass
    return fd


def send(fd, line, wait=1.0, quiet=False):
    """Envoie une commande et rend tout ce qui arrive jusqu au silence."""
    os.write(fd, (line + "\r\n").encode())
    return read_until(fd, wait, quiet)


def read_until(fd, wait, quiet=False):
    """Lit jusqu a OK/ERROR, ou jusqu au bout du delai."""
    import select
    out, end = "", time.time() + wait
    while time.time() < end:
```

```
        r, _, _ = select.select([fd], [], [], 0.1)
        if not r:
            continue
        try:
            chunk = os.read(fd, 4096).decode(errors="replace")
        except OSError:
            break
        if not chunk:
            break
        out += chunk
        if not quiet:
            sys.stdout.write(chunk)
            sys.stdout.flush()
        # Une reponse finale : inutile d attendre le reste du delai.
        for fin in ("\r\nOK\r\n", "\r\nERROR\r\n", "+CME ERROR", "+CMS ERROR"):
            if out.endswith(fin) or fin in out[-40:]:
                return out
    return out


def main():
    ap = argparse.ArgumentParser(description="client AT (modem virtuel ou vrai modem)")
    ap.add_argument("cmd", nargs="*", help="commande AT (sinon : session interactive)")
    ap.add_argument("-d", "--device", default=DEV, help="port (defaut %s)" % DEV)
    ap.add_argument("-t", "--timeout", type=float, default=2.0, help="delai de lecture")
    ap.add_argument("--sms", nargs=2, metavar=("NUMERO", "TEXTE"),
                    help="envoie un SMS : AT+CMGF=1, AT+CMGS, le texte, puis Ctrl-Z")
    ap.add_argument("--call", metavar="NUMERO", help="ATD<numero>;")
    ap.add_argument("--hangup", action="store_true", help="ATH")
    ap.add_argument("--status", action="store_true",
                    help="l etat en une passe : CPIN, CFUN, CREG, COPS, CSQ, CLCC")
    a = ap.parse_args()

    if not os.path.exists(a.device):
        print("[at-cmd] %s introuvable - osmo-ril-atmodem.py tourne ?" % a.device,
              file=sys.stderr)
        return 1
    try:
        fd = open_port(a.device)
    except OSError as e:
        print("[at-cmd] %s : %s" % (a.device, e), file=sys.stderr)
        return 1

    try:
        if a.status:
            for c in ("AT", "AT+CPIN?", "AT+CFUN?", "AT+CREG?", "AT+COPS?",
                      "AT+CSQ", "AT+CLCC"):
                print("--- %s" % c)
                send(fd, c, a.timeout)
            return 0
        if a.sms:
            num, text = a.sms
            send(fd, "AT+CMGF=1", a.timeout)
            print('--- AT+CMGS="%s"' % num)
            os.write(fd, ('AT+CMGS="%s"\r\n' % num).encode())
            read_until(fd, 1.0)                     # l invite « > »
            os.write(fd, (text + "\x1a").encode())  # le texte puis Ctrl-Z
            read_until(fd, max(a.timeout, 5.0))
            return 0
        if a.call:
            send(fd, "ATD%s;" % a.call, a.timeout)
            return 0
        if a.hangup:
            send(fd, "ATH", a.timeout)
            return 0
        if a.cmd:
```

```
            send(fd, " ".join(a.cmd), a.timeout)
            return 0

        # session interactive
        print("[at-cmd] %s - une commande par ligne, Ctrl-D pour sortir" % a.device)
        while True:
            try:
                line = input("AT> ").strip()
            except (EOFError, KeyboardInterrupt):
                print()
                return 0
            if not line:
                continue
            if line.lower() in ("quit", "exit"):
                return 0
            if not line.upper().startswith("AT"):
                line = "AT" + line
            send(fd, line, a.timeout)
    finally:
        os.close(fd)


if __name__ == "__main__":
    sys.exit(main())
```

3.386 /opt/GSM/osmo-operator/tools/conky-osmo-status.sh

30690 octets, 522 lignes → 522 lignes

```
#!/bin/bash
# conky-osmo-status.sh - les lignes du Conky du banc (configs/conky/osmo-conky.conf)
#   role | vue | net | hub | core | radio | lte | subs | services
# Chaque sortie est du texte conky (execpi) : ${color2} vert, ${color3} rouge,
# ${color4} jaune. Rapide et sans dependance dure : ce qui manque s affiche "-".
set -u
# [2026-10-12] LES GLYPHES EN DEJAVU SANS, LE TEXTE EN UBUNTU. Le Conky est
# tout en Ubuntu (osmo-conky.conf), qui n a ni ● ○ ni ↓ ↑ ◀ ▶ ☺ : rendus en
# carres. Chaque glyphe est donc enveloppe d un passage en DejaVu Sans (FS)
# puis d un retour a la police par defaut (FU = ${font}, donc Ubuntu).
# A LA MEME TAILLE (9) que la police par defaut, et ce n est pas un detail :
# Conky calcule la hauteur de sa fenetre avec la police par defaut, mais
# DESSINE chaque ligne a la hauteur de la plus haute police rencontree. Avec
# DejaVu en 8 (ou 7, ou Mono), chaque ligne a pastilles se dessinait 6 px plus
# bas que prevu : au bout de vingt lignes, le bas du Conky sortait de la
# fenetre (teste : 21 px par ligne pour 15 calcules). En 9, memes metriques,
# la fenetre tombe juste.
FS="\${font ${OSMO_CONKY_FS:-DejaVu Sans:size=9}}"; FU='${font}'
OK="${FS}\${color2}●\${color}${FU}"; KO="${FS}\${color3}○\${color}${FU}"; WARN="${FS}\${color4}●\${color}${FU}"
# un smiley pour CHAQUE arret du banc (op1, op2, op3, hub, 4G) : il sourit en
# vert quand l arret tourne, fait la tete en gris quand il est a l arret
SMILE="${FS}\${color2}☺\${color}${FU}"; SAD="${FS}\${color5}☹\${color}${FU}"
C1='${color1}'; C='${color}'; C2='${color2}'; AR='${alignr}'
have() { command -v "$1" >/dev/null 2>&1; }
# ── pgrep -x COMPARE AU NOM TRONQUE A 15 CARACTERES ──────────────────────
# [2026-09-04] « pgrep -x osmo-sip-connector » ne matche JAMAIS : le noyau ne
# garde que 15 caracteres de nom (comm), soit « osmo-sip-connec ». Idem pour
# proto-smsc-daemon (« proto-smsc-daem »). SIP et SMSC restaient rouges avec les
# deux demons en marche. On tronque donc le motif comme le noyau tronque le nom.
# ── L OPERATEUR CHOISI DANS L ENCART ─────────────────────────────────────
# [2026-09-04] tools/osmo-panel.py (fleches < >) ecrit /run/osmo-fft/operator :
# OP=, MODE=native|docker, IP=, NAME=osmo-operator-N. Sur un conteneur, chaque
# sonde (pgrep, port, sqlite) s execute DEDANS (docker exec) : le Conky suit
# l encart. Sans fichier ou en natif : la machine, comme avant.
OP_FILE="${OSMO_FFT_DIR:-/run/osmo-fft}/operator"
```

```
OP_ID=1; OP_MODE=native; OP_NAME=""
if [ -r "$OP_FILE" ]; then
    OP_ID="$(sed -n 's/^OP=//p' "$OP_FILE" | head -1)"; OP_ID="${OP_ID:-1}"
    OP_MODE="$(sed -n 's/^MODE=//p' "$OP_FILE" | head -1)"; OP_MODE="${OP_MODE:-native}"
    OP_NAME="$(sed -n 's/^NAME=//p' "$OP_FILE" | head -1)"
fi
# in_op CMD... : execute la commande ici, ou dans le conteneur de l operateur.
if [ "$OP_MODE" = docker ] && [ -n "$OP_NAME" ]; then
    in_op() { docker exec "$OP_NAME" "$@" 2>/dev/null; }
else
    in_op() { "$@"; }
fi
port_open() {
    if [ "$OP_MODE" = docker ]; then in_op timeout 1 bash -c "echo >/dev/tcp/127.0.0.1/$1" 2>/dev/null
    else timeout 1 bash -c "echo >/dev/tcp/127.0.0.1/$1" 2>/dev/null; fi
}
alive_x() { in_op pgrep -x "${1:0:15}" >/dev/null 2>&1; }
# ── LE BANC EST-IL DEBOUT ? ──────────────────────────────────────────
# [2026-09-04] Sans banc, les sections « Coeur GSM », « Radio / MS », « Abonnes »
# et « Services » affichaient TRENTE pastilles rouges - un mur d alarmes pour
# dire une seule chose : rien ne tourne. On ne lit plus rien la-dedans, et
# surtout on ne voit plus ce qui, LUI, compte quand le banc est a l arret : le
# reseau, l espace disque, la machine.
#
# Quand la pile est a terre, ces sections ne rendent donc RIEN : le Conky se
# replie sur la machine, et les titres restent comme reperes. La section Reseau
# est la seule qui parle TOUJOURS - elle ne depend d aucun operateur, et c est
# precisement ce qu on regarde quand on cherche pourquoi le banc ne monte pas.
#
# Le critere : le STP ou le BSC de l operateur affiche. Ce sont les deux
# processus qui existent des que le coeur est monte, dans les deux mondes
# (natif et conteneur) - et in_op les cherche au bon endroit.
banc_debout() { alive_x osmo-stp || alive_x osmo-bsc || alive_x osmo-msc; }
# Motif sur la ligne de commande complete (scripts python, chemins).
alive_f() { in_op pgrep -f "$1" >/dev/null 2>&1; }

# ── LA MATRICE : AFFICHAGE (instantane) ET MESURE (en tache de fond) ─────
MATRIX_FILE="${OSMO_FFT_DIR:-/run/osmo-fft}/ss7-matrix"
MATRIX_LOCK="${OSMO_FFT_DIR:-/run/osmo-fft}/.ss7-matrix.lock"
MATRIX_TTL="${OSMO_MATRIX_TTL:-300}"

matrice() {
    local now age=999999 mtime=0
    now="$(date +%s)"
    [ -f "$MATRIX_FILE" ] && {
        mtime="$(stat -c %Y "$MATRIX_FILE" 2>/dev/null || echo 0)"
        age=$(( now - mtime ))
    }
    # ── UNE MESURE D AVANT LE BANC N EST PAS UNE MESURE ──────────────────
    # [2026-09-07] « SS7 KO - 9 fail » affiche pendant cinq minutes sur un banc
    # parfaitement monte : la mesure datait du DEMARRAGE, quand osmo-multi
    # n avait pas encore lance osmo-operator-2/3 - les neuf tests des deux
    # operateurs docker echouaient parce que leurs conteneurs n existaient pas
    # encore. Le cache la gardait ensuite jusqu au bout du TTL (5 min).
    # Le cache ne doit pas survivre a ce qu il decrit : toute mesure anterieure
    # au dernier demarrage d osmo-banc ou d osmo-multi est perimee, quel que
    # soit son age. `systemctl show` repond sans privilege.
    local t u
    for u in osmo-banc.service osmo-multi.service; do
        t="$(systemctl show -p ActiveEnterTimestamp --value "$u" 2>/dev/null)"
        [ -n "$t" ] || continue
        t="$(date -d "$t" +%s 2>/dev/null)" || continue
        [ -n "$t" ] && [ "$mtime" -lt "$t" ] && age=$(( MATRIX_TTL + 1 ))
    done
    # Perime (ou jamais mesure) : on lance la mesure DETACHEE et on rend la
    # main tout de suite. Le verrou est un mkdir : atomique, et il disparait
```

```
    # avec le repertoire meme si la mesure est tuee (trap).
    local encours=0
    if [ "$age" -gt "$MATRIX_TTL" ]; then
        ( setsid "$0" --refresh-matrix >/dev/null 2>&1 & ) 2>/dev/null
        encours=1
    fi
    [ -d "$MATRIX_LOCK" ] && encours=1
    if [ -s "$MATRIX_FILE" ]; then
        cat "$MATRIX_FILE"
        # ss7_check.sh met une bonne minute : on DIT que ce qu on lit est vieux
        # et qu une mesure tourne, plutot que de laisser prendre un KO perime
        # pour l etat courant.
        if [ "$encours" = 1 ]; then
            echo "  \${color4}mesure en cours\${color} - affiche : il y a $(( age / 60 ))m$(( age % 60 ))s"
        fi
    else
        echo "  ${C1}matrice SS7${C} : premiere mesure en cours..."
    fi
}

refresh_matrice() {
    mkdir "$MATRIX_LOCK" 2>/dev/null || return 0     # une mesure suffit
    trap 'rmdir "$MATRIX_LOCK" 2>/dev/null' EXIT
    local dir out
    dir="$(cd "$(dirname "$0")/.." && pwd)"
    [ -x "$dir/checks/ss7_check.sh" ] || return 0
    # NO_COLOR + suppression des sequences ANSI : on repeint aux couleurs du
    # Conky (${color1..4}), qui ne comprend pas les codes d un terminal.
    out="$(cd "$dir" && NO_COLOR=1 timeout 240 ./checks/ss7_check.sh 2>/dev/null \
           | sed -e 's/\x1b\[[0-9;]*m//g')"
    [ -n "$out" ] || return 0
    {
        # Le tableau : la ligne d en-tete (les colonnes OpN) puis une ligne par
        # operateur, jusqu a la ligne vide qui clot le bloc.
        printf '%s\n' "$out" \
        | awk '/MATRICE DE CONNECTIVITE/{f=1; next}
               f && /^[[:space:]]*$/ && seen {exit}
               f && /Op/ {seen=1; print}' \
        | sed -e 's/[[:space:]]*$//' \
              -e 's/\bself\b/${color1}self${color}/g' \
              -e 's/\bvia\b/${color2}via${color}/g' \
              -e 's/\bFAIL\b/${color3}FAIL${color}/g'
        # Le bilan, tel que ss7_check le formule.
        printf '%s\n' "$out" | awk '/SS7 (OK|DEGRADE|KO)/{sub(/^[[:space:]]+/,""); print; exit}' \
        | sed -e 's/^SS7 OK/${color2}SS7 OK${color}/' \
              -e 's/^SS7 DEGRADE/${color4}SS7 DEGRADE${color}/' \
              -e 's/^SS7 KO/${color3}SS7 KO${color}/' \
              -e 's/\([0-9]\+\) pass/${color2}\1 pass${color}/' \
              -e 's/\([0-9]\+\) fail/${color3}\1 fail${color}/' \
              -e 's/\([0-9]\+\) warn/${color4}\1 warn${color}/'
    } > "$MATRIX_FILE.tmp" 2>/dev/null && mv -f "$MATRIX_FILE.tmp" "$MATRIX_FILE"
}

case "${1:-}" in
--refresh-matrix) refresh_matrice; exit 0 ;;
role)
    r="$(awk -F= '/^OSMO_ROLE=/{print $2}' /etc/osmo-role 2>/dev/null)"
    n="$(awk -F= '/^OSMO_NODE=|^NODE_ID=/{print $2}' /etc/osmo-role 2>/dev/null | head -1)"
    h="$(awk -F= '/^OSMO_HUB_IP=/{print $2}' /etc/osmo-role 2>/dev/null)"
    live=""; [ -e /run/live/rootfs/filesystem.squashfs ] && live=" \${color4}LIVE${C}"
    opl=""; [ "$OP_MODE" = docker ] && opl=" \${color2}op $OP_ID${C} ($OP_NAME)"
    echo "${r:-operateur}${n:+ noeud $n}${h:+ · hub $h}$live$opl" ;;
net)
    ip -o -4 addr show up 2>/dev/null | awk '$2!="lo" && $2!~/^(veth|br-|docker|apn)/ {print $2, $4}' | sort -u |
     ↪  head -5 \
    | while read -r ifc addr; do
```

```
        printf '${color6}%s${color} ${alignr}${color2}${downspeedf %s}${color} K/s %s  ${color4}${upspeedf
        ↪  %s}${color} K/s %s\n%-8s ${color1}%s${color}\n' \
            "$ifc" "$ifc" "${FS}\${color2}↓\${color}${FU}" "$ifc" "${FS}\${color4}↑\${color}${FU}" "" "$addr"
      done
    # Cette section ne se replie JAMAIS (voir banc_debout) : quand le banc est a
    # l arret, c est elle qu on lit. Si le filtre ci-dessus n a rien laisse
    # passer - que des veth/docker, ou aucune adresse - on le dit, plutot que
    # de rendre une section vide qu on prendrait pour une panne du Conky.
    if [ -z "$(ip -o -4 addr show up 2>/dev/null | awk '$2!="lo" && $2!~/^(veth|br-|docker|apn)/')" ]; then
        if [ -n "$(ip -o -4 addr show up 2>/dev/null | awk '$2!="lo"')" ]; then
            echo "${WARN} aucune carte physique - seulement les interfaces du banc"
        else
            echo "${WARN} aucune interface avec adresse"
        fi
    fi ;;
core)
    banc_debout || exit 0
    # nom:processus:port VTY
    for e in HLR:osmo-hlr:4258 MSC:osmo-msc:4254 BSC:osmo-bsc:4242 STP:osmo-stp:4239 \
             MGW:osmo-mgw:4243 SGSN:osmo-sgsn:4245 GGSN:osmo-ggsn:4260 PCU:osmo-pcu:4240 \
             BTS:osmo-bts-trx:4241 SIP:osmo-sip-connector:4256 SMSC:proto-smsc-daemon:0 PBX:asterisk:0; do
        IFS=: read -r name proc port <<< "$e"
        if alive_x "$proc"; then
            if [ "$port" != 0 ] && ! port_open "$port"; then s="$WARN"; else s="$OK"; fi
        else s="$KO"; fi
        printf '%s %-5s' "$s" "$name"
        i=$((${i:-0}+1)); [ $((i % 4)) -eq 0 ] && echo
    done
    # Pas de ligne vide en trop quand les douze pastilles tombent juste.
    [ $((i % 4)) -eq 0 ] || echo ;;
radio)
    banc_debout || exit 0
    phy="-"
    pgrep -f 'qemu-system-arm' >/dev/null 2>&1 && phy="qemu (Calypso emule)"
    pgrep -f 'fake_trx.py' >/dev/null 2>&1 && phy="faketrx"
    pgrep -x virtphy >/dev/null 2>&1 && phy="virtphy"
    pgrep -f 'pont/pont(_dsp)?\.py' >/dev/null 2>&1 && phy="$phy + pont"
    echo "PHY ${C1}${phy}${C}"
    # ── CE QUE CHAQUE PASTILLE CHERCHE VRAIMENT ──────────────────────
    # [2026-09-04] Le banc de l'ISO tourne en faketrx + pont : il n'y a AUCUN
    # binaire osmo-trx, le transceiver de la BTS est pont/pont.py et celui du
    # MS#2 fake_trx.py. gapk n'existe que sous le nom osmo-gapk, lance par
    # gapk-start.sh le temps d'un appel ; hors appel, le veilleur
    # (gapk-start.sh auto) EST l'etat nominal. gr-gsm est remplace par
    # qosmo-grgsm (gsm_sniff.py) : grgsm_decode n'est pas installe. Les trois
    # restaient rouges sur un banc en parfait etat.
    s=""
    { alive_f 'osmo-trx' || alive_f 'pont/pont(_dsp)?\.py' || alive_f 'fake_trx\.py'; } && s="$s$OK TRX     " ||
    ↪  s="$s$KO TRX     "
    alive_x mobile                  && s="$s$OK MOBILE  " || s="$s$KO MOBILE  "
    alive_x trxcon                  && s="$s$OK TRXCON  " || s="$s$KO TRXCON  "
    alive_f 'qemu-system-arm'       && s="$s$OK QEMU    " || s="$s$KO QEMU    "
    if alive_x osmo-gapk; then s="$s$OK GAPK    "
    elif alive_f 'gapk-start\.sh'; then s="$s$OK GAPK    "
    else s="$s$KO GAPK    "; fi
    { alive_f 'grgsm_(decode|livemon)' || alive_f 'gsm_sniff\.py' || alive_x qosmo-grgsm; } \
        && s="$s$OK GRGSM  " || s="$s$KO GRGSM  "
    echo "$s"
    arfcn="$(in_op cat /etc/osmocom/osmo-bsc.cfg 2>/dev/null | awk '/^ *arfcn /{print $2; exit}')"
    plmn="$(in_op cat /etc/osmocom/osmo-msc.cfg 2>/dev/null | awk '/network country code/{c=$4} /mobile network
    ↪  code/{n=$4} END{if(c) print c"-"n}')"
    a5="$(in_op cat /etc/osmocom/osmo-msc.cfg 2>/dev/null | awk '/encryption a5/{$1="";$2="";print; exit}' | sed
    ↪  's/^ *//')"
    echo "PLMN ${C1}${plmn:--}${C}  ARFCN ${C1}${arfcn:--}${C}  A5 ${C1}${a5:--}${C}" ;;
subs)
    banc_debout || exit 0
```

```
    db=/var/lib/osmocom/hlr.db
    # Le conteneur n a pas forcement sqlite3 : on lit sa base par une copie.
    # [2026-10-24] Avec son -wal : osmo-hlr est en WAL, les LU n y sont pas encore
    # reportes dans hlr.db (copie seule = « 0 rattaches » alors que tout l est).
    # Copie par utilisateur : celle de root bloquait le conky de nirvana, qui
    # relisait sans le dire la meme copie perimee.
    if [ "$OP_MODE" = docker ]; then
        dbl="/tmp/.conky-hlr-$(id -u)-${OP_NAME}.db"
        rm -f "$dbl-wal" "$dbl-shm"
        if docker cp "$OP_NAME:$db" "$dbl" >/dev/null 2>&1; then
            docker cp "$OP_NAME:$db-wal" "$dbl-wal" >/dev/null 2>&1
            db="$dbl"
        fi
    fi
    # ── L ETAT DU HLR NE SE LIT PAS DANS SA BASE ──────────────────────
    # [2026-09-04] Ce bloc ouvrait hlr.db et affichait « N abonnes · M
    # rattaches » sans jamais regarder si osmo-hlr tournait. La base survit au
    # demon : le tableau de bord montrait donc un HLR allume, et des abonnes
    # « rattaches » a un VLR disparu, des heures apres l arret du banc. On
    # regarde le demon d abord ; la base ne dit que ce qui est PROVISIONNE.
    if ! alive_x osmo-hlr; then
        _n=""
        have sqlite3 && [ -r "$db" ] && \
            _n="$(sqlite3 "$db" 'select count(*) from subscriber;' 2>/dev/null)"
        echo "HLR ${KO} arrete${_n:+ ${AR}${_n} abonnes en base}"
    elif have sqlite3 && [ -r "$db" ]; then
        tot="$(sqlite3 "$db" 'select count(*) from subscriber;' 2>/dev/null)"
        att="$(sqlite3 "$db" "select count(*) from subscriber where vlr_number is not null and vlr_number != '';"
         ↪  2>/dev/null)"
        last="$(sqlite3 "$db" 'select max(last_lu_seen) from subscriber;' 2>/dev/null)"
        echo "HLR ${C1}${tot:-0}${C} abonnes · ${C2}${att:-0}${C} rattaches ${AR}LU ${last:--}"
        sqlite3 "$db" "select imsi, coalesce(msisdn,'-'), case when vlr_number is not null and vlr_number != ''
         ↪  then 1 else 0 end from subscriber order by imsi limit 6;" 2>/dev/null \
        | awk -F'|' -v ok="$OK" -v ko="$KO" '{printf "%s %s ${alignr}%s\n", ($3==1?ok:ko), $1, $2}'
    else
        echo "HLR ${KO} base introuvable (${db})"
    fi ;;
vue)
    # ── LE SELECTEUR : ◄ ARRET ► , ET LA VUE QU IL COMMANDE ──────────
    # [2026-09-10] Le Conky montrait TOUJOURS le meme operateur - celui du
    # fichier - et empilait ses sections 2G les unes sous les autres, sans
    # jamais parler ni du hub inter-STP (sauf trois lignes sous Reseau) ni de
    # la 4G. Il porte maintenant UN selecteur et UNE vue : les arrets sont
    # ceux de « osmo-op --list-vue » (op1, les operateurs docker qui TOURNENT,
    # le hub s il tourne, la 4G si elle est installee) et la vue change avec
    # l arret - sections 2G pour un operateur, matrice SS7 pour le hub, coeur
    # EPC et radio LTE pour la 4G.
    #
    # Conky ne recoit PAS les clics (c est toute la raison d etre de
    # tools/osmo-panel.py) : les fleches ◄ ► se poussent avec celles de
    # l encart, avec Ctrl+AltGr+Gauche/Droite (raccourci pose par l ISO,
    # iso_modules/80-chroot.sh), ou en ligne de commande - « osmo-op --next »
    # et « --prev ».
    _op="${DIR:-/opt/GSM/osmo-operator}/tools/osmo-op.sh"
    [ -x "$_op" ] || _op="/opt/GSM/osmo-operator/tools/osmo-op.sh"
    [ -x "$_op" ] || { echo "${WARN} osmo-op.sh introuvable"; exit 0; }
    mapfile -t VUES < <("$_op" --list-vue 2>/dev/null)
    [ "${#VUES[@]}" -gt 0 ] || { echo "${WARN} aucun arret (osmo-op --list-vue)"; exit 0; }
    # ── L ARRET REGARDE A PU DISPARAITRE SOUS L ECRAN ─────────────────
    # On regardait op2, quelqu un a arrete son conteneur : l arret n existe
    # plus, et le Conky afficherait la vue d un operateur absent. On revient
    # au premier arret - et on l ECRIT, pour que l encart et les fleches
    # suivent au lieu de rester sur un fantome.
    _cur="$OP_ID"; _mode="$OP_MODE"; _connu=0
    for l in "${VUES[@]}"; do
        set -- $l
```

```
        [ "$1" = "$_cur" ] && { _connu=1; _mode="$2"; }
    done
    if [ "$_connu" = 0 ]; then
        set -- ${VUES[0]}
        _cur="$1"; _mode="$2"
        "$_op" --set "$_cur" >/dev/null 2>&1 || true
    fi
    # Le selecteur - seulement s il y a plus d un endroit ou aller.
    if [ "${#VUES[@]}" -gt 1 ]; then
        sel=""
        for l in "${VUES[@]}"; do
            set -- $l
            idx="$1"; mode="$2"; etat="${4:-arrete}"   # $3 = l IP, ou « - »
            case "$mode" in
                interstp) nom="hub" ;;
                lte)      nom="4G" ;;
                *)        nom="op$idx" ;;
            esac
            # [2026-10-12] un smiley devant chaque arret : sourire vert s il
            # tourne, mine grise sinon ; l arret choisi entre ◂ ▸
            [ "$etat" = actif ] && p="$SMILE" || p="$SAD"
            if [ "$idx" = "$_cur" ]; then
                sel="$sel ${FS}\${color4}◂\${color}${FU}$p${C2}$nom${C}${FS}\${color4}▸\${color}${FU}"
            else
                sel="$sel $p${C1}$nom${C}"
            fi
        done
        echo "${sel# }${AR}\${color5}Ctrl+AltGr+${FS}\${color4}◂▸\${color}${FU}"
    fi
    # La vue.
    T='${font Ubuntu:bold:size=10}${color1}'; TF='${font}${color}'
    case "$_mode" in
        interstp)
            echo "${T}Hub inter-STP${TF}"
            "$0" hub ;;
        lte)
            echo "${T}4G / LTE\${color4} Open5GS + srsRAN${TF}"
            "$0" lte ;;
        *)
            # ── PAS DE TITRE SANS RIEN DESSOUS ──────────────────────────
            # Ces trois sections ne rendent RIEN quand la pile est a terre
            # (banc_debout, plus haut) : les titres restaient alors seuls,
            # trois lignes qui se lisent comme un Conky casse. On ne les
            # imprime que s il y a quelque chose a mettre dessous, et le cas
            # « rien du tout » se dit en une ligne - celle qui sert.
            _vide=1
            for e in "Coeur GSM:core" "Radio / MS:radio" "Abonnes (HLR):subs"; do
                _out="$("$0" "${e##*:}" 2>/dev/null)"
                [ -n "$_out" ] || continue
                _vide=0
                echo "${T}${e%%:*}${TF}"
                echo "$_out"
            done
            [ "$_vide" = 0 ] || echo "${C1}banc a l arret${C} - ${C2}./start-native.sh${C}" ;;
    esac
    ;;
ops|hub)
    # ── LE HUB INTER-STP, ET LA MATRICE DE CONNECTIVITE ─────────────────
    # [2026-09-10] Le carrousel des operateurs (◂ op1 op2 ▸) qui ouvrait cette
    # sous-commande est devenu LE SELECTEUR de la vue - voir « vue » ci-dessus.
    # Il ne reste ici que ce qui parle du hub, et c est la vue « hub ».
    # (« ops » reste accepte : une ISO plus ancienne l appelle encore.)
    # Le hub inter-STP : present seulement s il tourne (banc multi uniquement).
    if have docker && docker ps --format '{{.Names}}' 2>/dev/null | grep -qx osmo-inter-stp; then
        _pc="$(awk -F= '/^MULTI_HUB_PC=/{print $2}' /etc/osmocom/osmo-multi.conf 2>/dev/null)"
        _hi="$(awk -F= '/^MULTI_HUB_IP=/{print $2}' /etc/osmocom/osmo-multi.conf 2>/dev/null)"
```

```
        echo "$OK inter-STP ${C1}${_hi:-172.20.0.10}${C} PC ${C1}${_pc:-0.0.0}${C}"
        # ── LA MATRICE DE CONNECTIVITE, EN CACHE ──────────────────────────
        # Elle vient de checks/ss7_check.sh - LA source, celle qui interroge
        # vraiment les VTY et compte les routes. On ne la recalcule pas ici :
        # deux implementations de la meme question finissent toujours par ne
        # plus dire la meme chose, et c est la copie que l on croit.
        #
        # MAIS ss7_check.sh met une bonne minute (31 tests, plusieurs telnet
        # VTY par operateur, chacun avec ses temporisations). Un `execpi` qui
        # l appellerait FIGERAIT LE CONKY entier a chaque rafraichissement -
        # l horloge, le CPU, le reseau, tout s arreterait pendant la mesure.
        # D ou le cache : on AFFICHE le dernier resultat, et on declenche la
        # mesure suivante EN TACHE DE FOND quand il a vieilli. Le conky ne
        # bloque jamais ; au pire la matrice a une minute de retard, ce qui est
        # sans importance pour une topologie qui ne bouge pas.
        matrice
    fi ;;
lte)
    # ── LA 4G DU BANC : Open5GS (le coeur EPC) + srsRAN sur ZeroMQ ──────────
    # [2026-09-10] Le Conky ne disait RIEN de la 4G, alors que le banc la porte
    # depuis tools/osmo-lte.sh : les huit demons Open5GS (+ mongod, sans qui le
    # HSS et le PCRF meurent a l init), l eNB et l UE srsRAN relies par ZeroMQ,
    # le S1 sur 36412 et ogstun pour le plan usager. Il fallait ouvrir un
    # terminal et taper « osmo-lte status » pour savoir si la 4G etait debout.
    #
    # Les sondes sont celles de tools/osmo-epc.sh et tools/osmo-lte.sh - LES
    # sources - mais SANS leur mesure de debit ZeroMQ : celle-la dort une
    # seconde entre deux lectures de compteur, et un execpi qui dort FIGE le
    # Conky entier (meme raison que le cache de la matrice SS7 plus haut).
    # Tout est natif ici : Open5GS n est pas conteneurise, donc pas de in_op.
    _o5gs=""
    for p in "${OPEN5GS_PREFIX:-}" /opt/LTE/open5gs/install /root/open5gs/install; do
        [ -n "$p" ] && [ -x "$p/bin/open5gs-mmed" ] && { _o5gs="$p"; break; }
    done
    _srs=""
    for b in /usr/local/bin/srsenb "${OSMO_SRSRAN_BUILD:-/opt/LTE/srsRAN_4G/build}/srsenb/src/srsenb"; do
        [ -x "$b" ] && { _srs="$b"; break; }
    done
    have srsenb && _srs="${_srs:-srsenb}"
    if [ -z "$_o5gs" ] && [ -z "$_srs" ]; then
        echo "${C1}4G non installee${C} - ${C2}osmo-lte-install --build${C}"; exit 0
    fi
    # ── UNE 4G A L ARRET NE MERITE PAS TREIZE PASTILLES ROUGES ──────────────
    # Meme regle que banc_debout pour la 2G : quand rien ne tourne, une ligne
    # qui dit quoi taper vaut mieux qu un mur d alarmes.
    _debout=0
    for d in open5gs-mmed srsenb srsue; do
        pgrep -x "$d" >/dev/null 2>&1 && _debout=1
    done
    if [ "$_debout" = 0 ]; then
        # La console des abonnes, elle, vit sa vie : elle ne lit que MongoDB et
        # sert justement a provisionner coeur eteint. Si elle ecoute, on le dit.
        _w=""
        ss -tanH state listening "( sport = :9999 )" 2>/dev/null | grep -q . \
            && _w="  ${AR}${OK} console ${C1}:9999${C}"
        echo "${C1}4G a l arret${C} - ${C2}osmo-lte start${C}${_w}"; exit 0
    fi
    # Le coeur : les huit demons dans l ordre de demarrage d osmo-epc.sh,
    # mongod - le HSS et le PCRF y lisent les abonnes et MEURENT sans lui (le
    # MME repond alors 3002 sur S6a et rejette tout attach) - et la console des
    # abonnes (open5gs-webui.service, :9999), qui lit la meme base.
    i=0
    for e in MME:open5gs-mmed HSS:open5gs-hssd SGWC:open5gs-sgwcd SGWU:open5gs-sgwud \
             SMF:open5gs-smfd UPF:open5gs-upfd PCRF:open5gs-pcrfd NRF:open5gs-nrfd \
             MONGO:mongod WEBUI:@9999; do
        IFS=: read -r name proc <<< "$e"
```

```
        # « @port » : une console, pas un demon - le WebUI d Open5GS est un
        # node, son nom de processus ne dit rien. On regarde ce qu il ECOUTE.
        case "$proc" in
            @*) ss -tanH state listening "( sport = :${proc#@} )" 2>/dev/null | grep -q . \
                    && s="$OK" || s="$KO" ;;
            *)  pgrep -x "$proc" >/dev/null 2>&1 && s="$OK" || s="$KO" ;;
        esac
        # %-6s et pas %-5s : « MONGO » fait cinq caracteres et collait a la
        # pastille suivante.
        printf '%s %-6s' "$s" "$name"
        i=$((i+1)); [ $((i % 5)) -eq 0 ] && echo
    done
    [ $((i % 5)) -eq 0 ] || echo
    # La radio et les liens : eNB, UE, S1 (SCTP 36412), ZeroMQ, ogstun.
    s=""
    pgrep -x srsenb >/dev/null 2>&1 && s="$s$OK eNB   " || s="$s$KO eNB   "
    pgrep -x srsue  >/dev/null 2>&1 && s="$s$OK UE    " || s="$s$KO UE    "
    ss -San 2>/dev/null | grep -q "ESTAB.*:36412" && s="$s$OK S1    " || s="$s$KO S1    "
    _zp="${OSMO_ZMQ_PORT:-2000}"
    ss -tanH state established "( sport = :$_zp or dport = :$_zp )" 2>/dev/null | grep -q . \
        && s="$s$OK ZMQ   " || s="$s$KO ZMQ   "
    ip -br link show ogstun >/dev/null 2>&1 && s="$s$OK TUN" || s="$s$KO TUN"
    echo "$s"
    # La cellule, telle que la declare enb.conf - celui qui a servi si on peut
    # le lire (root), celui du depot sinon : c est la source de la copie.
    _cfg=""
    for f in "${OSMO_SRSRAN_DIR:-/root/.config/srsran}/enb.conf" \
             "${DIR:-/opt/GSM/osmo-operator}/configs/srsran/enb.conf" \
             /opt/GSM/osmo-operator/configs/srsran/enb.conf; do
        [ -r "$f" ] && { _cfg="$f"; break; }
    done
    mcc=""; mnc=""; earfcn=""; prb=""
    if [ -n "$_cfg" ]; then
        eval "$(awk -F= '
            /^[[:space:]]*mcc[[:space:]]*=/       {gsub(/[[:space:]]/,"",$2); print "mcc="$2}
            /^[[:space:]]*mnc[[:space:]]*=/       {gsub(/[[:space:]]/,"",$2); print "mnc="$2}
            /^[[:space:]]*dl_earfcn[[:space:]]*=/ {gsub(/[[:space:]]/,"",$2); print "earfcn="$2}
            /^[[:space:]]*n_prb[[:space:]]*=/     {gsub(/[[:space:]]/,"",$2); print "prb="$2}' "$_cfg")"
    fi
    # Le nombre de PRB EST la largeur de canal : on affiche celle qu on lit.
    case "$prb" in 6) bw=1.4;; 15) bw=3;; 25) bw=5;; 50) bw=10;; 75) bw=15;; 100) bw=20;; *) bw="";; esac
    # L adresse de l UE vit dans un netns : « ip -n » demande le privilege.
    # Sous la session (uid non root) on ne l a pas - on ne dit rien plutot
    # que de laisser croire que l UE n est pas attache.
    ue="$(ip -n "${OSMO_LTE_NETNS:-ue1}" -4 -o addr show 2>/dev/null | awk '/tun_srsue/{print $4; exit}')"
    echo "PLMN ${C1}${mcc:--}-${mnc:--}${C}  EARFCN ${C1}${earfcn:--}${C}${bw:+  ${C1}${bw}${C} MHz}${ue:+ ${AR}UE
     ↪  ${C1}${ue}${C}}"
    ;;
services)
    # Banc a terre : une seule ligne, celle qui sert - les unites en echec.
    # Le reste (web, pulse, tmux, docker) ne dit rien d autre que « rien ne
    # tourne », ce que l absence des sections ci-dessus a deja dit.
    if ! banc_debout; then
        f="$(systemctl --failed --no-legend 2>/dev/null | wc -l)"
        [ "$f" -gt 0 ] && echo "${WARN} ${f} service(s) systemd en echec" \
                       || echo "${C1}banc a l arret${C} - ${C2}./start-native.sh${C}"
        exit 0
    fi
    s=""
    port_open 8080 || port_open 8443 && s="$s$OK web " || s="$s$KO web "
    pgrep -x pulseaudio >/dev/null 2>&1 && s="$s$OK pulse " || s="$s$KO pulse "
    # ── PLUS DE PASTILLES tmux NI docker ──────────────────────────────
    # [2026-09-04] Elles etaient rouges sur un banc en parfait etat, et pour la
    # meme raison toutes les deux : le Conky tourne sous le compte de la
    # SESSION, et ces deux sondes repondaient a la question « qu est-ce que MOI
    # je vois », pas « qu est-ce qui tourne sur cette machine ».
```

```
    #   tmux   : le banc ouvre ses sessions sous ROOT, sur /tmp/tmux-0/default.
    #            `tmux list-sessions` sous l utilisateur regarde
    #            /tmp/tmux-1001/default, qui n existe pas :
    #              error connecting to /tmp/tmux-1001/default
    #            Zero session, pastille rouge - avec `calypso` et `gapk`
    #            bien ouvertes a cote.
    #   docker : « permission denied ... /var/run/docker.sock » tant que le
    #            compte n est pas dans le groupe docker, donc zero conteneur -
    #            avec trois qui tournent.
    # On pourrait corriger chaque sonde ; on les RETIRE. Ni l une ni l autre ne
    # dit quoi que ce soit d utile sur l etat du banc : les sections Coeur GSM
    # et Radio disent deja si la pile est debout, et la ligne des operateurs
    # (sous-commande `ops`) dit quels conteneurs tournent - celle-la sait
    # distinguer « arrete » de « invisible ». Deux voyants de moins qui
    # clignotent pour rien.
    have nvidia-smi && s="$s$OK nvidia "
    echo "$s"
    # ── ET ON NOMME CE QUI ECHOUE ──────────────────────────────────────
    # « 1 service(s) systemd en echec » envoyait chercher lequel a la main.
    # C etait systemd-networkd-wait-online.service : masque par l image (il
    # coutait deux minutes de boot) mais reste `failed` de son echec d avant le
    # masquage - masquer n efface pas l etat, il faut `systemctl reset-failed`.
    # Un nom, et le diagnostic prend trois secondes au lieu de dix minutes.
    # `systemctl --failed --no-legend` commence par une PUCE (●), pas par le nom :
    # un `awk '{print $1}'` rendait « ● ». On prend le champ qui ressemble a une
    # unite, quelle que soit sa position.
    f="$(systemctl --failed --no-legend 2>/dev/null \
         | awk '{for(i=1;i<=NF;i++) if($i ~ /\.(service|socket|target|timer|mount|path)$/) {print $i; break}}' \
         | sed 's/\.service$//' | paste -sd' ' -)"
    [ -n "$f" ] && echo "${WARN} en echec : ${C1}${f}${C}" || echo "${OK} systemd sans echec"
    ;;
*) echo "usage: $0 role|vue|net|hub|core|radio|lte|subs|services" >&2; exit 2 ;;
esac
```

3.387 /opt/GSM/osmo-operator/tools/daram_sweep.sh

5457 octets, 103 lignes → 103 lignes

```
#!/bin/bash
# tools/daram_sweep.sh - balaye CALYPSO_BSP_DARAM_ADDR en full mode et compare
# l'activite du correlateur FB-det (fb0_att / snr / toa / pm).
# But : trancher empiriquement quelle zone DARAM debloque le FB-det.
#
# Robuste : run.sh est lance NO_ATTACH EN AVANT-PLAN (il rend la main seul
# apres avoir lance qemu, qui survit orphelin). On NE backgroude PAS run.sh
# avec & (ca cassait le pipe du log → qemu mourait). On verifie que qemu est
# vivant avant de mesurer ; sinon on le signale.
#
# Usage (conteneur) : /opt/GSM/daram_sweep.sh
# Env : ADDRS (defaut "0x0080 0x2a00 0x0060"), SETTLE (attente post-boot, 50s)
set -u
RUN=/opt/GSM/qosmo-grgsm/run.sh
QLOG=/root/qemu.log
ADDRS="${ADDRS:-0x0080 0x2a00 0x0060}"
SETTLE="${SETTLE:-50}"
# Profil d'emulation du balayage. "full" a disparu de environment/modes.env :
# les profils sont empty | shunt_legit | shunt_legit_no_inject | native |
# native_twl | native_twl_host_demod | native_helped.
MODE="${MODE:-${CALYPSO_MODE:-native}}"
declare -A R_FB R_SB R_NZ R_BEST R_ALIVE R_STATES R_TOT

kill_all() {
    tmux kill-session -t calypso 2>/dev/null
    for p in $(pgrep -f qemu-system-arm); do kill -9 "$p" 2>/dev/null; done
```

```
    pkill -9 -f osmo-bts-trx 2>/dev/null; pkill -9 -f osmo-trx-ipc 2>/dev/null
    pkill -9 -f "mobile" 2>/dev/null; pkill -9 -f osmocon 2>/dev/null
    sleep 4
}

for a in $ADDRS; do
    echo "############ DARAM_ADDR=$a (profil $MODE) ############ $(date +%H:%M:%S)"
    kill_all
    if pgrep -f qemu-system-arm >/dev/null; then
        echo "  !! un qemu survit a kill_all : la supervision le relance." >&2
        echo "     Arretez la pile (CALYPSO_NO_RESTART=1 ou ./run.sh --stop) avant le balayage." >&2
        exit 3
    fi
    : > "$QLOG"
    # run.sh NO_ATTACH rend la main seul apres le launch (~30-60s). timeout = filet.
    echo "  launch run.sh (NO_ATTACH, avant-plan, le boot prend ~30-60s)..."
    timeout 150 env CALYPSO_NO_ATTACH=1 CALYPSO_MODE="$MODE" CALYPSO_ICOUNT=off \
      CALYPSO_NO_RESTART=1 CALYPSO_BSP_DARAM_ADDR="$a" "$RUN" \
      </dev/null >"/tmp/sweep_$a.out" 2>&1
    rc_run=$?
    # Le profil a-t-il seulement ete applique ? Sans ce controle, un mode
    # inconnu faisait avorter run.sh a son deuxieme module, l'ancien qemu
    # (relance par la supervision) continuait d'ecrire dans le MEME journal,
    # et le balayage mesurait cette instance-la : les trois adresses rendaient
    # alors la meme "premiere detection", au caractere pres.
    if grep -qa "CALYPSO_MODE inconnu\|sequence aborted" "/tmp/sweep_$a.out"; then
        echo "  !! run.sh a avorte : profil '''$MODE''' refuse par environment/modes.env" >&2
        sed -n '''/valeurs :/,+2p''' "/tmp/sweep_$a.out" | sed "s/^/     /" >&2
        echo "     Relancez avec MODE=<profil valide> $0" >&2
        exit 2
    fi
    echo "  run.sh rendu (code=$?). qemu doit tourner orphelin. settle ${SETTLE}s..."
    alive=no
    for i in $(seq 1 $((SETTLE/5))); do
        sleep 5
        if pgrep -f qemu-system-arm >/dev/null; then alive=yes; else alive=no; fi
        # stop tot si on a deja du FB data non trivial
        grep -qa "real DSP path" "$QLOG" 2>/dev/null && [ "$i" -ge 4 ] && break
    done

    fb=$(grep -a "real DSP path" "$QLOG" 2>/dev/null | grep -oE "fb0_att=[0-9]+" | grep -oE "[0-9]+" | sort -n |
     ↪  tail -1)
    sb=$(grep -a "real DSP path" "$QLOG" 2>/dev/null | grep -oE "sb_att=[0-9]+"  | grep -oE "[0-9]+" | sort -n |
     ↪  tail -1)
    # Le format des lignes FBSB a change : le correlateur s'ecrit desormais
    #   data[](det=0 toa=0 pm=0 ang=0 snr=0x0000) api[](...)
    # et non plus "last(snr=... toa=... ang=... pm=...)". Les deux motifs sont
    # acceptes : sans cela le balayage annoncait "non0=0 / 1er_corr=none" pour
    # TOUTES les adresses, y compris celles ou le correlateur voit tres bien
    # l'I/Q - un faux negatif qui condamnait la bonne zone DARAM.
    NUL='(det=0 toa=0 pm=0 ang=0 snr=0x0000)|snr=0 toa=0 ang=0 pm=0'
    nz=$(grep -a "real DSP path" "$QLOG" 2>/dev/null | grep -avcE "$NUL")
    best=$(grep -a "real DSP path" "$QLOG" 2>/dev/null | grep -avE "$NUL" \
             | grep -oE "(data\[\]\(det=[0-9-]+ toa=[0-9-]+ pm=[0-9-]+ ang=[0-9-]+
              ↪  snr=0x[0-9a-fA-F]+\))|(last\(snr=[0-9-]+ toa=[0-9-]+ ang=[0-9-]+ pm=[0-9a-fx]+\))" \
             | head -1)
    # Combien de passages en tout : le taux de detection compte autant que le
    # premier succes (19 % ne suffit pas a verrouiller la synchro).
    tot=$(grep -ac "real DSP path" "$QLOG" 2>/dev/null)
    states=$(grep -aoE 'state=[A-Z0-9_]+' "$QLOG" 2>/dev/null | sort | uniq -c | tr '\n' ' ')
    qlines=$(wc -l < "$QLOG" 2>/dev/null)
    R_FB[$a]="${fb:-0}"; R_SB[$a]="${sb:-0}"; R_NZ[$a]="${nz:-0}"
    R_BEST[$a]="${best:-none}"; R_ALIVE[$a]="$alive"; R_STATES[$a]="${states:-<aucun>}"
    R_TOT[$a]="${tot:-0}"
   echo "  -> qemu_vivant=$alive  qemu.log=${qlines}l  fb0_att=${fb:-0}  sb_att=${sb:-0}  non0=${nz:-0}/${tot:-0}"
    echo "  -> etats: ${states:-<aucun>}"
```

```
    echo "  -> 1er correlateur non-nul: ${best:-none}"
done

kill_all
echo
echo "================= COMPARAISON ================="
printf "%-9s %-7s %-9s %-8s %-12s %s\n" "ADDR" "alive" "fb0_att" "sb_att" "non0/total" "1er_corr"
for a in $ADDRS; do
    printf "%-9s %-7s %-9s %-8s %-12s %s\n" \
      "$a" "${R_ALIVE[$a]}" "${R_FB[$a]}" "${R_SB[$a]}" \
      "${R_NZ[$a]}/${R_TOT[$a]}" "${R_BEST[$a]}"
done
echo
echo "Gagnant = fb0_att>0 et/ou non0>0 (correlateur voit l'I/Q). alive=no => boot KO."
```

3.388 /opt/GSM/osmo-operator/tools/deaccent.pl

1212 octets, 29 lignes → 29 lignes

```
#!/usr/bin/perl
# deaccent.pl [fichiers...] — translittere accents FR + ponctuation typo vers
# ASCII, en place. Laisse intacts box-drawing, fleches et symboles (► ✓ ⚠ •).
# Sans argument : filtre stdin -> stdout.
use strict; use warnings; use utf8;
sub xlate {
  my $l = shift;
  $l =~ tr/àâäãáéèêëíìîïóòôöõúùûüçñýÿ/aaaaaeeeeiiiioooooouuuucnyy/;
  $l =~ tr/ÀÂÄÃÁÉÈÊËÍÌÎÏÓÒÔÖÕÚÙÛÜÇÑÝ/AAAAAEEEEIIIIOOOOOUUUUCNY/;
  $l =~ s/œ/oe/g; $l =~ s/Œ/OE/g; $l =~ s/æ/ae/g; $l =~ s/Æ/AE/g;
  $l =~ s/\x{00A0}/ /g; $l =~ s/\x{202F}/ /g;
  $l =~ s/[\x{2018}\x{2019}]/'/g; $l =~ s/[\x{201C}\x{201D}]/"/g;
  $l =~ s/\x{2026}/.../g; $l =~ s/[\x{2013}\x{2014}]/-/g;
  $l =~ s/\x{00AB}\s*/"/g; $l =~ s/\s*\x{00BB}/"/g;
  return $l;
}
if (!@ARGV) {
  binmode(STDIN,':utf8'); binmode(STDOUT,':utf8');
  print xlate($_) while <STDIN>; exit 0;
}
for my $f (@ARGV) {
  open(my $in,'<:utf8',$f) or do { warn "skip $f: $!\n"; next };
  local $/; my $c = <$in>; close $in;
  my $o = join('', map { xlate($_) } split(/(?<=\n)/, $c));
  next if $o eq $c;
  open(my $out,'>:utf8',$f) or do { warn "write $f: $!\n"; next };
  print $out $o; close $out;
  print "  fixed $f\n";
}
```

3.389 /opt/GSM/osmo-operator/tools/dockerfile-to-install-md.sh

13325 octets, 283 lignes → 283 lignes

```
#!/usr/bin/env bash
# dockerfile-to-install-md.sh - le Dockerfile, etape par etape, en Markdown « installation depuis zero ».
#
# [2026-10-20] Le Dockerfile est la seule description complete et testee de l'installation
# (paquets apt, pile Osmocom compilee, osmocom-bb + firmware, qosmo, gr-gsm, node, LTE...).
# Plutot que de maintenir a la main un second document qui derive, on le genere :
#   chaque FROM devient un chapitre, chaque RUN un bloc bash a coller tel quel dans un shell
#   root, chaque COPY un cp depuis le depot, les commentaires du Dockerfile deviennent la prose.
#
```

```
#   tools/dockerfile-to-install-md.sh                    # Dockerfile -> doc/install-from-scratch.md
#   tools/dockerfile-to-install-md.sh Dockerfile.run -o doc/x.md
#   tools/dockerfile-to-install-md.sh --annexe doc/c54x_exe-depuis-ubuntu-vierge.md
#
# Premiere version volontairement simple (python3 embarque, aucune dependance) ; a ameliorer.
set -euo pipefail
HERE="$(cd "$(dirname "${BASH_SOURCE[0]}")/.." && pwd)"
DOCKERFILE="$HERE/Dockerfile"
OUT="$HERE/doc/install-from-scratch.md"
ANNEXES=()
LANG_DOC=fr
NOTES=""
while [ $# -gt 0 ]; do
    case "$1" in
        -o|--out)    OUT="$2"; shift ;;
        --annexe)    ANNEXES+=("$2"); shift ;;
        # --lang en : prose anglaise ; les commentaires (francais) du Dockerfile sont remplaces par les
        # notes par etape de --notes <fichier> (sections « ## <nom d'etape> »), les blocs de commandes restent.
        --lang)      LANG_DOC="$2"; shift ;;
        --notes)     NOTES="$2"; shift ;;
        -h|--help)   sed -n '2,15p' "$0"; exit 0 ;;
        *)           DOCKERFILE="$1" ;;
    esac
    shift
done
[ -r "$DOCKERFILE" ] || { echo "Dockerfile illisible : $DOCKERFILE" >&2; exit 1; }
mkdir -p "$(dirname "$OUT")"
python3 - "$DOCKERFILE" "$OUT" "$HERE" "$LANG_DOC" "$NOTES" "${ANNEXES[@]}" <<'EOF'
import re, sys, datetime, os
src, out, root, lang, notes_file, *annexes = sys.argv[1:]
EN = (lang == "en")
notes = {}
if notes_file and os.path.isfile(notes_file):
    cur = None
    for l in open(notes_file, encoding="utf-8").read().split("\n"):
        m = re.match(r"^## (\S+)", l)
        if m:
            cur = m.group(1); notes[cur] = []
        elif cur is not None:
            notes[cur].append(l)
    notes = {k: "\n".join(v).strip() for k, v in notes.items()}
lines = open(src, encoding="utf-8", errors="replace").read().split("\n")

# 1. recoller les continuations (\ en fin de ligne), garder les commentaires qui precedent
items = []          # (comments:list[str], instr:str, args:str)
pending = []
i = 0
while i < len(lines):
    l = lines[i]
    if not l.strip():
        if pending and items and not items[-1][1]:
            pass
        i += 1
        continue
    if l.lstrip().startswith("#"):
        # on garde l'indentation apres le « # » : les lignes indentees (schemas, listes
        # « RAND=0 -> ... ») sont rendues verbatim, dans un bloc
        pending.append(l.lstrip()[1:].rstrip())
        i += 1
        continue
    buf = [l]
    # continuation : tant que la derniere ligne NON commentaire finit par « \ » ; Docker admet
    # des lignes « # ... » au milieu d'un RUN continue, elles ne portent pas de « \ »
    def derniere_utile():
        for b in reversed(buf):
            if not b.lstrip().startswith("#"):
```

```
                return b
        return ""
    while derniere_utile().rstrip().endswith("\\") and i + 1 < len(lines):
        i += 1
        buf.append(lines[i])
    i += 1
    text = "\n".join(buf)
    m = re.match(r"\s*([A-Z]+)\s*(.*)", text, re.S)
    if not m:
        continue
    items.append((pending, m.group(1), m.group(2)))
    pending = []

def prose(comments):
    """Les commentaires du Dockerfile, en paragraphes. Les titres « N. ... » et « -- ... -- » deviennent des
↪   sous-titres."""
    outl = []
    if EN:
        return outl
    para = []
    def flush():
        if para:
            outl.append(" ".join(para)); outl.append("")
            para.clear()
    verb = []
    def flush_verb():
        if verb:
            flush(); outl.append("```text"); outl.extend(verb); outl.append("```"); outl.append("")
            verb.clear()
    for c in comments:
        if c.startswith("  ") and c.strip():        # ligne indentee : verbatim
            verb.append(c[1:] if c.startswith(" ") else c)
            continue
        flush_verb()
        c = c.strip()
        if c.startswith("syntax="):
            continue
        s = re.sub(r"^[=\-]{3,}\s*", "", c); s = re.sub(r"\s*[=\-]{3,}$", "", s)
        if not s:
            flush(); continue
        if re.match(r"^(\d+\.\s+|[A-ZÉ][^a-z]{6,}$)", s) or (c.startswith("==") or c.startswith("---") or
↪   c.startswith("==")):
            flush(); outl.append("#### " + s.strip(" :")); outl.append("")
            continue
        para.append(s)
    flush_verb(); flush()
    return outl

def strip_mounts(cmd):
    cmd = re.sub(r"--mount=\S+(\s*\\\n\s*)?", "", cmd)
    return cmd

def run_block(args):
    cmd = strip_mounts(args).strip()
    # commentaires internes aux RUN : ils sont deja du bash (#), on les garde
    return ["```bash", cmd, "```", ""]

doc = []
if EN:
    doc.append("# Installing from scratch (generated from the Dockerfile)")
    doc.append("")
    doc.append("> Generated by `tools/dockerfile-to-install-md.sh --lang en` on %s from `%s`. Do not edit by hand:
↪   fix the Dockerfile, the script or `%s`, then regenerate. The French version (`doc/install-from-scratch.md`)
↪   carries the full rationale written in the Dockerfile comments; this one keeps the commands and a short English
↪   note per stage."
```

```
              % (datetime.date.today().isoformat(), os.path.relpath(src, root), os.path.relpath(notes_file, root)
 ↪  if notes_file else "the notes file"))
    doc.append("")
    doc.append("""## How to read this page

The osmo-operator Dockerfile is the complete, tested description of the installation. This page replays it
step by step on a bare machine (Ubuntu 24.04, as root), without Docker:

- every `FROM ... AS <name>` of the Dockerfile is a chapter; on a single machine the stages simply follow
  each other and the `COPY --from=<stage>` instructions are not needed (everything is already in place);
- every `RUN` is a bash block to paste as is into a root shell, in order;
- every `ARG`/`ENV` becomes an `export` to set before the blocks that follow (in Docker an `ARG` does not
  survive a `FROM`; here it just stays in the shell);
- every `COPY` becomes a `cp` from a checkout of the osmo-operator repository (`$REPO`, `/opt/GSM/osmo-operator`
 ↪  by default).

The `.deb` package cache (`osmo-deb install <name> <version>`) is an optimisation of the Docker build:
without the cache the command fails and the `if ! osmo-deb install ...; then ... fi` branch builds from
source, which is exactly what a bare machine needs. `osmo-deb pack` then wraps the result (`make install`);
without `/var/cache/osmo-debs` it just installs.

```bash
export REPO=/opt/GSM/osmo-operator      # checkout of the repository (git clone
 ↪  https://github.com/bbaranoff/osmo-operator)
export ROOT=/opt/GSM
mkdir -p "$ROOT"
```
""")
fr = []
fr.append("# Installation depuis zero (generee depuis le Dockerfile)")
fr.append("")
fr.append("> Fichier genere par `tools/dockerfile-to-install-md.sh` le %s depuis `%s`. Ne pas editer a la main :
 ↪ corriger le Dockerfile ou le script, puis regenerer."
          % (datetime.date.today().isoformat(), os.path.relpath(src, root)))
fr.append("")
fr.append("""## Comment lire cette page

Le Dockerfile d'osmo-operator est la description complete et testee de l'installation. Cette page le rejoue
etape par etape sur une machine nue (Ubuntu 24.04, en root), sans Docker :

- chaque `FROM ... AS <nom>` du Dockerfile est un chapitre ; sur une seule machine les etapes se suivent
  simplement, les `COPY --from=<etape>` ne sont plus necessaires (tout est deja en place) ;
- chaque `RUN` est un bloc bash a coller tel quel dans un shell root, dans l'ordre ;
- chaque `ARG`/`ENV` devient un `export` a poser avant les blocs qui suivent (les `ARG` ne traversent pas
  un `FROM` dans Docker, ici on les garde dans le shell) ;
- chaque `COPY` devient un `cp` depuis une copie du depot osmo-operator (`$REPO`, par defaut
 ↪ `/opt/GSM/osmo-operator`).

Le cache de paquets `.deb` (`osmo-deb install <nom> <version>`) est une optimisation du build Docker :
sans cache, la commande echoue et la branche `if ! osmo-deb install ...; then ... fi` compile depuis les
sources, ce qui est exactement ce qu'on veut sur une machine nue. `osmo-deb pack` emballe ensuite le
resultat (`make install`) ; sans `/var/cache/osmo-debs` il se contente d'installer.

```bash
export REPO=/opt/GSM/osmo-operator      # copie du depot (git clone https://github.com/bbaranoff/osmo-operator)
export ROOT=/opt/GSM
mkdir -p "$ROOT"
```
""")

if not EN:
    doc.extend(fr)

# sommaire
stages = [(k, a) for (c, k, a) in items if k == "FROM"]
doc.append("## Table of contents" if EN else "## Sommaire")
```

```
doc.append("")
n = 0
for c, k, a in items:
    if k == "FROM":
        n += 1
        m = re.match(r"(\S+)(?:\s+AS\s+(\S+))?", a, re.I)
        base, name = m.group(1), (m.group(2) or m.group(1))
        if EN:
            doc.append("%d. [%s](#stage-%d--%s) (from `%s`)" % (n, name, n, name.lower().replace("_", "-"), base))
        else:
            doc.append("%d. [%s](#etape-%d--%s) (a partir de `%s`)" % (n, name, n, name.lower().replace("_", "-"),
↪  base))
doc.append("")

n = 0
for comments, k, a in items:
    if k == "FROM":
        n += 1
        m = re.match(r"(\S+)(?:\s+AS\s+(\S+))?", a, re.I)
        base, name = m.group(1), (m.group(2) or m.group(1))
        doc.append("")
        doc.append(("## Stage %d : %s" if EN else "## Etape %d : %s") % (n, name))
        doc.append("")
        doc.extend(prose(comments))
        if EN and notes.get(name):
            doc.append(notes[name]); doc.append("")
        if base == "scratch":
            doc.append("_Docker stage without content of its own (`scratch` image): nothing to do on a machine._"
↪  if EN else "_Etape Docker sans contenu propre (image `scratch`) : rien a faire sur une machine._")
            doc.append("")
        else:
            doc.append(("_In Docker: `FROM %s`. On a machine, carry on in the same system._" if EN else "_Dans
↪  Docker : `FROM %s`. Sur une machine, on continue dans le meme systeme._") % base)
            doc.append("")
        continue
    doc.extend(prose(comments))
    if k in ("ARG", "ENV"):
        body = a.replace("\\\n", "\n").strip()
        exports = []
        for kv in re.findall(r"([A-Za-z_][A-Za-z0-9_]*)=((?:\"[^\"]*\")|(?:'[^']*')|\S*)", body):
            exports.append("export %s=%s" % (kv[0], kv[1] if kv[1] != "" else '""'))
        if not exports and body:
            exports.append("export " + body)
        doc.append("```bash"); doc.extend(exports); doc.append("```"); doc.append("")
    elif k == "RUN":
        doc.extend(run_block(a))
    elif k == "COPY" or k == "ADD":
        parts = a.split()
        opts = [p for p in parts if p.startswith("--")]
        files = [p for p in parts if not p.startswith("--")]
        frm = [o for o in opts if o.startswith("--from=")]
        if frm:
            doc.append(("_Copy between Docker stages (`%s`): on a single machine %s is already in place._" if EN
↪  else "_Copie entre etapes Docker (`%s`) : sur une seule machine, %s est deja en place._") % (a.replace("\n", "
↪  "), ", ".join("`%s`" % f for f in files[:-1])))
            doc.append("")
        elif len(files) >= 2:
            dst = files[-1]
            doc.append("```bash")
            for s in files[:-1]:
                if dst.endswith("/"):
                    doc.append('mkdir -p %s && cp -r "$REPO"/%s %s' % (dst, s, dst))
                else:
                    doc.append('mkdir -p "$(dirname %s)" && cp -r "$REPO"/%s %s' % (dst, s, dst))
            doc.append("```"); doc.append("")
    elif k == "WORKDIR":
```

```
        doc.append("```bash"); doc.append("cd %s" % a.strip()); doc.append("```"); doc.append("")
    elif k in ("STOPSIGNAL", "ENTRYPOINT", "CMD", "EXPOSE", "LABEL", "HEALTHCHECK", "USER", "VOLUME", "SHELL"):
        doc.append(("_Docker metadata, no equivalent on a machine: `%s %s`._" if EN else "_Metadonnee Docker, sans
 ↪  equivalent sur une machine : `%s %s`._") % (k, a.replace("\n", " ").strip()))
        doc.append("")
    else:
        doc.append(("_Unhandled instruction: `%s %s`._" if EN else "_Instruction non traitee : `%s %s`._") % (k,
 ↪  a[:80]))
        doc.append("")

for an in annexes:
    if not os.path.isfile(an):
        continue
    doc.append("")
    doc.append("---")
    doc.append("")
    doc.append(("# Appendix: %s" if EN else "# Annexe : %s") % os.path.basename(an))
    doc.append("")
    doc.append(("_Contents of `%s`, included as is._" if EN else "_Contenu de `%s`, repris tel quel._") %
 ↪  os.path.relpath(an, root))
    doc.append("")
    txt = open(an, encoding="utf-8", errors="replace").read()
    # decaler les titres d'un niveau
    txt = re.sub(r"^(#{1,5}) ", lambda m: "#" + m.group(1) + " ", txt, flags=re.M)
    doc.append(txt)

open(out, "w", encoding="utf-8").write("\n".join(doc).rstrip() + "\n")
print("%s : %d lignes, %d etapes FROM, %d RUN" % (out, len(doc), len(stages), sum(1 for c,k,a in items if
 ↪  k=="RUN")))
EOF
```

3.390 /opt/GSM/osmo-operator/tools/fft.sh

9402 octets, 199 lignes → 199 lignes

```
#!/usr/bin/env bash
# tools/fft.sh - FFT "jolie" du flux I/Q Calypso, lue DIRECTEMENT dans le docker.
#
# Lit /tmp/record.cfile (le ring record CONTINU, ecrit par record_drain.py qui
# draine iq_record.fifo en O_RDWR - DECOUPLE de la boucle relay temps-reel) DANS
# le container, le rapatrie en fc32 (complex64), et affiche sur le X de l'HOTE :
# PSD moyennee (Welch maison) + waterfall defilant. Boucle (FuncAnimation).
#
# /!\ Le mode 'fifo' (tap LIVE de /tmp/iq_fft.fifo) a ete RETIRE : ouvrir ce fifo
#     en lecture force l'IPC device a l'ecrire et perturbe le relai temps-reel ->
#     le decode grgsm s'arrete (si-bridge 'fini') et le camping fige. record.cfile
#     est near-live et n'a AUCUN effet sur le pipeline.
#
# Usage :
#   ./tools/fft.sh                      # tail de record.cfile (defaut, non-perturbant)
#   FFT_SRC=sweep ./tools/fft.sh        # balaye tout le ring record
#   ./tools/fft.sh --save spectre.png   # une capture PNG (si pas de X)
#   CONTAINER=osmo-operator-1 CFILE=/tmp/record.cfile RATE=1083333 \
#   NSAMP=262144 REFRESH=1.0 ARFCN=514 ./tools/fft.sh
#
set -euo pipefail
export CONTAINER="${CONTAINER:-osmo-operator-1}"
export FFT_SRC="${FFT_SRC:-tail}"   # tail (defaut, NON-perturbant) | sweep - lisent record.cfile
# NB: le mode 'fifo' (tap LIVE de /tmp/iq_fft.fifo) a ete RETIRE : ouvrir ce
# fifo en lecture force l'IPC device a l'ecrire, ce qui perturbe la boucle relay
# temps-reel et COUPE le decode grgsm (si-bridge 'fini') -> camping fige. On lit
# desormais record.cfile, alimente par record_drain.py (decouple, always-on).
export CFILE="${CFILE:-/dev/shm/dsp_iq.cfile}"    # entree DSP shunt = MS (le mobile qui repond a la BTS ; cfile
 ↪  fc32 en /dev/shm RAM). Fichier qui GRANDIT -> tail-c lit le frais. (override CFILE=/dev/shm/record.cfile = BTS)
```

```
export RATE="${RATE:-1083333}"      # 4 SPS natif = 26e6/24
export NSAMP="${NSAMP:-32768}"      # complex samples par fenetre (256 KB) - court = ca gigote
export REFRESH="${REFRESH:-0.25}"   # periode de rafraichissement (s) - court = fluide
export ARFCN="${ARFCN:-514}"        # juste pour le titre (DCS1800)
export DISPLAY="${DISPLAY:-:0}"
export FFT_SAVE=""
[ "${1:-}" = "--save" ] && export FFT_SAVE="${2:?--save needs a path}"

# garde-fous
command -v docker >/dev/null || { echo "docker introuvable"; exit 1; }
case "$FFT_SRC" in
  sweep|tail) ;;
  *) echo "[fft] FFT_SRC='$FFT_SRC' invalide. Le mode 'fifo' (tap live) a ete retire"
     echo "[fft] (il perturbe le relai temps-reel et coupe le decode grgsm). -> tail."
     export FFT_SRC=tail ;;
esac
docker exec "$CONTAINER" test -f "$CFILE" \
  || { echo "record absent $CONTAINER:$CFILE (lance ./run.sh d'abord ; record_drain.py alimente record.cfile)";
   ↪  exit 1; }

python3 - <<'PYEOF'
import os, sys, subprocess, numpy as np
import matplotlib
SAVE = os.environ.get("FFT_SAVE","")
if SAVE:
    matplotlib.use("Agg")
else:
    for bk in ("TkAgg","Qt5Agg","GTK3Agg"):
        try: matplotlib.use(bk); break
        except Exception: continue
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

CONT  = os.environ["CONTAINER"]; CFILE = os.environ["CFILE"]
RATE  = float(os.environ["RATE"]); NS = int(os.environ["NSAMP"])
REFRESH = float(os.environ["REFRESH"]); ARFCN = os.environ["ARFCN"]
NBYTES = NS*8                       # complex64 = 8 o/sample
NFFT  = 4096                        # resolution FFT (segment Welch)
WROWS = 120                         # lignes du waterfall

win = np.hanning(NFFT).astype(np.float32)
freqs = np.fft.fftshift(np.fft.fftfreq(NFFT, 1.0/RATE)) / 1e3   # kHz

SRC  = os.environ.get("FFT_SRC","tail")    # tail (defaut) | sweep - lisent record.cfile
if SRC not in ("sweep","tail"): SRC = "tail"   # 'fifo' retire (tap live nocif)
BS   = 4096; NBLK = max(1, NBYTES//BS)

# ---- modes fichier : balaye le ring (sweep) ou lit le tail de record.cfile
#      (alimente par record_drain.py, decouple de la boucle relay temps-reel).
def _filesize():
    try:
        return int(subprocess.run(["docker","exec",CONT,"stat","-c","%s",CFILE],
                                  capture_output=True,timeout=5).stdout or 0)
    except Exception: return NBYTES
def _ringsize():
    # taille du ring publiee par record_drain (modulo pour le wrap LIVE)
    try:
        v = int(subprocess.run(["docker","exec",CONT,"cat",CFILE+".ring"],
                               capture_output=True,timeout=5).stdout or 0)
        if v > 0: return v
    except Exception: pass
    return _filesize()
RING = _ringsize()
_off = [0]

def _dd(skip_blk, count_blk):
```

```
    return subprocess.run(["docker","exec",CONT,"dd","if="+CFILE,"bs="+str(BS),
        "skip="+str(skip_blk),"count="+str(count_blk),"status=none"],
        capture_output=True, timeout=8).stdout

def grab():
    """Rend NBYTES d'I/Q complex64 (le plus frais), depuis record.cfile.
    SRC=tail  : derniers octets (defaut, near-live, non-perturbant).
    SRC=sweep : balaye le ring record (offset qui avance)."""
    if SRC == "sweep":
        skip = _off[0] // BS
        try:
            raw = subprocess.run(["docker","exec",CONT,"dd","if="+CFILE,
                "bs="+str(BS),"skip="+str(skip),"count="+str(NBLK),"status=none"],
                capture_output=True, timeout=8).stdout
        except Exception:
            return None
        _off[0] += NBYTES
        if _off[0] + NBYTES > RING: _off[0] = 0
    else:
        # LIVE (tail) : lit la fenetre fraiche FINISSANT a l'offset d'ecriture
        # publie par record_drain (<cfile>.off). Corrige le freeze : avec le ring
        # seek(0) le frais est a `w` (milieu), PAS au tail. Aligne blocs + wrap.
        try:
            w = int(subprocess.run(["docker","exec",CONT,"cat",CFILE+".off"],
                                   capture_output=True, timeout=5).stdout or -1)
        except Exception:
            w = -1
        try:
            if w >= 0:
                wblk = w // BS; start_blk = wblk - NBLK
                if start_blk >= 0:
                    raw = _dd(start_blk, NBLK)
                else:                                  # fenetre a cheval sur le wrap
                    ring_blk = RING // BS; n_end = -start_blk
                    raw = _dd(ring_blk - n_end, n_end) + _dd(0, wblk)
            else:                                       # repli : pas d'offset publie
                raw = subprocess.run(["docker","exec",CONT,"tail","-c",str(NBYTES),CFILE],
                                     capture_output=True, timeout=8).stdout
        except Exception:
            return None
    n = (len(raw)//8)*8
    if n < NFFT*8: return None
    return np.frombuffer(raw[:n], dtype=np.complex64)

def welch_psd(iq):
    """PSD moyennee (segments NFFT, overlap 50%, fenetre Hann) en dB"""
    step = NFFT//2; nseg = max(1,(len(iq)-NFFT)//step + 1)
    acc = np.zeros(NFFT, dtype=np.float64)
    for i in range(nseg):
        seg = iq[i*step:i*step+NFFT]
        if len(seg) < NFFT: break
        sp = np.fft.fftshift(np.fft.fft(seg*win))
        acc += (sp.real**2 + sp.imag**2)
    psd = acc/max(1,nseg)
    return 10*np.log10(psd/ (NFFT*RATE) + 1e-20)

# ---- figure "jolie" : PSD (haut) + waterfall (bas) ALIGNEES verticalement.
#      GridSpec 2x2 : col 0 = les 2 plots (MEME largeur), col 1 = colorbar (seulement
#      sur la ligne waterfall) -> la colorbar ne retrecit plus la waterfall, donc x=0
#      tombe sur la MEME verticale dans la PSD et la waterfall. sharex lie les axes. ----
plt.style.use("dark_background")
fig = plt.figure(figsize=(11,8))
gs = fig.add_gridspec(2, 2, width_ratios=[40, 1], height_ratios=[1, 1.4],
                      hspace=0.18, wspace=0.015)
axp = fig.add_subplot(gs[0, 0])
axw = fig.add_subplot(gs[1, 0], sharex=axp)   # x partage -> aligne avec la PSD
```

```
cax = fig.add_subplot(gs[1, 1])              # colorbar dans sa colonne (n'affecte pas axw)
fig.suptitle(f"Calypso I/Q live - {CONT}:{CFILE}  |  ARFCN={ARFCN}  Fs={RATE/1e6:.4f} MHz",
             fontsize=11, color="#7fd")
(line,) = axp.plot(freqs, np.full(NFFT,-120), lw=0.5, color="#39f")
axp.set_xlim(freqs[0],freqs[-1]); axp.set_ylim(-130,-30)
axp.set_ylabel("PSD (dB)"); axp.grid(alpha=0.25)
axp.axvline(0, color="#f55", lw=0.7, alpha=0.7)
plt.setp(axp.get_xticklabels(), visible=False)   # l'axe x est porte par la waterfall en bas

wf = np.full((WROWS,NFFT), -120.0)
im = axw.imshow(wf, aspect="auto", origin="upper", cmap="turbo",
                extent=[freqs[0],freqs[-1],0,WROWS], vmin=-120, vmax=-50,
                interpolation="bilinear")
axw.set_ylabel("temps (frames)"); axw.set_xlabel("offset (kHz)")
axw.axvline(0, color="#fff", lw=0.7, alpha=0.5)  # meme 0 vertical que la PSD
fig.colorbar(im, cax=cax, label="dB")
fig.subplots_adjust(left=0.08, right=0.92, top=0.92, bottom=0.08)

def update(_):
    iq = grab()
    if iq is None: return line, im
    psd = welch_psd(iq)
    line.set_ydata(psd)
    lo,hi = np.percentile(psd,5), np.percentile(psd,99.5)
    axp.set_ylim(lo-5, hi+8)
    global wf
    wf = np.roll(wf,1,axis=0); wf[0,:] = psd
    im.set_data(wf); im.set_clim(lo, hi+5)
    return line, im

if SAVE:
    update(0)                        # une passe
    fig.savefig(SAVE, dpi=110, facecolor=fig.get_facecolor())
    print("ecrit", SAVE)
else:
    ani = FuncAnimation(fig, update, interval=int(REFRESH*1000),
                        blit=False, cache_frame_data=False)
    print("FFT live - ferme la fenetre pour quitter.")
    plt.show()
PYEOF
```

3.391 /opt/GSM/osmo-operator/tools/fft2.sh

686 octets, 14 lignes → 14 lignes

```
#!/usr/bin/env bash
# tools/fft2.sh - FFT du 2e cfile. Reutilise tools/fft.sh avec un autre CFILE.
#
# tools/fft.sh  = dsp_iq.cfile = MS  (entree DSP Calypso : le mobile qui repond a la BTS)
# tools/fft2.sh = record.cfile = BTS (ce que la BTS emet)
# -> MEME signal aux deux bouts de la chaine (BTS TX <-> MS RX) ; la difference
#    entre les deux spectres = ce que la chaine (shunt/gr-gsm) fait au signal.
#
# Usage : ./tools/fft2.sh            (defaut CFILE=/dev/shm/record.cfile)
#         CFILE=... ./tools/fft2.sh
HERE="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
export CFILE="${CFILE:-/dev/shm/record.cfile}"
export ARFCN="${ARFCN:-BTS-record}"
exec bash "$HERE/fft.sh" "$@"
```

3.392 /opt/GSM/osmo-operator/tools/fft_global.sh

5678 octets, 146 lignes → 146 lignes

```
#!/usr/bin/env bash
# tools/fft_global.sh - FFT "jolie" d'un cfile LOCAL passe en argument (fichier statique).
#
# Derive de tools/fft.sh (osmo-operator) mais SANS docker / ring / tail-live : lit
# directement un cfile sur l'hote, le balaye en boucle, et affiche
# PSD moyennee (Welch maison) + waterfall defilant (FuncAnimation).
#
# Usage :
#   ./tools/fft_global.sh my.cfile
#   ./tools/fft_global.sh my.cfile --save spectre.png
#   RATE=1083333 NSAMP=32768 REFRESH=0.25 FORMAT=cf32 ARFCN=514 ./tools/fft_global.sh my.cfile
#
# FORMAT : cf32 (complex64, defaut, ce qu'ecrit le shunt) | cs8 | cs16
#          (record.cfile d'osmo-trx est souvent cs8 -> FORMAT=cs8)
set -euo pipefail

CFILE="${1:?usage: tools/fft_global.sh <fichier.cfile> [--save out.png]}"
[ -f "$CFILE" ] || { echo "[fft] introuvable: $CFILE"; exit 1; }
export CFILE="$(readlink -f "$CFILE")"

export RATE="${RATE:-1083333}"      # 4 SPS natif = 26e6/24
export NSAMP="${NSAMP:-32768}"      # complex samples par fenetre
export REFRESH="${REFRESH:-0.25}"   # periode de rafraichissement (s)
export ARFCN="${ARFCN:-514}"        # juste pour le titre
export FORMAT="${FORMAT:-cf32}"     # cf32 | cs8 | cs16
export DISPLAY="${DISPLAY:-:0}"
export FFT_SAVE=""
[ "${2:-}" = "--save" ] && export FFT_SAVE="${3:?--save needs a path}"

python3 - <<'PYEOF'
import os, numpy as np
import matplotlib
SAVE = os.environ.get("FFT_SAVE","")
if SAVE:
    matplotlib.use("Agg")
else:
    for bk in ("TkAgg","Qt5Agg","GTK3Agg"):
        try: matplotlib.use(bk); break
        except Exception: continue
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

CFILE = os.environ["CFILE"]
RATE  = float(os.environ["RATE"]); NS = int(os.environ["NSAMP"])
REFRESH = float(os.environ["REFRESH"]); ARFCN = os.environ["ARFCN"]
FORMAT = os.environ.get("FORMAT","cf32").lower()
NFFT  = 4096                        # resolution FFT
WROWS = 120                         # lignes du waterfall

# ---- format d'echantillon -> dtype brut + bytes/sample complexe ----
if FORMAT in ("cf32","fc32","complex64"):
    RAWDT, ITEMS = np.float32, 2          # I,F float32
elif FORMAT in ("cs16","sc16","int16"):
    RAWDT, ITEMS = np.int16, 2
elif FORMAT in ("cs8","sc8","int8"):
    RAWDT, ITEMS = np.int8, 2
else:
    raise SystemExit(f"FORMAT inconnu: {FORMAT} (cf32|cs16|cs8)")
SAMP_BYTES = np.dtype(RAWDT).itemsize * ITEMS     # octets / sample complexe
FSIZE = os.path.getsize(CFILE)
NSAMP_TOT = FSIZE // SAMP_BYTES
print(f"[fft] {CFILE}  {FSIZE} o  format={FORMAT}  {NSAMP_TOT} samples  Fs={RATE/1e6:.4f} MHz")
```

```
if NSAMP_TOT < NFFT:
    raise SystemExit("[fft] fichier trop court pour une FFT")

win = np.hanning(NFFT).astype(np.float32)
freqs = np.fft.fftshift(np.fft.fftfreq(NFFT, 1.0/RATE)) / 1e3   # kHz

_off = [0]    # offset courant en SAMPLES (balayage en boucle)

def grab():
    """Lit NS samples complexes a l'offset courant, avance, boucle a la fin."""
    start = _off[0]
    if start + NS > NSAMP_TOT:
        start = 0
    raw = np.fromfile(CFILE, dtype=RAWDT, count=NS*ITEMS,
                      offset=start*SAMP_BYTES)
    _off[0] = start + NS
    if len(raw) < NFFT*ITEMS:
        return None
    raw = raw.astype(np.float32)
    iq = raw[0::2] + 1j*raw[1::2]
    if RAWDT == np.int8:   iq /= 128.0
    if RAWDT == np.int16:  iq /= 32768.0
    return iq.astype(np.complex64)

def welch_psd(iq):
    """PSD moyennee (segments NFFT, overlap 50%, Hann) en dB."""
    step = NFFT//2; nseg = max(1,(len(iq)-NFFT)//step + 1)
    acc = np.zeros(NFFT, dtype=np.float64)
    used = 0
    for i in range(nseg):
        seg = iq[i*step:i*step+NFFT]
        if len(seg) < NFFT: break
        sp = np.fft.fftshift(np.fft.fft(seg*win))
        acc += (sp.real**2 + sp.imag**2); used += 1
    psd = acc/max(1,used)
    return 10*np.log10(psd/(NFFT*RATE) + 1e-20)

# ---- figure "jolie" : PSD (haut) + waterfall (bas) alignees ----
plt.style.use("dark_background")
fig = plt.figure(figsize=(11,8))
gs = fig.add_gridspec(2, 2, width_ratios=[40, 1], height_ratios=[1, 1.4],
                      hspace=0.18, wspace=0.015)
axp = fig.add_subplot(gs[0, 0])
axw = fig.add_subplot(gs[1, 0], sharex=axp)
cax = fig.add_subplot(gs[1, 1])
fig.suptitle(f"FFT - {os.path.basename(CFILE)}  |  ARFCN={ARFCN}  Fs={RATE/1e6:.4f} MHz  fmt={FORMAT}",
             fontsize=11, color="#7fd")
(line,) = axp.plot(freqs, np.full(NFFT,-120), lw=0.5, color="#39f")
axp.set_xlim(freqs[0],freqs[-1]); axp.set_ylim(-130,-30)
axp.set_ylabel("PSD (dB)"); axp.grid(alpha=0.25)
axp.axvline(0, color="#f55", lw=0.7, alpha=0.7)
plt.setp(axp.get_xticklabels(), visible=False)

wf = np.full((WROWS,NFFT), -120.0)
im = axw.imshow(wf, aspect="auto", origin="upper", cmap="turbo",
                extent=[freqs[0],freqs[-1],0,WROWS], vmin=-120, vmax=-50,
                interpolation="bilinear")
axw.set_ylabel("temps (fenetres)"); axw.set_xlabel("offset (kHz)")
axw.axvline(0, color="#fff", lw=0.7, alpha=0.5)
fig.colorbar(im, cax=cax, label="dB")
fig.subplots_adjust(left=0.08, right=0.92, top=0.92, bottom=0.08)

def update(_):
    iq = grab()
    if iq is None: return line, im
    psd = welch_psd(iq)
```

```
    line.set_ydata(psd)
    lo,hi = np.percentile(psd,5), np.percentile(psd,99.5)
    axp.set_ylim(lo-5, hi+8)
    global wf
    wf = np.roll(wf,1,axis=0); wf[0,:] = psd
    im.set_data(wf); im.set_clim(lo, hi+5)
    return line, im

if SAVE:
    update(0)
    fig.savefig(SAVE, dpi=110, facecolor=fig.get_facecolor())
    print("ecrit", SAVE)
else:
    ani = FuncAnimation(fig, update, interval=int(REFRESH*1000),
                        blit=False, cache_frame_data=False)
    print("FFT - ferme la fenetre pour quitter.")
    plt.show()
PYEOF
```

3.393 /opt/GSM/osmo-operator/tools/glava-build.sh

6122 octets, 117 lignes → 117 lignes

```
#!/bin/bash
# glava-build.sh - GLava + le module « NCS Spectrum », compiles et poses.
#
#     tools/glava-build.sh            sur la machine (/), en root
#     tools/glava-build.sh $ROOTFS    dans le rootfs de l ISO (iso_modules/85-glava.sh)
#
# [2026-10-23] Le visualiseur audio du bureau : le cercle « NCS Spectrum »
# (particules qui tournent autour d un disque, sur le son qui sort des
# haut-parleurs), pose en fond de bureau. Deux depots, figes a un commit :
#   - GLava (GPL-3) : pas de paquet dans noble, on le compile. Le module NCS a
#     besoin d un patch de glava/render.c (textures image pour les particules,
#     imageLoad/imageStore - OpenGL >= 4.2), fourni par le depot NCS ;
#   - NCS_Spectrum_GLava : ncs.glsl et ncs/ vont a cote des modules de GLava,
#     dans /etc/xdg/glava. Ce depot n a PAS de licence : il est clone au build,
#     pas copie dans le notre.
# rc.glsl (systeme) est regle comme le demande le README NCS : module ncs,
# OpenGL 4.5 / GLSL 450, setmirror (les deux canaux), fenetre de bureau sans
# bord ni clic. La position et la taille viennent de tools/osmo-glava.sh, qui
# les calcule d apres l ecran au lancement.
#
# Compilation sur l HOTE (meme Ubuntu que l image), installation par DESTDIR :
# pas de paquets -dev a installer puis retirer dans le rootfs.
set -euo pipefail

DEST="${1:-}"
DIR="$(cd "$(dirname "$0")/.." && pwd)"
GLAVA_REPO="${GLAVA_REPO:-https://gitlab.com/wild-turtles-publicly-release/glava/glava.git}"
GLAVA_COMMIT="${GLAVA_COMMIT:-e03a8a5b721f393dd7b80738a6ac5c9bccc8f050}"
NCS_REPO="${NCS_REPO:-https://github.com/Roonil/NCS_Spectrum_GLava.git}"
NCS_COMMIT="${NCS_COMMIT:-09844550865575 84e53da12915e9dbd24255fed2}"

manque=""
for t in git meson ninja cc pkg-config patch; do
    command -v "$t" >/dev/null 2>&1 || manque="$manque $t"
done
for h in pulse/simple.h X11/Xlib.h X11/extensions/Xcomposite.h X11/extensions/Xrender.h X11/extensions/shape.h
 ↪ GL/gl.h; do
    [ -f "/usr/include/$h" ] || manque="$manque $h"
done
if [ -n "$manque" ]; then
    echo "glava-build : manque$manque" >&2
```

```
    echo "  apt-get install -y git meson ninja-build gcc pkg-config patch libpulse-dev libx11-dev libxext-dev
     ↪  libxrender-dev libxcomposite-dev libgl-dev" >&2
    exit 1
fi

TMP="$(mktemp -d)"
trap 'rm -rf "$TMP"' EXIT
clone() {   # depot commit dossier
    git init -q "$3"
    git -C "$3" fetch -q --depth 1 "$1" "$2"
    git -C "$3" checkout -q FETCH_HEAD
}
clone "$GLAVA_REPO" "$GLAVA_COMMIT" "$TMP/glava"
clone "$NCS_REPO" "$NCS_COMMIT" "$TMP/ncs"

patch -s -d "$TMP/glava" glava/render.c < "$TMP/ncs/imageOperations.patch"
# -Dobs=false : le greffon OBS demande libobs, sans usage ici
meson setup "$TMP/glava/build" "$TMP/glava" --prefix /usr -Dobs=false >/dev/null
ninja -C "$TMP/glava/build" >/dev/null
DESTDIR="$DEST" ninja -C "$TMP/glava/build" install >/dev/null

XDG="$DEST/etc/xdg/glava"
cp -r "$TMP/ncs/ncs" "$TMP/ncs/ncs.glsl" "$XDG/"
# [2026-10-24] Particules en ORANGE (celui du schema BTS -> Core du fond,
# 255/140/40), a la meme intensite que le bleu d origine (0.0118, 0.1412,
# 0.3412) : le melange est additif, une couleur plus forte sature en blanc.
sed -i 's/^#define color vec3(0.0118, 0.1412, 0.3412)/#define color vec3(0.3412, 0.1873, 0.0535)/' "$XDG/ncs.glsl"
grep -q '^#define color vec3(0.3412, 0.1873, 0.0535)' "$XDG/ncs.glsl" \
    || echo "glava-build : couleur NCS non changee (ncs.glsl a bouge ?)" >&2
# [2026-10-24] La bande du bas (module bars, lancee par osmo-glava a cote du
# cercle) : barres epaisses (12 px, ecart 3), bleu clair, fond transparent.
sed -i -e 's/^#define BAR_WIDTH 5$/#define BAR_WIDTH 12/' \
       -e 's/^#define BAR_GAP 1$/#define BAR_GAP 3/' \
       -e 's/^#define USE_ALPHA 0$/#define USE_ALPHA 1/' \
       -e 's/^#define COLOR @fg:mix(#3366b2, #a0a0b2, clamp(d \/ GRADIENT, 0, 1))$/#define COLOR @fg:mix(#78c3ff,
        ↪  #d6eeff, clamp(d \/ GRADIENT, 0, 1))/' \
       "$XDG/bars.glsl"
grep -q '^#define COLOR @fg:mix(#78c3ff' "$XDG/bars.glsl" \
    || echo "glava-build : couleur des barres non changee (bars.glsl a bouge ?)" >&2
sed -i -e 's/^#request mod bars/#request mod ncs/' \
       -e 's/^#request setdecorated true/#request setdecorated false/' \
       -e 's/^#request setmirror false/#request setmirror true/' \
       -e 's/^#request setversion 3 3/#request setversion 4 5/' \
       -e 's/^#request setshaderversion 330/#request setshaderversion 450/' \
       -e 's/^#request setxwintype "normal"/#request setxwintype "desktop"/' \
       -e 's/^#request setclickthrough false/#request setclickthrough true/' \
       "$XDG/rc.glsl"

install -Dm755 "$DIR/tools/osmo-glava.sh" "$DEST/usr/local/bin/osmo-glava"
# [2026-10-24] UN SERVICE UTILISATEUR, PLUS UN AUTOSTART. Lance par un .desktop
# de /etc/xdg/autostart, GLava n avait pas de commande d arret : il fallait
# pkill. En service systemd --user, cercle ET barres sont dans le meme cgroup :
#   systemctl --user stop glava      arrete les deux (et retire le mixage)
#   systemctl --user start glava     les relance
#   systemctl --user disable glava   plus de GLava a la connexion (ce compte)
# Active pour tous les comptes (graphical-session.target.wants dans
# /etc/systemd/user) : demarre avec la session graphique, qui fournit DISPLAY.
rm -f "$DEST/etc/xdg/autostart/osmo-glava.desktop"
install -d "$DEST/etc/systemd/user/graphical-session.target.wants"
cat > "$DEST/etc/systemd/user/glava.service" <<'UNIT'
[Unit]
Description=GLava : cercle NCS et barres du bas (micro + haut-parleurs)
PartOf=graphical-session.target
After=graphical-session.target pulseaudio.service pipewire-pulse.service

[Service]
```

```
ExecStart=/usr/local/bin/osmo-glava
ExecStopPost=/usr/local/bin/osmo-glava --retirer-mix
Restart=on-failure
RestartSec=5

[Install]
WantedBy=graphical-session.target
UNIT
chmod 644 "$DEST/etc/systemd/user/glava.service"
ln -sfn ../glava.service "$DEST/etc/systemd/user/graphical-session.target.wants/glava.service"

if [ -n "$DEST" ]; then chroot "$DEST" ldconfig 2>/dev/null || true; else ldconfig; fi
echo "GLava + NCS : $(git -C "$TMP/glava" rev-parse --short HEAD) / $(git -C "$TMP/ncs" rev-parse --short HEAD) ->
 ↪  ${DEST:-/}"
```

3.394 /opt/GSM/osmo-operator/tools/globals-complet.py

9210 octets, 199 lignes → 199 lignes

```
#!/usr/bin/env python3
# [2026-10-21] Les PATRONS de la page web (web/, servie par tools/osmo-reglages.py) :
#
#   globals.conf.example          les reglages reseau de globals.conf + les variables du LANCEUR
#                                 (start.sh, startup_scripts/*.sh) : globales a la chaine
#   instances/instance.env.example  les variables du BANC DSP (c54x_exe/run.sh) : UNE VALEUR PAR INSTANCE
#                                 Calypso (1 = BTS/TRX ou MS sur le pont, 2 = MS QEMU par l'air, 3 = TRX TS1),
#                                 ecrites par la page dans instances/instance-N.env et lues par run.sh
#
#   python3 tools/globals-complet.py            # (re)genere les deux patrons
#   python3 tools/globals-complet.py --voir     # les affiche sans rien ecrire
#
# Chaque variable porte son defaut en commentaire (fichier:ligne) et, quand les scripts la comparent a des
# valeurs fixes (0/1, dsp/grgsm, ...), une ligne « choix : a | b | c » : la page en fait un menu deroulant.
# Valeur VIDE = defaut du script. globals.conf est source avec `set -a` apres le traitement des options
# (03-affichage.sh) : une variable deja posee et non vide (ligne de commande) garde sa valeur ; la page ecrit
# commentees les variables du lanceur laissees vides (exportees vides, elles cassaient le lancement).
#
# Exclus : l'etat interne du lanceur (C_*, *_CHOISI, chemins calcules), les variables lues sans deux-points
# (${VAR-x} : vide et absent different). Pour run.sh, seules les variables qu'il renormalise lui-meme
# (VAR="${VAR:-x}", export compris) : une valeur vide n'atteint jamais le C.

import argparse, glob, os, re, shutil, time

RACINE = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
GLOBALS = os.path.join(RACINE, "globals.conf")
EXEMPLE = GLOBALS + ".example"
INSTANCES = os.path.join(RACINE, "instances")
EXEMPLE_INST = os.path.join(INSTANCES, "instance.env.example")
RUN_DSP = "/opt/GSM/c54x_exe/run.sh"
DEBUT = "# >>> VARIABLES DU LANCEUR : bloc genere par tools/globals-complet.py (ne pas editer les marqueurs) >>>"
FIN = "# <<< FIN DES VARIABLES DU LANCEUR <<<"

INTERNES = {"RUNTIME_ENV", "OQC_ROOT", "NITB_TREE", "RAND", "NODE_ID_SRC", "GEN_CONF_OUT", "SAP_FILE",
            "SOFTSIM_KI", "HUB_IP", "NO_COLOR", "RUN_SH", "HERE", "RUNDIR", "INSTANCE", "CALYPSO_NS"}
# choix a la main (les scripts ne les comparent pas tous a des valeurs fixes)
CHOIX = {
    "BAND": "GSM900 | DCS1800 | PCS1900 | GSM850",
    "ENCRYPTION": "a5 0 | a5 1 | a5 3 | a5 0 1 | a5 0 1 3",
    "SIM_ALGO": "comp128 | comp128v2 | comp128v3 | xor | milenage",
    "CALYPSO_PROFILE": "faketrx-qemu | faketrx | qemu | noproc",
    "MODE": "dsp | grgsm",
    "OSMO_TRX_TS": "1 | 2 | 2/1",
    "SAP_TYPE": "auto | pcsc | softsim",
}
```

```
DEFAUT = re.compile(r'\$\{([A-Z][A-Z0-9_]*):([-=])')
RENORMALISE = re.compile(r'^\s*(?:export\s+)?([A-Z][A-Z0-9_]*)="?\$\{\1:-')


def defaut_apres(ligne, i):
    """Le texte du defaut d'un ${VAR:-...} a partir de l'indice i, accolades equilibrees."""
    prof, j = 1, i
    while j < len(ligne) and prof:
        if ligne[j] == "{":
            prof += 1
        elif ligne[j] == "}":
            prof -= 1
        j += 1
    return ligne[i:j - 1]


def extraire(fichier, seulement_renormalisees=False):
    vus = {}
    for n, l in enumerate(open(fichier, encoding="utf-8", errors="replace").read().split("\n"), 1):
        if l.lstrip().startswith("#"):
            continue
        m_r = RENORMALISE.match(l)
        if seulement_renormalisees and not m_r:
            continue
        for m in DEFAUT.finditer(l):
            v = m.group(1)
            if seulement_renormalisees and v != m_r.group(1):
                continue
            if v not in vus:
                vus[v] = (defaut_apres(l, m.end()), n)
    return vus


def sans_deux_points(fichiers):
    r = set()
    for f in fichiers:
        r |= set(re.findall(r'\$\{([A-Z][A-Z0-9_]*)[-=]', open(f, encoding="utf-8", errors="replace").read()))
    return r


def choix_auto(textes):
    """Les valeurs fixes auxquelles les scripts comparent chaque variable : [ "$VAR" = 1 ], != "off"..."""
    comp = {}
    pat = re.compile(r'\$\{?([A-Z][A-Z0-9_]*)(?::-[^}]*)?\}?"?\s*(?:=|==|!=)\s*"?([A-Za-z0-9_./-]+)"?')
    for t in textes:
        for l in t.split("\n"):
            if l.lstrip().startswith("#"):
                continue
            for m in pat.finditer(l):
                comp.setdefault(m.group(1), []).append(m.group(2))
    return comp


def choix_de(var, defaut, comp):
    if var in CHOIX:
        return CHOIX[var]
    vals = []
    for v in comp.get(var, []):
        if v not in vals:
            vals.append(v)
    if defaut in ("0", "1") and all(v in ("0", "1") for v in vals):
        return "0 | 1"
    if len(vals) >= 2 and all(len(v) <= 24 for v in vals):
        if defaut and defaut not in vals and re.match(r'^[A-Za-z0-9_./-]+$', defaut):
            vals.insert(0, defaut)
        return " | ".join(vals)
```

```
    return ""


def titre(nom):
    return "# ── %s %s" % (nom, "─" * max(3, 58 - len(nom)))


def bloc_variables(sources, exclues, comp, ligne_var):
    deja, lignes, n = set(), [], 0
    for f, renorm in sources:
        vars_f = extraire(f, renorm)
        garde = [(v, d, l) for v, (d, l) in vars_f.items()
                 if v not in exclues and v not in deja and not v.startswith("C_") and not v.endswith("_CHOISI")]
        if not garde:
            continue
        nom = os.path.relpath(f, RACINE) if f.startswith(RACINE) else f
        lignes += ["", titre(nom), ""]
        for v, d, l in garde:
            deja.add(v)
            d = d.replace("\n", " ")
            lignes.append("# defaut : %s   (%s:%d)" % (d if d else "(vide)", os.path.basename(f), l))
            c = choix_de(v, d, comp)
            if c:
                lignes.append("# choix : %s" % c)
            lignes.append(ligne_var(v))
            n += 1
    return lignes, n


def tete_globals():
    texte = open(GLOBALS, encoding="utf-8").read()
    if DEBUT in texte and FIN in texte:
        tete, reste = texte.split(DEBUT, 1)
        queue = reste.split(FIN, 1)[1]
    else:
        tete, queue = texte, ""
    # le patron porte les reglages reseau VIDES de valeur ? Non : on garde les lignes telles quelles mais on
    # ajoute les choix a la main ; la page lit les valeurs dans globals.conf, pas ici.
    sortie = []
    for l in tete.rstrip("\n").split("\n"):
        m = re.match(r"^([A-Z_][A-Z0-9_]*)=", l)
        if m and m.group(1) in CHOIX and (not sortie or not sortie[-1].startswith("# choix :")):
            sortie.append("# choix : " + CHOIX[m.group(1)])
        sortie.append(l)
    declarees = set(re.findall(r"^([A-Z_][A-Z0-9_]*)=", tete + queue, re.M))
    return "\n".join(sortie), queue.strip("\n"), declarees


def generer():
    modules = [os.path.join(RACINE, "start.sh")] + sorted(glob.glob(os.path.join(RACINE, "startup_scripts",
↪   "*.sh")))
    textes = [open(f, encoding="utf-8", errors="replace").read() for f in modules + [RUN_DSP] if os.path.exists(f)]
    comp = choix_auto(textes)
    tete, queue, declarees = tete_globals()
    exclues = sans_deux_points(modules + [RUN_DSP]) | INTERNES | declarees
    lignes, n = bloc_variables([(m, False) for m in modules], exclues, comp, lambda v: '%s=""' % v)
    corps = [DEBUT, "#", "#  Valeur VIDE = defaut du script (en commentaire). Regenerer : python3
↪   tools/globals-complet.py", "#"] + lignes + ["", FIN]
    globals_txt = tete + "\n\n" + "\n".join(corps) + ("\n\n" + queue if queue else "") + "\n"

    inst_tete = [
        "# instances/instance.env.example — patron des reglages PAR INSTANCE Calypso (c54x_exe/run.sh).",
        "# Genere par tools/globals-complet.py ; la page web (tools/osmo-reglages.py) ecrit instance-N.env :",
        "#   1 = la radio de la BTS (TRX Calypso, --bts) ou le mobile sur le pont (--bts-classique)",
        "#   2 = le mobile QEMU par l air (--bts --ms, --ms-magnetite)",
        "#   3 = le 2e TRX Calypso, TS1 (--ts 2)",
```

```
        "# run.sh lit instances/instance-$INSTANCE.env au demarrage, sous la forme : \"${VAR:=valeur}\" :",
        "# l environnement (start-native.sh, ligne de commande) garde la priorite, la valeur du fichier vient",
        "# ensuite, puis le defaut de run.sh. Valeur VIDE = defaut.",
    ]
    lignes_i, n_i = bloc_variables([(RUN_DSP, True)], sans_deux_points([RUN_DSP]) | INTERNES, comp, lambda v:
↪   '%s=""' % v)
    inst_txt = "\n".join(inst_tete + lignes_i) + "\n"
    return globals_txt, n, inst_txt, n_i


if __name__ == "__main__":
    a = argparse.ArgumentParser(description="patrons de la page de reglages")
    a.add_argument("--voir", action="store_true")
    o = a.parse_args()
    g, n, i, n_i = generer()
    if o.voir:
        print(g)
        print(i)
    else:
        for chemin, texte in ((EXEMPLE, g), (EXEMPLE_INST, i)):
            os.makedirs(os.path.dirname(chemin), exist_ok=True)
            if os.path.exists(chemin):
                shutil.copy2(chemin, chemin + ".bak-" + time.strftime("%Y%m%d-%H%M%S"))
            open(chemin, "w", encoding="utf-8").write(texte)
        print("globals.conf.example : %d variables du lanceur ; instances/instance.env.example : %d variables de
        ↪  run.sh" % (n, n_i))
```

3.395 /opt/GSM/osmo-operator/tools/gnome-extensions-install.sh

2210 octets, 42 lignes → 42 lignes

```
#!/bin/bash
# gnome-extensions-install.sh - extensions GNOME Shell actives par defaut.
#
#     tools/gnome-extensions-install.sh            sur la machine (/), en root
#     tools/gnome-extensions-install.sh $ROOTFS    dans le rootfs de l ISO
#                                                  (iso_modules/85-extensions-gnome.sh)
#
# [2026-10-23] EasyScreenCast (GPL-3, extensions.gnome.org n° 690) : un menu
# dans la barre du haut pour l enregistreur d ecran de GNOME (son, webcam,
# zone, minuterie). Pas de paquet : le zip de la version 53 (GNOME 46 a 48)
# est telecharge, verifie par son sha256 et pose pour tous les comptes dans
# /usr/share/gnome-shell/extensions.
# Le pont navigateur chrome-gnome-shell (gnome-browser-connector) est un
# paquet apt : il est dans la liste du bureau (iso_modules/80-chroot.sh).
#
# « Active par defaut » = enabled-extensions dans un override de schema, en
# section nue ET :ubuntu (la session ubuntu lit celle-la d abord - meme raison
# que le fond et le dock dans 80-chroot.sh). Un compte qui a deja sa propre
# liste (gsettings set ...) la garde : l override n est qu un defaut.
set -euo pipefail

DEST="${1:-}"
ESC_UUID="EasyScreenCast@iacopodeenosee.gmail.com"
ESC_URL="https://extensions.gnome.org/download-extension/$ESC_UUID.shell-extension.zip?version_tag=63091"
ESC_SHA256="e0fce1b69cf05eb365fe3d4a3e8c627c6a93beffc18a042b7896249961bc6777"

TMP="$(mktemp -d)"
trap 'rm -rf "$TMP"' EXIT
curl -fsSL -o "$TMP/esc.zip" "$ESC_URL"
echo "$ESC_SHA256  $TMP/esc.zip" | sha256sum -c --quiet -

EXT="$DEST/usr/share/gnome-shell/extensions/$ESC_UUID"
rm -rf "$EXT"
```

```
install -d "$EXT"
python3 -I -c "import sys, zipfile; zipfile.ZipFile(sys.argv[1]).extractall(sys.argv[2])" "$TMP/esc.zip" "$EXT"
chmod -R u=rwX,go=rX "$EXT"

SCH="$DEST/usr/share/glib-2.0/schemas"
printf "[org.gnome.shell]\nenabled-extensions=['%s']\n\n[org.gnome.shell:ubuntu]\nenabled-extensions=['%s']\n" \
    "$ESC_UUID" "$ESC_UUID" > "$SCH/99-osmo-extensions.gschema.override"
if [ -n "$DEST" ]; then chroot "$DEST" glib-compile-schemas /usr/share/glib-2.0/schemas; else
 ↪  glib-compile-schemas "$SCH"; fi
echo "EasyScreenCast v53 -> ${DEST:-/}usr/share/gnome-shell/extensions, active par defaut"
```

3.396 /opt/GSM/osmo-operator/tools/grgsm_relay_decode.py

3118 octets, 70 lignes → 70 lignes

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#
# tools/grgsm_relay_decode.py — décodeur gr-gsm sur l'I/Q CONTINU relayée (mode
# full-grgsm). Lit l'I/Q du BTS relayée par calypso-ipc-device (UDP 5810,
# fc32), la passe dans le receiver gr-gsm complet (gsm.receiver : FCCH/SCH
# sync + démod GMSK), décode le BCCH/CCCH, et envoie le L2 en GSMTAP au
# listener du shunt (4730) → feed_si → a_cd → le mobile (sur osmocon) campe.
#
# C'est un DÉCODEUR (pas un transceiver) : pas besoin de trxcon. La chaîne
# est celle de grgsm_decode mais avec une source UDP au lieu d'un fichier.
#
#   udp_source(5810, fc32) → lpf → gsm.receiver(osr) → C0 bursts →
#       gsm_bcch_ccch_demapper → control_channels_decoder → GSMTAP(4730)
#
# osr DOIT matcher le SPS d'osmo-trx (tx-sps=1 → osr=1). Si la sync n'accroche
# pas à osr=1, passer osmo-trx en SPS=4 et CALYPSO_TRX_OSR=4.
import os, sys

from gnuradio import gr, network, filter
from gnuradio.filter import firdes
from gnuradio.fft import window
from gnuradio import gsm

# OSR = oversampling vu par gsm.receiver. Il a besoin de ~4 pour locker la
# FCCH/SCH et la récupération de timing GMSK. L'I/Q relayée est à 1 SPS
# (osmo-trx tx-sps=1) → on RESAMPLE 1→OSR dans le flowgraph (pas de chirurgie
# device, pas de SPS=4 côté osmo-trx).
OSR        = int(os.environ.get("CALYPSO_TRX_OSR", "4"))
IN_SPS     = int(os.environ.get("CALYPSO_TRX_IN_SPS", "1"))   # SPS du relais (osmo-trx)
SYM_RATE   = 1625000.0 / 6.0                       # 270833.33 Hz
SAMP_RATE  = SYM_RATE * OSR                         # rate après resampling
RX_PORT    = int(os.environ.get("CALYPSO_TRX_IQ_RX_PORT", "5810"))
GSMTAP_HOST = os.environ.get("GSMTAP_HOST", "127.0.0.1")
GSMTAP_PORT = int(os.environ.get("CALYPSO_SHUNT_GSMTAP_PORT", "4730"))
TS         = int(os.environ.get("TS", "0"))

tb = gr.top_block("grgsm relay decode")

# Source : I/Q continu relayé (fc32, 625 samples/datagram = 5000 o). src_zeros
# → flux continu si pas de data (le receiver ne stalle pas).
src = network.udp_source(gr.sizeof_gr_complex, 1, RX_PORT,
                         0, 5000, False, True, False)

# Resampling 1→OSR (IN_SPS→OSR) : donne à gsm.receiver les ~4 SPS qu'il attend.
resamp = filter.rational_resampler_ccc(interpolation=OSR, decimation=IN_SPS)

lpf = filter.fir_filter_ccf(1, firdes.low_pass(
    1, SAMP_RATE, 125e3, 5e3, window.WIN_HAMMING, 6.76))
```

```
rx  = gsm.receiver(OSR, [0], [])              # cell_allocation [0], tseq []
dm  = gsm.gsm_bcch_ccch_demapper(TS)
dec = gsm.control_channels_decoder()
snk = network.socket_pdu("UDP_CLIENT", GSMTAP_HOST, str(GSMTAP_PORT), 10000)

tb.connect((src, 0), (resamp, 0))
tb.connect((resamp, 0), (lpf, 0))
tb.connect((lpf, 0), (rx, 0))
tb.msg_connect((rx, 'C0'), (dm, 'bursts'))
tb.msg_connect((dm, 'bursts'), (dec, 'bursts'))
tb.msg_connect((dec, 'msgs'), (snk, 'pdus'))

print("[grgsm-relay-decode] I/Q udp:%d (osr=%d, %.0f Hz) -> GSMTAP %s:%d -> feed_si"
      % (RX_PORT, OSR, SAMP_RATE, GSMTAP_HOST, GSMTAP_PORT), flush=True)

tb.start()
try:
    tb.wait()
except KeyboardInterrupt:
    tb.stop(); tb.wait()
```

3.397 /opt/GSM/osmo-operator/tools/grub-theme-render.py

6769 octets, 163 lignes → 163 lignes

```
#!/usr/bin/env python3
# grub-theme-render.py - fabrique le theme GRUB « osmo ».
#
#     python3 tools/grub-theme-render.py --out /boot/grub/themes/osmo
#
# [2026-10-23] Comme le theme Plymouth (tools/plymouth-render.py) : LE THEME
# EST GENERE, PAS STOCKE. Le dessin est le code ; on change une couleur, on
# rejoue le script. Appele par iso_modules/86-finitions.sh (systeme installe)
# et iso_modules/90-iso.sh (menu du live).
#
# CE QU ON VOIT :
#   - en fond, la ville du fond d ecran (configs/gsm-lab-ville.jpg) floutee et
#     assombrie, dans le meme degrade violet sombre -> bleu ;
#   - en haut, « osmo-operator » et, en orange, « localhost · no radio emission »
#     (le cadre du schema du bureau) ;
#   - au centre, le menu dans un panneau translucide a bord bleu ; l entree
#     choisie est une pastille en degrade violet -> bleu ;
#   - en bas, le compte a rebours (orange) et l aide clavier.
#
# Les polices (.pf2) sont faites par grub-mkfont (grub-common) depuis Ubuntu
# (fonts-ubuntu) : la meme famille que le bureau.
import argparse
import os
import subprocess
from PIL import Image, ImageDraw, ImageFilter, ImageFont

REPO = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
FONTS = "/usr/share/fonts/truetype/ubuntu"
W, H = 1920, 1080
ORANGE = (255, 140, 40)
VIOLET = (58, 30, 110)
BLEU = (70, 150, 245)
BORD = (110, 175, 255)

p = argparse.ArgumentParser()
p.add_argument("--out", required=True)
p.add_argument("--fond", default=os.path.join(REPO, "configs", "gsm-lab-ville.jpg"))
a = p.parse_args()
```

```
os.makedirs(a.out, exist_ok=True)
out = lambda n: os.path.join(a.out, n)


def police(f, t):
    return ImageFont.truetype(os.path.join(FONTS, f), t)


# ── fond ──────────────────────────────────────────────────────────────
src = Image.open(a.fond).convert("RGB")
r = max(W / src.width, H / src.height)
src = src.resize((round(src.width * r), round(src.height * r)), Image.LANCZOS)
x0, y0 = (src.width - W) // 2, (src.height - H) // 2
fond = src.crop((x0, y0, x0 + W, y0 + H)).filter(ImageFilter.GaussianBlur(30))
# voile : violet sombre en haut a gauche -> bleu nuit en bas a droite
voile = Image.new("RGB", (W, H))
dv = ImageDraw.Draw(voile)
for i in range(W + H):
    t = i / (W + H)
    c = (round(22 * (1 - t) + 6 * t), round(8 * (1 - t) + 14 * t), round(44 * (1 - t) + 40 * t))
    dv.line([(i, 0), (i - H, H)], fill=c, width=2)
fond = Image.blend(fond, voile, 0.62)

d = ImageDraw.Draw(fond)
titre = police("Ubuntu-B.ttf", 74)
d.text((W / 2, 150), "osmo-operator", font=titre, fill=(245, 245, 250), anchor="mm")
sous = police("Ubuntu-M.ttf", 30)
d.text((W / 2, 222), "localhost  ·  no radio emission", font=sous, fill=ORANGE, anchor="mm")
lw = d.textlength("localhost  ·  no radio emission", font=sous)
d.line([(W / 2 - lw / 2, 252), (W / 2 + lw / 2, 252)], fill=ORANGE + (255,), width=2)
fond.save(out("background.png"), optimize=True)


# ── panneau du menu (9 morceaux : menu_*.png) ─────────────────────────
def neuf(prefixe, taille, rayon, remplissage, bord, epais):
    """Dessine un rectangle arrondi et le coupe en 9 morceaux pour GRUB."""
    s = taille
    im = Image.new("RGBA", (3 * s, 3 * s), (0, 0, 0, 0))
    ImageDraw.Draw(im).rounded_rectangle((0, 0, 3 * s - 1, 3 * s - 1), radius=rayon,
                                         fill=remplissage, outline=bord, width=epais)
    noms = {"nw": (0, 0), "n": (1, 0), "ne": (2, 0), "w": (0, 1), "c": (1, 1),
            "e": (2, 1), "sw": (0, 2), "s": (1, 2), "se": (2, 2)}
    for nom, (i, j) in noms.items():
        im.crop((i * s, j * s, (i + 1) * s, (j + 1) * s)).save(out("%s_%s.png" % (prefixe, nom)))


neuf("menu", 24, 22, (12, 8, 30, 175), BORD + (200,), 2)

# ── entree choisie (select_w / select_c / select_e) : pastille en degrade ──
HS = 44
pastille = Image.new("RGBA", (400, HS), (0, 0, 0, 0))
masque = Image.new("L", (400, HS), 0)
ImageDraw.Draw(masque).rounded_rectangle((0, 0, 399, HS - 1), radius=HS // 2, fill=255)
deg = Image.new("RGBA", (400, HS))
dd = ImageDraw.Draw(deg)
for x in range(400):
    t = x / 399
    dd.line([(x, 0), (x, HS)], fill=tuple(round(VIOLET[k] * (1 - t) + BLEU[k] * t) for k in range(3)) + (235,))
pastille.paste(deg, (0, 0), masque)
pastille.crop((0, 0, HS // 2, HS)).save(out("select_w.png"))
pastille.crop((HS // 2, 0, 400 - HS // 2, HS)).save(out("select_c.png"))
pastille.crop((400 - HS // 2, 0, 400, HS)).save(out("select_e.png"))

# ── polices ───────────────────────────────────────────────────────────
# -r : ASCII, Latin-1 et Latin etendu A (les accents des entrees), le point
# median de l aide. -n fixe le nom : « Osmo Regular 24 ».
```

```
plages = "0x20-0x7E,0xA0-0x17F,0x2022-0x2022"
for fichier, nom, taille in (("Ubuntu-R.ttf", "Osmo", 24), ("Ubuntu-M.ttf", "OsmoM", 22),
                             ("Ubuntu-R.ttf", "Osmo", 18)):
    subprocess.run(["grub-mkfont", "-s", str(taille), "-n", nom, "-r", plages,
                    "-o", out("%s-%d.pf2" % (nom.lower(), taille)), os.path.join(FONTS, fichier)],
                   check=True)

# ── theme.txt ──────────────────────────────────────────────────────────
theme = """# GENERE par tools/grub-theme-render.py - ne pas modifier a la main.
title-text: ""
desktop-image: "background.png"
desktop-color: "#0b0820"
terminal-font: "Osmo Regular 18"
terminal-box: "menu_*.png"

+ boot_menu {
    left = 27%
    top = 33%
    width = 46%
    height = 42%
    menu_pixmap_style = "menu_*.png"
    item_font = "Osmo Regular 24"
    item_color = "#c9cde0"
    selected_item_font = "OsmoM Regular 22"
    selected_item_color = "#ffffff"
    selected_item_pixmap_style = "select_*.png"
    item_height = 44
    item_spacing = 8
    item_padding = 18
    icon_width = 0
    icon_height = 0
    item_icon_space = 0
    scrollbar = false
}

+ label {
    id = "__timeout__"
    left = 0
    top = 84%
    width = 100%
    align = "center"
    font = "Osmo Regular 24"
    color = "#ff8c28"
    text = "Démarrage automatique dans %d s"
}

+ label {
    left = 0
    top = 91%
    width = 100%
    align = "center"
    font = "Osmo Regular 18"
    color = "#8f96b8"
    text = "flèches  choisir    •    Entrée  démarrer    •    e  modifier    •    c  console"
}
"""
with open(out("theme.txt"), "w") as f:
    f.write(theme)
print(a.out)
```

3.398 /opt/GSM/osmo-operator/tools/make-docker-image.sh

527 octets, 11 lignes → 11 lignes

```
#!/bin/bash
# tools/make-docker-image.sh - construit l image docker osmocom-nitb.
#
# [2026-09-03] Ce script etait une copie de build.sh, moins toast et --lite,
# et les deux divergeaient. Il delegue desormais a build.sh : apt-fast,
# docker compose v2 + buildx dans les dependances, cache .deb sur l hote
# (/var/cache/osmo-debs). Tous les arguments sont transmis tels quels.
#
#   sudo ./tools/make-docker-image.sh [--no-cache] [--lite] [--stp]
set -euo pipefail
exec bash "$(dirname "$(readlink -f "$0")")/../build.sh" "$@"
```

3.399 /opt/GSM/osmo-operator/tools/matrix.sh

6147 octets, 119 lignes → 119 lignes

```
#!/usr/bin/env bash
# tools/matrix.sh <burst|si> - MATRICE DYNAMIQUE du treillis 51-multiframe (cote mobile).
#
# Deux vues "temps/canal" complementaires des 2 FFT (frequence) :
#   tools/matrix.sh burst  -> dynamique des BURSTS (TRXD ts0 DL :5702) :
#                       lignes = FN%51 (structure canal), colonnes = temps,
#                       couleur = vide / dummy(idle) / reel(actif).
#   tools/matrix.sh si     -> dynamique des SI/L2 decodes (GSMTAP :4729/:4730) :
#                       lignes = FN%51, couleur = type de message (SI/IMM/UA/PAGING/?).
#
# Sniff PASSIF (AF_PACKET via gsm_sniff.py dans le conteneur, ne vole rien, aucune
# fifo) -> pipe -> rendu matplotlib sur l'hote (X). Ferme la fenetre pour quitter.
#
# Usage : ./tools/matrix.sh burst    |    ./tools/matrix.sh si
#         CONTAINER=osmo-operator-66 TS=0 W=240 ./tools/matrix.sh burst
set -euo pipefail
MODE="${1:?usage: tools/matrix.sh <burst|si>}"
case "$MODE" in burst|si) ;; *) echo "mode invalide: $MODE (burst|si)"; exit 1;; esac
export MODE
export CONTAINER="${CONTAINER:-osmo-operator-66}"
export TS="${TS:-0}"          # timeslot a afficher (0 = canal combine CCCH+SDCCH4)
export W="${W:-240}"          # colonnes (multiframes d'historique)
export DISPLAY="${DISPLAY:-:0}"
PY="${PY:-/root/.env/bin/python3}"; command -v "$PY" >/dev/null 2>&1 || PY=python3

SNIFF=/opt/GSM/qosmo-grgsm/opt-gsm-scripts/gsm_sniff.py
docker exec "$CONTAINER" test -f "$SNIFF" || { echo "gsm_sniff absent dans $CONTAINER"; exit 1; }

# gsm_sniff (conteneur, passif) -> stdout -> python hote (rendu matrice live)
docker exec "$CONTAINER" sh -c "SNIFF_IFACE=lo stdbuf -oL python3 $SNIFF $MODE" \
| "$PY" - <<'PYEOF'
import sys, os, re, threading, collections
import numpy as np, matplotlib
matplotlib.use("TkAgg") if not os.environ.get("MATRIX_SAVE") else matplotlib.use("Agg")
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
from matplotlib.colors import ListedColormap, BoundaryNorm

MODE = os.environ.get("MODE","burst"); TS = int(os.environ.get("TS","0")); W = int(os.environ.get("W","240"))
DUMMY = "00011111011011101100000101001001"   # debut du dummy burst GSM (idle)

# --- type SI/L2 -> code couleur (mode si) ---
SI_CODE = {"?":1,"SI1":2,"SI2":3,"SI3":4,"SI4":5,"SI2bis":6,"SI2ter":7,"SI13":8,
           "PAGING-REQ-1":9,"PAGING-REQ-2":9,"PAGING-REQ-3":9,
           "IMM-ASSIGN":10,"IMM-ASSIGN-EXT":10,"IMM-ASSIGN-REJ":10}
# LAPDm (SDCCH/SACCH) detecte par octet de controle : on tag "DEDIE"
SI_LABELS = ["·","?","SI1","SI2","SI3","SI4","SI2b","SI2t","SI13","PAG","IMM","DEDIE"]
SI_COLORS = ["#101018","#888","#1f77b4","#2ca02c","#17becf","#9467bd","#8c564b","#e377c2",
```

```
            "#7f7f7f","#ff7f0e","#d62728","#ffd700"]

mat = np.zeros((51, W)); curmf=[None]; lock=threading.Lock()

re_ts  = re.compile(r"ts(\d)\b")
re_m51 = re.compile(r"%51=(\d+)")
re_fn  = re.compile(r"\bfn=(\d+)|\bFN=(\d+)")
re_148 = re.compile(r"\b([01]{148})\b")
re_si  = re.compile(r"\b(SI2bis|SI2ter|SI1[3]?|SI[2-4]|IMM-ASSIGN-?\w*|PAGING-REQ-\d)\b")
re_port= re.compile(r":(\d{4})\b")

def reader():
    for line in sys.stdin:
        m51 = re_m51.search(line);  mts = re_ts.search(line)
        if not m51: continue
        if mts and int(mts.group(1)) != TS:   # filtre timeslot (defaut ts0)
            # en mode si les lignes :4730 n'ont pas toujours ts -> on garde
            if MODE=="burst": continue
        fnm = re_fn.search(line); fn = int(fnm.group(1) or fnm.group(2)) if fnm else None
        r = int(m51.group(1)) % 51
        if MODE=="burst":
            mb = re_148.search(line)
            if "NOPE" in line or "vide" in line: v = 0.0          # burst vide
            elif mb and mb.group(1).startswith(DUMMY): v = 1.0    # dummy (idle)
            elif mb: v = 2.0                                       # reel (actif)
            else: v = 0.0
        else:
            ms = re_si.search(line)
            if ms: v = float(SI_CODE.get(ms.group(1), 1))
            elif "?" in line:                                     # LAPDm dedie (SDCCH/SACCH UA/I)
                v = 11.0 if (22<=r<=25 or 26<=r<=39 or 42<=r<=49) else 1.0
            else: v = 1.0
        mf = fn//51 if fn is not None else (curmf[0] or 0)
        with lock:
            if curmf[0] is None: curmf[0]=mf
            while mf > curmf[0]:                # nouvelle multiframe -> shift gauche
                mat[:, :-1] = mat[:, 1:]; mat[:, -1] = 0; curmf[0]+=1
            if mf == curmf[0]:
                mat[r, -1] = v

threading.Thread(target=reader, daemon=True).start()

plt.style.use("dark_background")
fig, ax = plt.subplots(figsize=(12,7))
title = ("BURSTS ts%d (·=vide  bleu=dummy/idle  rouge=ACTIF)" % TS) if MODE=="burst" \
        else "SI / L2 decodes (type par couleur)"
fig.suptitle("Matrice dynamique 51-multiframe - %s  [%s]" % (title, os.environ.get("CONTAINER","")),
             color="#7fd", fontsize=11)
if MODE=="burst":
    cmap = ListedColormap(["#101018","#1f77b4","#d62728"]); norm = BoundaryNorm([-.5,.5,1.5,2.5], cmap.N)
else:
    cmap = ListedColormap(SI_COLORS); norm = BoundaryNorm([i-.5 for i in range(len(SI_COLORS)+1)], cmap.N)
im = ax.imshow(mat, aspect="auto", origin="lower", cmap=cmap, norm=norm, interpolation="nearest")
ax.set_ylabel("FN % 51  (FCCH=0,10..  SCH=1,11..  BCCH=2-5  CCCH=6-19  SDCCH=22-39  SACCH=42-49)")
ax.set_xlabel("temps (multiframes 51) →")
# reperes des bandes canal (combine BCCH+SDCCH4 ts0)
for y in (0,10,20,30,40): ax.axhline(y, color="#444", lw=.4)
for y0,y1,lbl,c in [(2,5,"BCCH","#17becf"),(22,25,"SDCCH/0","#ffd700"),(42,45,"SACCH/0","#ff7f0e")]:
    ax.axhspan(y0-.5,y1+.5, xmin=0, xmax=0.015, color=c, alpha=.8)
if MODE=="si":
    from matplotlib.patches import Patch
    ax.legend(handles=[Patch(color=SI_COLORS[i], label=SI_LABELS[i]) for i in range(1,len(SI_LABELS))],
              loc="upper left", ncol=4, fontsize=7, framealpha=.3)

def update(_):
    with lock: im.set_data(mat.copy())
```

```
    return (im,)
ani = FuncAnimation(fig, update, interval=250, blit=False, cache_frame_data=False)
print("[matrix:%s] fenetre live - ferme pour quitter." % MODE, flush=True)
plt.show()
PYEOF
```

3.400 /opt/GSM/osmo-operator/tools/osmo-banc-cmd.sh

1966 octets, 39 lignes → 39 lignes

```
#!/bin/bash
# osmo-banc-cmd.sh - les alias et fonctions du banc (scripts/osmo-banc-aliases.sh) en VRAIES commandes.
#
# [2026-10-24] banc-rssi, banc-trx, banc-mobile... n existaient que dans un shell de connexion qui
# source /etc/profile.d/98-osmo-banc-aliases.sh : rien depuis Alt+F2, sudo, un script ou un shell
# qui ne lit pas profile.d. Ce script est lie sous chacun de ces noms dans /usr/local/bin ; il
# source le fichier d alias et rejoue son propre nom avec ses arguments. La definition reste UNE :
# scripts/osmo-banc-aliases.sh.
#
#   tools/osmo-banc-cmd.sh --install [DIR]   pose les liens (defaut /usr/local/bin ; DIR peut etre
#                                            un $ROOTFS/usr/local/bin) pour chaque banc-* et c123-*
#                                            du fichier (c123 lui-meme est un lien direct vers
#                                            tools/osmo-c123-gtk.py, sans --instance). Les liens
#                                            visent $OSMO_BANC_TREE/tools (defaut /opt/GSM/osmo-operator)
REPO=/opt/GSM/osmo-operator
SELF="$(readlink -f "${BASH_SOURCE[0]}")"
ALIASES="$(dirname "$SELF")/../scripts/osmo-banc-aliases.sh"
[ -f "$ALIASES" ] || ALIASES="$REPO/scripts/osmo-banc-aliases.sh"

noms() {
    sed -n -e 's/^alias \([a-z0-9-]*\)=.*/\1/p' -e 's/^function \([a-z0-9-]*\) .*/\1/p' "$ALIASES" \
        | grep -E '^(banc-|c123-)' | sort -u
}

if [ "${1:-}" = "--install" ]; then
    dest="${2:-/usr/local/bin}"
    install -d "$dest"
    for n in $(noms); do ln -sf "${OSMO_BANC_TREE:-$REPO}/tools/osmo-banc-cmd.sh" "$dest/$n"; done
    echo "commandes du banc -> $dest : $(noms | tr '\n' ' ')"
    exit 0
fi

nom="$(basename "$0")"
shopt -s expand_aliases
# shellcheck source=../scripts/osmo-banc-aliases.sh
. "$ALIASES"
type "$nom" >/dev/null 2>&1 || { echo "$nom : inconnu de $ALIASES" >&2; exit 127; }
# eval : l alias n est developpe qu a la lecture d une NOUVELLE ligne, apres le source
eval "$nom \"\$@\""
```

3.401 /opt/GSM/osmo-operator/tools/osmo-c123-gtk.py

27384 octets, 620 lignes → 620 lignes

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-
"""osmo-c123-gtk.py - le Motorola C123 emule, en photo : ecran LCD incruste, touches cliquables.

[2026-10-19] Le QEMU du depot (qosmo) n'a pas de backend gtk ; il publie l'ecran
ST7558 dans /dev/shm/calypso_lcd (hw/arm/calypso/calypso_st7558.c) et lit les
touches sur un socket Unix datagramme (/tmp/calypso_keys.sock,
calypso_armio.c). Cette fenetre GTK3 fait le reste :
```

```
  - la photo du telephone (data/c123.jpg ou .png ; sans photo, un schema) ;
    [2026-10-21] --modele c139 : data/c139.jpg + data/c139-zones.json et l'ecran
    couleur 96x64 publie par calypso_uwire.c (FreeCalypso Magnetite c139) ;
    auto (defaut) = d'apres le magic de l'image partagee ;
  - l'ecran 96x65 redessine a chaque changement, incruste dans la zone « lcd » ;
  - un clic sur une touche de la photo = press/release vers QEMU ; le clavier
    du PC marche aussi (0-9, * ou s, # ou h, fleches, Entree = menu,
    Entree pave ou o = ok, Echap ou p = power, F1/F2 = touches logicielles).
[2026-10-21] Plusieurs instances (c54x_exe/run.sh INSTANCE=N : ecran
/dev/shm/calypsoN/calypso_lcd, touches /tmp/calypso_keysN.sock). Sans
--instance, la fenetre choisit la premiere instance vivante qui N'EST PAS un
TRX de la BTS (firmware trx : /tmp/c54x-pontN/transceiver.log ou « BTS MODE »
dans osmocon.log) - le montage --bts --ms met le mobile en instance 2. La
touche power (rouge, Echap, p) eteint le firmware des l'appui (trx comme
layer1 : « Powering off due to keypress ») : elle n'est envoyee qu'apres un
APPUI LONG (%.1f s), et jamais a une instance TRX (le 21/10 a 08:18 un power
envoye a l'instance 1 a eteint le TRX et la BTS a perdu son horloge).
Zones : data/c123-zones.json (x, y, l, h normalises). `--calibrate` les
redefinit a la souris : pour chaque zone annoncee, cliquer le coin haut-gauche
puis le coin bas-droit ; `s` sauve, `q` quitte.

  /usr/bin/python3 tools/osmo-c123-gtk.py [--image PHOTO] [--zones JSON]
        [--instance N] [--lcd /dev/shm/calypso_lcd] [--keys /tmp/calypso_keys.sock]
        [--scale 2.5] [--calibrate] [--power-libre] [--toutes [1,2,3]]
(python3 systeme : celui de /root/.env n'a pas PyGObject.)
"""
import argparse
import json
import os
import socket
import struct
import sys
import time

try:
    import gi
except ModuleNotFoundError:
    # [2026-10-21] lance avec le python du venv (/root/.env : pas de PyGObject) : on se relance
    # avec le python systeme, qui a gi ; « pip install gi » n'est pas le bon paquet.
    if os.path.realpath(sys.executable) != os.path.realpath("/usr/bin/python3") and
     ↪  os.path.exists("/usr/bin/python3"):
        os.execv("/usr/bin/python3", ["/usr/bin/python3", os.path.abspath(__file__)] + sys.argv[1:])
    sys.stderr.write("PyGObject (gi) introuvable : apt install python3-gi gir1.2-gtk-3.0, et lancer avec
 ↪  /usr/bin/python3\n")
    sys.exit(1)
gi.require_version("Gtk", "3.0")
gi.require_version("GdkPixbuf", "2.0")
from gi.repository import Gtk, Gdk, GLib, GdkPixbuf  # noqa: E402
import cairo  # noqa: E402

HERE = os.path.dirname(os.path.realpath(__file__))
DATA = os.path.join(HERE, "..", "data")
LCD_W, LCD_H, LCD_PAGES, LCD_HDR = 96, 65, 9, 32
POWER_APPUI_LONG_S = 1.2       # duree d'appui sur power avant de l'envoyer au firmware
__doc__ = __doc__ % POWER_APPUI_LONG_S
ORDRE_ZONES = ["lcd", "lsb", "rsb", "up", "left", "menu", "right", "down", "ok", "power",
               "1", "2", "3", "4", "5", "6", "7", "8", "9", "star", "0", "hash"]
LIBELLES = {"lsb": "—", "rsb": "—", "up": "▲", "down": "▼", "left": "◀", "right": "▶",
            "menu": "●", "ok": "☎", "power": "⏻", "star": "*", "hash": "#"}
CLAVIER = {  # nom de touche GDK -> touche du C123
    "0": "0", "1": "1", "2": "2", "3": "3", "4": "4", "5": "5", "6": "6", "7": "7", "8": "8", "9": "9",
    "KP_0": "0", "KP_1": "1", "KP_2": "2", "KP_3": "3", "KP_4": "4", "KP_5": "5", "KP_6": "6",
    "KP_7": "7", "KP_8": "8", "KP_9": "9", "asterisk": "star", "KP_Multiply": "star", "s": "star",
    "numbersign": "hash", "h": "hash", "Up": "up", "Down": "down", "Left": "left", "Right": "right",
    "Return": "menu", "KP_Enter": "ok", "o": "ok", "Escape": "power", "p": "power",
    "F1": "lsb", "F2": "rsb",
```

```
}


def rundir_instance(n):
    return "/tmp/c54x-pont" if n == 1 else "/tmp/c54x-pont%d" % n


def chemins_instance(n):
    """(ecran, socket des touches) de l'instance n (c54x_exe/run.sh : INSTANCE=1 sans suffixe)."""
    if n == 1:
        return "/dev/shm/calypso_lcd", "/tmp/calypso_keys.sock"
    return "/dev/shm/calypso%d/calypso_lcd" % n, "/tmp/calypso_keys%d.sock" % n


def instance_vivante(n):
    try:
        with open(os.path.join(rundir_instance(n), "dsp.pid")) as f:
            pid = int(f.read().strip() or 0)
        os.kill(pid, 0)
        return True
    except (OSError, ValueError):
        return False


def instance_est_trx(n):
    """Vrai si l'instance n fait tourner le firmware trx (radio de la BTS) : un power l'eteint et la BTS
    perd son horloge. Indices : transceiver.log dans son RUNDIR (run.sh OSMO_TRX=1), ou « BTS MODE » sur
    sa console (trx.highram imprime BTS MODE: ... des le L1CTL_RESET)."""
    rd = rundir_instance(n)
    if os.path.isfile(os.path.join(rd, "transceiver.log")):
        return True
    try:
        with open(os.path.join(rd, "osmocon.log"), "rb") as f:
            return b"BTS MODE:" in f.read(1 << 20)
    except OSError:
        return False


def instance_par_defaut():
    """La premiere instance vivante qui n'est pas un TRX (le mobile du montage --bts --ms est la 2) ;
    a defaut la premiere vivante ; a defaut 1."""
    vivantes = [n for n in range(1, 9) if instance_vivante(n)]
    for n in vivantes:
        if not instance_est_trx(n):
            return n
    return vivantes[0] if vivantes else 1


def magic_lcd(path):
    """Le magic de l'image partagee : « C123LCD » (ST7558 mono) ou « C139LCD » (couleur), None si absente."""
    try:
        with open(path, "rb") as f:
            m = f.read(7)
    except OSError:
        return None
    return m.decode("ascii", "replace") if m in (b"C123LCD", b"C139LCD") else None


class Lcd:
    """L'image partagee : en-tete de 32 octets (magic, largeur, hauteur, pages ou octets/point, drapeaux,
    compteur) puis les points. Deux magics : « C123LCD » (calypso_st7558.c : 96 x 9 pages, 1 bit par point,
    bit 0 en haut) et « C139LCD » (calypso_uwire.c, Magnetite c139 : 96x64 RGB565 petit-boutiste)."""

    def __init__(self, path):
        self.path = path
        self.seq = -1
```

```
        self.flags = 0
        self.pages = b""
        self.w, self.h, self.modele = LCD_W, LCD_H, "c123"
        self.surface = cairo.ImageSurface(cairo.FORMAT_RGB24, self.w, self.h)
        self.vivant = False
        self._peindre()

    def lire(self):
        try:
            with open(self.path, "rb") as f:
                buf = f.read(LCD_HDR + 96 * 64 * 2)
        except OSError:
            if self.vivant:
                self.vivant = False
                return True
            return False
        if len(buf) < LCD_HDR + 8 or buf[:7] not in (b"C123LCD", b"C139LCD"):
            return False
        w, h, pages, flags, _pad, seq = struct.unpack_from("<HHHBBI", buf, 8)
        modele = "c139" if buf[:7] == b"C139LCD" else "c123"
        taille = w * h * pages if modele == "c139" else w * pages
        if not (1 <= w <= 256 and 1 <= h <= 256) or len(buf) < LCD_HDR + taille:
            return False
        if seq == self.seq and self.vivant:
            return False
        if (w, h, modele) != (self.w, self.h, self.modele):
            self.w, self.h, self.modele = w, h, modele
            self.surface = cairo.ImageSurface(cairo.FORMAT_RGB24, w, h)
        self.seq, self.flags, self.pages, self.vivant = seq, flags, buf[LCD_HDR:LCD_HDR + taille], True
        self._peindre()
        return True

    def _peindre(self):
        if self.modele == "c139":
            self._peindre_c139()
            return
        inverse, eteint = bool(self.flags & 1), bool(self.flags & 2)
        on, off = (0x1a2a1a, 0xc8d8b0) if not inverse else (0xc8d8b0, 0x1a2a1a)
        if eteint or not self.vivant:
            on = off = 0x9aa892
        data = self.surface.get_data()
        stride = self.surface.get_stride()
        pages = self.pages
        for y in range(self.h):
            page, mask = y >> 3, 1 << (y & 7)
            base = page * self.w
            for x in range(self.w):
                bit = pages[base + x] & mask if pages else 0
                c = on if bit else off
                o = y * stride + x * 4
                data[o:o + 4] = struct.pack("<I", c)
        self.surface.mark_dirty()

    def _peindre_c139(self):
        """RGB565 -> RGB24 ; sans QEMU : gris uniforme."""
        data = self.surface.get_data()
        stride = self.surface.get_stride()
        src = self.pages
        for y in range(self.h):
            for x in range(self.w):
                if src and self.vivant:
                    i = (y * self.w + x) * 2
                    p = src[i] | (src[i + 1] << 8)
                    r, g, b = (p >> 11) & 0x1f, (p >> 5) & 0x3f, p & 0x1f
                    c = ((r << 3 | r >> 2) << 16) | ((g << 2 | g >> 4) << 8) | (b << 3 | b >> 2)
                else:
```

```
                    c = 0x9aa892
                o = y * stride + x * 4
                data[o:o + 4] = struct.pack("<I", c)
        self.surface.mark_dirty()


class Clavier:
    """Le socket datagramme de calypso_armio.c : « press X », « release X », « tap X ms »."""

    def __init__(self, path):
        self.path = path
        self.sock = socket.socket(socket.AF_UNIX, socket.SOCK_DGRAM)
        self.erreur = None

    def envoyer(self, cmd, touche):
        try:
            self.sock.sendto(("%s %s\n" % (cmd, touche)).encode(), self.path)
            self.erreur = None
            return True
        except OSError as e:
            self.erreur = str(e)
            return False


FENETRES = []


def fenetre_fermee(_w):
    """Une fenetre fermee ; Gtk.main s'arrete avec la derniere."""
    if _w in FENETRES:
        FENETRES.remove(_w)
    if not FENETRES:
        Gtk.main_quit()


class Fenetre(Gtk.Window):
    def __init__(self, args):
        # [2026-10-21] --modele c139 (FreeCalypso Magnetite) : photo data/c139.jpg, zones data/c139-zones.json,
        # ecran couleur « C139LCD » ; auto = d'apres le magic de l'image partagee, c123 a defaut.
        self.modele = args.modele
        if self.modele == "auto":
            self.modele = "c139" if magic_lcd(args.lcd) == "C139LCD" else "c123"
        if not args.zones:
            args.zones = os.path.join(DATA, "%s-zones.json" % self.modele)
        self.modele_force = args.modele != "auto"
        self.args = args
        super().__init__(title=self._titre())
        self.zones = self._charger_zones(args.zones)
        self.pixbuf = self._charger_image(args.image)
        self.lcd = Lcd(args.lcd)
        self.clavier = Clavier(args.keys)
        self.enfoncees = set()
        # un C139 Magnetite n'est jamais un TRX de la BTS (les traces trx d'un ancien run ne comptent pas)
        self.trx = instance_est_trx(args.instance) if args.instance and self.modele != "c139" else False
        self.power_depuis = None      # instant de l'appui sur power (appui long en cours)
        self.power_envoyee = False    # press power deja transmis au firmware
        self.calibrage = [] if args.calibrate else None   # [(zone, x1, y1)] en cours
        self.cal_index = 0
        self.cal_premier = None

        self.lw, self.lh = self._taille()

        boite = Gtk.Box(orientation=Gtk.Orientation.VERTICAL)
        self.add(boite)
        self.zone = Gtk.DrawingArea()
        self.zone.set_size_request(self.lw, self.lh)
```

```
    self.zone.add_events(Gdk.EventMask.BUTTON_PRESS_MASK | Gdk.EventMask.BUTTON_RELEASE_MASK
                         | Gdk.EventMask.POINTER_MOTION_MASK)
    self.zone.connect("draw", self.dessiner)
    self.zone.connect("button-press-event", self.clic)
    self.zone.connect("button-release-event", self.relacher)
    boite.pack_start(self.zone, True, True, 0)
    self.statut = Gtk.Label(xalign=0)
    self.statut.set_margin_start(6)
    boite.pack_start(self.statut, False, False, 2)
    self.connect("key-press-event", self.touche_pc)
    self.connect("key-release-event", self.touche_pc)
    self.connect("destroy", fenetre_fermee)
    GLib.timeout_add(40, self.tic)
    self._statut()

# --- chargements -------------------------------------------------------
def _titre(self):
    role = "TRX de la BTS" if (self.args.instance and instance_est_trx(self.args.instance)) else "mobile"
    if self.modele == "c139":
        role = "mobile Magnetite"
    titre = "Motorola %s - Calypso emule" % self.modele.upper()
    if self.args.instance:
        titre += " - instance %d (%s)" % (self.args.instance, role)
    return titre

def _taille(self):
    if self.pixbuf:
        iw, ih = self.pixbuf.get_width(), self.pixbuf.get_height()
    else:
        iw, ih = 300, 630
    scale = self.args.scale or min(2.5, 900.0 / ih)
    return int(iw * scale), int(ih * scale)

def _changer_modele(self, modele):
    """[2026-10-21] l'ecran partage change de magic (ex. --bts --ms-magnetite : l'instance 2
    devient un C139 Magnetite apres l'ouverture de la fenetre) : photo, zones, taille et titre suivent."""
    self.modele = modele
    self.args.zones = os.path.join(DATA, "%s-zones.json" % modele)
    self.zones = self._charger_zones(self.args.zones)
    self.pixbuf = self._charger_image(None)
    self.lw, self.lh = self._taille()
    self.zone.set_size_request(self.lw, self.lh)
    self.resize(self.lw, self.lh + 24)
    self.set_title(self._titre())
    self._statut("modele %s (ecran %dx%d)" % (modele, self.lcd.w, self.lcd.h))

def _charger_zones(self, path):
    with open(path) as f:
        z = json.load(f)
    return {k: v for k, v in z.items() if not k.startswith("_")}

def _charger_image(self, path):
   candidats = [path] if path else [os.path.join(DATA, self.modele + ext) for ext in (".jpg", ".png", ".jpeg")]
    for p in candidats:
        if p and os.path.isfile(p):
            try:
                return GdkPixbuf.Pixbuf.new_from_file(p)
            except GLib.Error as e:
                print("photo illisible %s : %s" % (p, e), file=sys.stderr)
    print("pas de photo (%s) : schema dessine a la place ; posez la photo du %s dans data/%s.jpg"
          % (", ".join(c for c in candidats if c), self.modele.upper(), self.modele), file=sys.stderr)
    return None

# --- geometrie ---------------------------------------------------------
def rect(self, nom):
    x, y, w, h = self.zones[nom]
```

```
    return x * self.lw, y * self.lh, w * self.lw, h * self.lh

def zone_sous(self, px, py):
    for nom in self.zones:
        if nom == "lcd":
            continue
        x, y, w, h = self.rect(nom)
        if x <= px <= x + w and y <= py <= y + h:
            return nom
    return None

# --- dessin ------------------------------------------------------------
def dessiner(self, _w, cr):
    if self.pixbuf:
        cr.save()
        cr.scale(self.lw / self.pixbuf.get_width(), self.lh / self.pixbuf.get_height())
        Gdk.cairo_set_source_pixbuf(cr, self.pixbuf, 0, 0)
        cr.paint()
        cr.restore()
    else:
        self._schema(cr)
    # l'ecran
    x, y, w, h = self.rect("lcd")
    cr.save()
    cr.translate(x, y)
    cr.scale(w / self.lcd.w, h / self.lcd.h)
    cr.set_source_surface(self.lcd.surface, 0, 0)
    cr.get_source().set_filter(cairo.FILTER_NEAREST)
    cr.paint()
    cr.restore()
    if not self.lcd.vivant:
        cr.set_source_rgba(0, 0, 0, 0.55)
        cr.rectangle(x, y, w, h)
        cr.fill()
        cr.set_source_rgb(1, 1, 1)
        cr.set_font_size(max(10, h / 6))
        cr.move_to(x + 6, y + h / 2)
        cr.show_text("QEMU absent")
    # touches enfoncees
    for nom in self.enfoncees:
        if nom in self.zones:
            x, y, w, h = self.rect(nom)
            cr.set_source_rgba(1.0, 0.85, 0.1, 0.45)
            cr.rectangle(x, y, w, h)
            cr.fill()
    # calibrage : zones deja posees + zone en cours
    if self.calibrage is not None:
        cr.set_line_width(1.5)
        for nom in self.zones:
            x, y, w, h = self.rect(nom)
            cr.set_source_rgba(0.1, 0.6, 1.0, 0.8)
            cr.rectangle(x, y, w, h)
            cr.stroke()
            cr.move_to(x + 2, y + h - 2)
            cr.set_font_size(10)
            cr.show_text(nom)
        if self.cal_premier:
            cr.set_source_rgba(1, 0, 0, 0.9)
            cr.arc(self.cal_premier[0], self.cal_premier[1], 4, 0, 6.3)
            cr.fill()
    return False

def _schema(self, cr):
    cr.set_source_rgb(0.82, 0.82, 0.84)
    cr.rectangle(0, 0, self.lw, self.lh)
    cr.fill()
```

```
    cr.set_source_rgb(0.25, 0.25, 0.28)
    cr.rectangle(self.lw * 0.04, self.lh * 0.02, self.lw * 0.92, self.lh * 0.96)
    cr.fill()
    for nom in self.zones:
        if nom == "lcd":
            continue
        x, y, w, h = self.rect(nom)
        cr.set_source_rgb(0.85, 0.85, 0.87)
        cr.rectangle(x, y, w, h)
        cr.fill()
        cr.set_source_rgb(0.1, 0.1, 0.1)
        cr.set_font_size(max(9, h * 0.6))
        cr.move_to(x + w * 0.3, y + h * 0.75)
        cr.show_text(LIBELLES.get(nom, nom))

# --- entrees -----------------------------------------------------------
def clic(self, _w, ev):
    if self.calibrage is not None:
        return self._clic_calibrage(ev)
    nom = self.zone_sous(ev.x, ev.y)
    if nom:
        self.appuyer(nom)
    return True

def relacher(self, _w, ev):
    if self.calibrage is not None:
        return True
    for nom in list(self.enfoncees):
        self.lacher(nom)
    self.zone.queue_draw()
    return True

# --- touches : press/release vers QEMU, power sous appui long ---------
def appuyer(self, nom):
    if nom in self.enfoncees:
        return
    self.enfoncees.add(nom)
    if nom == "power" and not self.args.power_libre:
        self.power_depuis = time.monotonic()
        self.power_envoyee = False
        if self.trx:
            self._statut("power ignoree : l'instance %s est le TRX de la BTS (un power l'eteint et la BTS perd
             ↪  son horloge)"
                         % self.args.instance)
        else:
            self._statut("power : maintenir %.1f s pour eteindre le firmware" % POWER_APPUI_LONG_S)
    else:
        self.clavier.envoyer("press", nom)
        self._statut("touche %s" % nom)
    self.zone.queue_draw()

def lacher(self, nom):
    if nom not in self.enfoncees:
        return
    self.enfoncees.discard(nom)
    if nom == "power" and not self.args.power_libre:
        if self.power_envoyee:
            self.clavier.envoyer("release", nom)
        elif not self.trx:
            self._statut("power relachee avant %.1f s : rien envoye" % POWER_APPUI_LONG_S)
        self.power_depuis = None
        self.power_envoyee = False
    else:
        self.clavier.envoyer("release", nom)
    self.zone.queue_draw()
```

```
def touche_pc(self, _w, ev):
    nom_gdk = Gdk.keyval_name(ev.keyval)
    if self.calibrage is not None:
        if ev.type == Gdk.EventType.KEY_PRESS and nom_gdk == "s":
            self._sauver_zones()
        elif ev.type == Gdk.EventType.KEY_PRESS and nom_gdk == "q":
            Gtk.main_quit()
        return True
    nom = CLAVIER.get(nom_gdk)
    if not nom:
        return False
    if ev.type == Gdk.EventType.KEY_PRESS:
        self.appuyer(nom)
    else:
        self.lacher(nom)
    return True

# --- calibrage ----------------------------------------------------------
def _clic_calibrage(self, ev):
    if self.cal_index >= len(ORDRE_ZONES):
        self._statut("calibrage termine : s = sauver, q = quitter")
        return True
    nom = ORDRE_ZONES[self.cal_index]
    if self.cal_premier is None:
        self.cal_premier = (ev.x, ev.y)
        self._statut("%s : coin bas-droit" % nom)
    else:
        x1, y1 = self.cal_premier
        x2, y2 = ev.x, ev.y
        self.zones[nom] = [round(min(x1, x2) / self.lw, 4), round(min(y1, y2) / self.lh, 4),
                           round(abs(x2 - x1) / self.lw, 4), round(abs(y2 - y1) / self.lh, 4)]
        self.cal_premier = None
        self.cal_index += 1
        if self.cal_index < len(ORDRE_ZONES):
            self._statut("%s : coin haut-gauche" % ORDRE_ZONES[self.cal_index])
        else:
            self._statut("calibrage termine : s = sauver, q = quitter")
    self.zone.queue_draw()
    return True

def _sauver_zones(self):
    z = {"_doc": "Zones cliquables de la photo du C123, [x, y, largeur, hauteur] normalises ; "
                 "recalibrer : osmo-c123-gtk.py --modele %s --calibrate" % self.modele}
    z.update({k: self.zones[k] for k in ORDRE_ZONES if k in self.zones})
    with open(self.args.zones, "w") as f:
        json.dump(z, f, indent=2)
    self._statut("zones sauvees dans %s" % self.args.zones)

# --- boucle -------------------------------------------------------------
def tic(self):
    if self.lcd.vivant and not self.modele_force and self.lcd.modele != self.modele:
        self._changer_modele(self.lcd.modele)
    if self.lcd.lire():
        self.zone.queue_draw()
        self._statut()
    if (self.power_depuis is not None and not self.power_envoyee and not self.trx
            and time.monotonic() - self.power_depuis >= POWER_APPUI_LONG_S):
        self.power_envoyee = True
        self.clavier.envoyer("press", "power")
        self._statut("power envoyee a l'instance %s (appui long)" % self.args.instance)
    return True

def _statut(self, msg=None):
    if self.calibrage is not None and msg is None:
        msg = "calibrage : %s : coin haut-gauche" % ORDRE_ZONES[0]
    etat = "LCD %s (v%d)" % ("vivant" if self.lcd.vivant else "absent", max(self.lcd.seq, 0))
```

```
        clav = "touches -> %s%s" % (self.args.keys, " (%s)" % self.clavier.erreur if self.clavier.erreur else "")
        inst = "instance %s%s · " % (self.args.instance, " (TRX de la BTS)" if self.trx else "") if
↪   self.args.instance else ""
        self.statut.set_text("%s%s · %s%s" % (inst, etat, clav, " · " + msg if msg else ""))


def main():
    ap = argparse.ArgumentParser(description=__doc__, formatter_class=argparse.RawDescriptionHelpFormatter)
    ap.add_argument("--modele", choices=["auto", "c123", "c139"], default="auto",
                    help="telephone : c123 (ST7558 mono, osmocom-bb) ou c139 (ecran couleur, FreeCalypso Magnetite)
                      ↪  ; "
                         "auto = d'apres l'ecran partage (defaut)")
    ap.add_argument("--image", help="photo du telephone (defaut data/<modele>.jpg|png)")
    ap.add_argument("--zones", default=None, help="zones cliquables (defaut data/<modele>-zones.json)")
    ap.add_argument("--instance", type=int, default=0,
                    help="instance c54x_exe (1 = /dev/shm/calypso_lcd, N = /dev/shm/calypsoN/... et
                     ↪  /tmp/calypso_keysN.sock) ; "
                         "defaut : la premiere instance vivante qui n'est pas un TRX de la BTS")
    ap.add_argument("--lcd", default=os.environ.get("CALYPSO_LCD_SHM"), help="ecran partage (prime sur
↪   --instance)")
    ap.add_argument("--keys", default=os.environ.get("CALYPSO_KEYS_SOCK"), help="socket des touches (prime sur
↪   --instance)")
    ap.add_argument("--power-libre", action="store_true",
                    help="touche power envoyee des l'appui, meme a un TRX (defaut : appui long de %.1f s, jamais a
                     ↪  un TRX)"
                         % POWER_APPUI_LONG_S)
    ap.add_argument("--toutes", nargs="?", const="1,2,3,4", metavar="N,M,...",
                    help="une fenetre par instance (defaut 1,2,3,4 : TRX TS0, mobile, TRX TS1, rssi ; plus toute
                     ↪  autre instance vivante, et celles qui "
                         "apparaissent ensuite : guet toutes les 2 s), cote a cote")
    ap.add_argument("--scale", type=float, default=0, help="facteur d'agrandissement de la photo (defaut : hauteur
↪   900 px max)")
    ap.add_argument("--calibrate", action="store_true", help="redefinir les zones a la souris")
    ap.add_argument("--autotest", action="store_true", help="sans fenetre : lit l'image LCD une fois et sort")
    args = ap.parse_args()
    if args.toutes:
        if args.calibrate or args.autotest:
            ap.error("--toutes ne va pas avec --calibrate / --autotest")
        voulues = [int(x) for x in args.toutes.replace(" ", ",").split(",") if x]
        for n in range(1, 9):
            if n not in voulues and instance_vivante(n):
                voulues.append(n)
        ouvertes = {}
        pos = [0]

        def ouvrir(n):
            a = argparse.Namespace(**vars(args))
            a.instance, a.lcd, a.keys = n, None, None
            a.lcd, a.keys = chemins_instance(n)
            if not args.scale:
                a.scale = 1.1                    # quatre telephones de front sur un ecran 1080p
            f = Fenetre(a)
            FENETRES.append(f)
            ouvertes[n] = f
            f.move(pos[0], 0)
            pos[0] += f.lw + 24
            f.show_all()

        for n in voulues:
            ouvrir(n)

        # [2026-10-21] une instance lancee APRES la fenetre (banc-rssi, INSTANCE=4...) avait pas de fenetre :
        # on guette les instances 1..8 toutes les 2 s et on ouvre celles qui apparaissent.
        def guetter():
            for n in range(1, 9):
                if n not in ouvertes and instance_vivante(n):
```

```
                    print("instance %d apparue : fenetre ouverte" % n, file=sys.stderr)
                    ouvrir(n)
            return True
        GLib.timeout_add(2000, guetter)
        Gtk.main()
        return 0
    if not args.instance and not (args.lcd and args.keys):
        args.instance = instance_par_defaut()
    if args.instance:
        lcd, keys = chemins_instance(args.instance)
        args.lcd = args.lcd or lcd
        args.keys = args.keys or keys
        if instance_est_trx(args.instance):
            print("instance %d : firmware trx (TRX de la BTS) ; touche power refusee sans --power-libre" %
             ↪  args.instance,
                  file=sys.stderr)
    if args.autotest:
        lcd = Lcd(args.lcd)
        ok = lcd.lire()
        print("lcd %s : vivant=%s seq=%d flags=%d" % (args.lcd, lcd.vivant, lcd.seq, lcd.flags))
        return 0 if ok else 1
    f = Fenetre(args)
    FENETRES.append(f)
    f.show_all()
    Gtk.main()
    return 0


if __name__ == "__main__":
    sys.exit(main())
```

3.402 /opt/GSM/osmo-operator/tools/osmo-dino-ciel.py

24547 octets, 516 lignes → 516 lignes

```
#!/usr/bin/env python3
# osmo-dino-ciel.py - le ciel du dino, aux vraies valeurs de la NASA.
#
# Le modele : un dino court sur l'EQUATEUR, plein est, vers le soleil levant,
# a une date AVANT J.-C., et le temps est ACCELERE (par defaut une heure de
# ciel par seconde, comme dans le jeu, dont la journee dure 24 s). Le Soleil et
# la Lune sont ceux du JPL (NASA) - service Horizons, ephemeride DE441 - vus
# depuis la position courante du dino, qui gagne des degres de longitude en
# courant. Sur l'equateur les astres montent a la verticale ; l'INCLINAISON
# apparait dans l'endroit ou ils se levent : le Soleil a `declinaison` degres a
# gauche (nord) ou a droite (sud) de la route, la Lune jusqu'a 28,6 degres
# (23,4 d'obliquite + 5,1 d'inclinaison de son orbite), et sa latitude
# ecliptique dit de combien elle s'ecarte du chemin du Soleil.
#
# Sources :
#   horizons  https://ssd.jpl.nasa.gov/api/horizons.api  (par defaut ; requiert
#             le reseau ; la Lune y existe de 9999 av. J.-C. a 9999 ap.)
#   meeus     formules de J. Meeus, Astronomical Algorithms (Soleil ch. 25,
#             Lune ch. 47 tronquee, ~0,3 degre) - repli hors ligne, sans NASA.
#
# Ce que le script fait lui-meme : la position du dino (longitude qui avance),
# le temps sideral, le passage geocentrique -> topocentrique (parallaxe de la
# Lune, jusqu'a 1 degre), azimut/hauteur, le repere du dino (devant/gauche/
# droite/derriere), l'angle du limbe eclaire de la Lune, le calendrier julien
# pour les dates anciennes (comme Horizons avant le 15 octobre 1582).
#
# Exemples :
#   osmo-dino-ciel.py                              # 9999 av. J.-C., 1 h/s
#   osmo-dino-ciel.py --date "BC 3000-06-21 04:00" --accel 600
```

```
#   osmo-dino-ciel.py --une-fois --json              # un instant, en JSON
#   osmo-dino-ciel.py --json --tick 0.5 > ciel.ndjson
#   osmo-dino-ciel.py --source meeus                 # sans reseau
import argparse
import json
import math
import os
import re
import sys
import threading
import time
import urllib.parse
import urllib.request

HORIZONS = "https://ssd.jpl.nasa.gov/api/horizons.api"
AU_KM = 149597870.7
R_TERRE_KM = 6378.137
DEG = math.pi / 180
J2000 = 2451545.0
JD_GREGORIEN = 2299160.5          # 1582-10-15 00:00 : avant, calendrier julien
JD_MIN_HORIZONS = -1931446.5      # BC 9999-Mar-16 00:00 UT (la Lune n'existe pas avant)
MOIS = ["janvier", "fevrier", "mars", "avril", "mai", "juin", "juillet",
        "aout", "septembre", "octobre", "novembre", "decembre"]


# ------------------------------------------------------------------ calendrier
def cal_vers_jd(y, m, d, frac=0.0, julien=None):
    """Calendrier -> jour julien. y = annee astronomique (0 = 1 av. J.-C.).
    Julien avant le 15/10/1582 comme Horizons, gregorien apres (ou force)."""
    a = (14 - m) // 12
    yy = y + 4800 - a
    mm = m + 12 * a - 3
    jdn_j = d + (153 * mm + 2) // 5 + 365 * yy + yy // 4 - 32083
    jdn_g = d + (153 * mm + 2) // 5 + 365 * yy + yy // 4 - yy // 100 + yy // 400 - 32045
    if julien is None:
        julien = (jdn_g - 0.5) < JD_GREGORIEN
    return (jdn_j if julien else jdn_g) - 0.5 + frac


def jd_vers_cal(jd):
    """Jour julien -> (annee astronomique, mois, jour, fraction de jour)."""
    z = math.floor(jd + 0.5)
    f = jd + 0.5 - z
    if jd < JD_GREGORIEN:                       # julien (division entiere = plancher)
        c = z + 32082
        d = (4 * c + 3) // 1461
        e = c - (1461 * d) // 4
        m = (5 * e + 2) // 153
        day = e - (153 * m + 2) // 5 + 1
        month = m + 3 - 12 * (m // 10)
        year = d - 4800 + m // 10
    else:
        a = z + 32044
        b = (4 * a + 3) // 146097
        c = a - (146097 * b) // 4
        d = (4 * c + 3) // 1461
        e = c - (1461 * d) // 4
        m = (5 * e + 2) // 153
        day = e - (153 * m + 2) // 5 + 1
        month = m + 3 - 12 * (m // 10)
        year = 100 * b + d - 4800 + m // 10
    return year, month, day, f


def date_fr(jd):
    y, m, d, f = jd_vers_cal(jd)
```

```
    h = f * 24
    an = f"an {1 - y} av. J.-C." if y <= 0 else f"an {y}"
    return f"{d} {MOIS[m - 1]} de l'{an}", f"{int(h):02d} h {int(h % 1 * 60):02d}"


def parse_date(s):
    """'BC 3000-06-21 04:00', '-2999-06-21', '3000 av. J.-C. 06-21', 'JD 625844.5',
    '2026-09-08T06:00'. Les annees BC : 'BC n' ou 'av' = annee 1-n."""
    s = s.strip()
    m = re.match(r"^JD\s*([-+]?\d+(?:\.\d*)?)$", s, re.I)
    if m:
        return float(m.group(1))
    bc = False
    m = re.match(r"^(?:BC|B\.C\.|av(?:\.| J\.?-?C\.?|ant)?(?: J\.?-?C\.?)?)\s+(.*)$", s, re.I)
    if m:
        bc, s = True, m.group(1)
    m = re.match(r"^(?:(\d+)\s+av\.?\s*J\.?-?C\.?\s+)?([-+]?\d+)?-?(\d{1,2})-(\d{1,2})"
                 r"(?:[ T](\d{1,2})(?::(\d{2}))?(?::(\d{2}))?)?$", s)
    if not m:
        raise ValueError(f"date illisible : {s!r}")
    if m.group(1):
        bc, y = True, int(m.group(1))
    else:
        y = int(m.group(2))
    if bc:
        y = 1 - abs(y)
    mo, d = int(m.group(3)), int(m.group(4))
    frac = (int(m.group(5) or 0) + int(m.group(6) or 0) / 60 + int(m.group(7) or 0) / 3600) / 24
    return cal_vers_jd(y, mo, d, frac)


def horizons_date(jd):
    return f"JD {jd:.8f}"


# ---------------------------------------------------------------- geometrie
def gmst_deg(jd_ut):
    """Temps sideral moyen de Greenwich (Meeus 12.4), degres."""
    t = (jd_ut - J2000) / 36525
    g = 280.46061837 + 360.98564736629 * (jd_ut - J2000) + 0.000387933 * t * t - t ** 3 / 38710000
    return g % 360


def radec_vec(ra, dec, r=1.0):
    return (r * math.cos(dec * DEG) * math.cos(ra * DEG),
            r * math.cos(dec * DEG) * math.sin(ra * DEG),
            r * math.sin(dec * DEG))


def norm(v):
    l = math.sqrt(v[0] ** 2 + v[1] ** 2 + v[2] ** 2) or 1.0
    return (v[0] / l, v[1] / l, v[2] / l)


def dot(a, b):
    return a[0] * b[0] + a[1] * b[1] + a[2] * b[2]


def cross(a, b):
    return (a[1] * b[2] - a[2] * b[1], a[2] * b[0] - a[0] * b[2], a[0] * b[1] - a[1] * b[0])


def wrap180(a):
    return (a + 180) % 360 - 180
```

```
def topocentrique(ra, dec, dist_km, jd_ut, lon_deg):
    """Astre geocentrique (RA/DEC de la date, distance) vu depuis l'equateur a
    la longitude lon : direction unitaire dans le repere equatorial, plus les
    axes est / nord / zenith de l'observateur, dans ce meme repere."""
    theta = (gmst_deg(jd_ut) + lon_deg) * DEG           # temps sideral local
    obs = (R_TERRE_KM * math.cos(theta), R_TERRE_KM * math.sin(theta), 0.0)
    geo = radec_vec(ra, dec, dist_km)
    d = norm((geo[0] - obs[0], geo[1] - obs[1], geo[2] - obs[2]))
    est = (-math.sin(theta), math.cos(theta), 0.0)
    nord = (0.0, 0.0, 1.0)                               # sur l'equateur, le pole est a l'horizon
    zen = (math.cos(theta), math.sin(theta), 0.0)
    return d, est, nord, zen


def az_el(d, est, nord, zen):
    az = math.degrees(math.atan2(dot(d, est), dot(d, nord))) % 360
    el = math.degrees(math.asin(max(-1, min(1, dot(d, zen)))))
    return az, el


# ---------------------------------------------------------------- Horizons
class Horizons:
    """Ephemeride geocentrique apparente du JPL, par lots, avec prelecture.
    Chaque ligne : jd, ra, dec, illum (0-1), delta (km), elong (deg), waxing,
    ecl_lon, ecl_lat."""
    NOM = "JPL Horizons (NASA, DE441)"

    def __init__(self, pas_min, lot=600):
        self.pas = max(1, int(round(pas_min)))
        self.lot = lot
        self.buf = {10: [], 301: []}
        self.fin = None
        self.lock = threading.Lock()
        self.prefetch = None

    def _requete(self, corps, jd0, jd1):
        q = {
            "format": "text", "COMMAND": f"'{corps}'", "OBJ_DATA": "'NO'",
            "MAKE_EPHEM": "'YES'", "EPHEM_TYPE": "'OBSERVER'", "CENTER": "'500@399'",
            "START_TIME": f"'{horizons_date(jd0)}'", "STOP_TIME": f"'{horizons_date(jd1)}'",
            "STEP_SIZE": f"'{self.pas}m'", "QUANTITIES": "'2,10,20,23,31'",
            "CAL_FORMAT": "'JD'", "ANG_FORMAT": "'DEG'", "CSV_FORMAT": "'YES'",
            "EXTRA_PREC": "'YES'",
        }
        url = HORIZONS + "?" + urllib.parse.urlencode(q, quote_via=urllib.parse.quote)
        with urllib.request.urlopen(url, timeout=60) as r:
            txt = r.read().decode("utf-8", "replace")
        if "$$SOE" not in txt:
            err = [l for l in txt.splitlines() if "No ephemeris" in l or "rror" in l]
            raise RuntimeError("Horizons : " + (err[0].strip() if err else txt[:300]))
        rows = []
        for line in txt.split("$$SOE", 1)[1].split("$$EOE", 1)[0].splitlines():
            c = [x.strip() for x in line.split(",")]
            if len(c) < 12 or not c[0]:
                continue
            # JD, sol, lun, RA, DEC, Illu%, delta, deldot, S-O-T, /r, EcLon, EcLat
            rows.append((float(c[0]), float(c[3]), float(c[4]), float(c[5]) / 100,
                         float(c[6]) * AU_KM, float(c[8]), c[9].strip("/") == "T",
                         float(c[10]), float(c[11])))
        return rows

    def _charger(self, jd0):
        jd1 = jd0 + self.pas / 1440 * self.lot
        s = self._requete(10, jd0, jd1)
        m = self._requete(301, jd0, jd1)
        with self.lock:
```

```
            for k, rows in ((10, s), (301, m)):
                deja = {r[0] for r in self.buf[k]}
                self.buf[k].extend(r for r in rows if r[0] not in deja)
                self.buf[k].sort()
                # on ne garde que le recent
                if len(self.buf[k]) > 4 * self.lot:
                    self.buf[k] = self.buf[k][-3 * self.lot:]
            self.fin = min(self.buf[10][-1][0], self.buf[301][-1][0])

    def _assurer(self, jd):
        if self.fin is None or jd > self.fin - self.pas / 1440:
            if self.prefetch and self.prefetch.is_alive():
                self.prefetch.join()
            if self.fin is None or jd > self.fin - self.pas / 1440:
                self._charger(jd if self.fin is None else self.fin)
        # prelecture quand on a consomme la moitie du lot
        if (self.prefetch is None or not self.prefetch.is_alive()) and \
                jd > self.fin - self.pas / 1440 * self.lot / 2:
            self.prefetch = threading.Thread(target=self._charger, args=(self.fin,), daemon=True)
            self.prefetch.start()

    @staticmethod
    def _interp(rows, jd):
        # deux lignes encadrantes, interpolation (RA/DEC via vecteurs)
        lo, hi = 0, len(rows) - 1
        while hi - lo > 1:
            mid = (lo + hi) // 2
            if rows[mid][0] <= jd:
                lo = mid
            else:
                hi = mid
        a, b = rows[lo], rows[hi]
        t = 0.0 if b[0] == a[0] else max(0.0, min(1.0, (jd - a[0]) / (b[0] - a[0])))
        va, vb = radec_vec(a[1], a[2]), radec_vec(b[1], b[2])
        v = norm(tuple(va[i] + (vb[i] - va[i]) * t for i in range(3)))
        ra = math.degrees(math.atan2(v[1], v[0])) % 360
        dec = math.degrees(math.asin(max(-1, min(1, v[2]))))
        lin = lambda i: a[i] + (b[i] - a[i]) * t
        elon = (a[7] + wrap180(b[7] - a[7]) * t) % 360
        return dict(ra=ra, dec=dec, illum=lin(3), dist_km=lin(4), elong=lin(5),
                    croissante=a[6] if t < 0.5 else b[6], ecl_lon=elon, ecl_lat=lin(8))

    def at(self, jd):
        if jd < JD_MIN_HORIZONS:
            raise RuntimeError("Horizons n'a pas la Lune avant le 16 mars 9999 av. J.-C. "
                               "(--source meeus pour aller plus loin, sans la NASA)")
        self._assurer(jd)
        with self.lock:
            s, m = self.buf[10], self.buf[301]
            return self._interp(s, jd), self._interp(m, jd)


# ---------------------------------------------------------------- Meeus (repli)
class Meeus:
    """Soleil (ch. 25) et Lune (ch. 47, termes principaux), geocentriques
    apparents, coordonnees de la date. Sans reseau. ~0,3 degre pour la Lune ;
    les polynomes derivent aux dates tres anciennes (quelques degres a -10000)."""
    NOM = "formules de Meeus (hors ligne, approximatif)"

    LR = [(0, 0, 1, 0, 6288774, -20905355), (2, 0, -1, 0, 1274027, -3699111),
          (2, 0, 0, 0, 658314, -2955968), (0, 0, 2, 0, 213618, -569925),
          (0, 1, 0, 0, -185116, 48888), (0, 0, 0, 2, -114332, -3149),
          (2, 0, -2, 0, 58793, 246158), (2, -1, -1, 0, 57066, -152138),
          (2, 0, 1, 0, 53322, -170733), (2, -1, 0, 0, 45758, -204586),
          (0, 1, -1, 0, -40923, -129620), (1, 0, 0, 0, -34720, 108743),
          (0, 1, 1, 0, -30383, 104755), (2, 0, 0, -2, 15327, 10321),
```

```
      (0, 0, 1, 2, -12528, 0), (0, 0, 1, -2, 10980, 79661),
      (4, 0, -1, 0, 10675, -34782), (0, 0, 3, 0, 10034, -23210),
      (4, 0, -2, 0, 8548, -21636), (2, 1, -1, 0, -7888, 24208),
      (2, 1, 0, 0, -6766, 30824), (1, 0, -1, 0, -5163, -8379),
      (1, 1, 0, 0, 4987, -16675), (2, -1, 1, 0, 4036, -12831),
      (2, 0, 2, 0, 3994, -10445), (4, 0, 0, 0, 3861, -11650),
      (2, 0, -3, 0, 3665, 14403), (0, 1, -2, 0, -2689, -7003),
      (2, 0, -1, 2, -2602, 0), (2, -1, -2, 0, 2390, 10056)]
B = [(0, 0, 0, 1, 5128122), (0, 0, 1, 1, 280602), (0, 0, 1, -1, 277693),
     (2, 0, 0, -1, 173237), (2, 0, -1, 1, 55413), (2, 0, -1, -1, 46271),
     (2, 0, 0, 1, 32573), (0, 0, 2, 1, 17198), (2, 0, 1, -1, 9266),
     (0, 0, 2, -1, 8822), (2, -1, 0, -1, 8216), (2, 0, -2, -1, 4324),
     (2, 0, 1, 1, 4200), (2, 1, 0, -1, -3359), (2, -1, -1, 1, 2463),
     (2, -1, 0, 1, 2211), (2, -1, -1, -1, 2065), (0, 1, -1, -1, -1870),
     (4, 0, -1, -1, 1828), (0, 1, 0, 1, -1794)]

def __init__(self, *_a, **_k):
    pass

@staticmethod
def _nutation_obliquite(t):
    om = (125.04452 - 1934.136261 * t) * DEG
    ls = (280.4665 + 36000.7698 * t) * DEG
    lm = (218.3165 + 481267.8813 * t) * DEG
    dpsi = (-17.20 * math.sin(om) - 1.32 * math.sin(2 * ls) - 0.23 * math.sin(2 * lm)
            + 0.21 * math.sin(2 * om)) / 3600
    deps = (9.20 * math.cos(om) + 0.57 * math.cos(2 * ls) + 0.10 * math.cos(2 * lm)
            - 0.09 * math.cos(2 * om)) / 3600
    eps0 = 23.4392911 - 0.01300417 * t - 1.639e-7 * t * t + 5.036e-7 * t ** 3
    return dpsi, eps0 + deps

@staticmethod
def _ecl_vers_eq(lon, lat, eps):
    l, b, e = lon * DEG, lat * DEG, eps * DEG
    ra = math.atan2(math.sin(l) * math.cos(e) - math.tan(b) * math.sin(e), math.cos(l))
    dec = math.asin(math.sin(b) * math.cos(e) + math.cos(b) * math.sin(e) * math.sin(l))
    return math.degrees(ra) % 360, math.degrees(dec)

def _soleil(self, t, dpsi, eps):
    l0 = 280.46646 + 36000.76983 * t + 0.0003032 * t * t
    m = (357.52911 + 35999.05029 * t - 0.0001537 * t * t) * DEG
    c = ((1.914602 - 0.004817 * t - 0.000014 * t * t) * math.sin(m)
         + (0.019993 - 0.000101 * t) * math.sin(2 * m) + 0.000289 * math.sin(3 * m))
    nu = m + c * DEG
    e = 0.016708634 - 0.000042037 * t - 0.0000001267 * t * t
    r = 1.000001018 * (1 - e * e) / (1 + e * math.cos(nu))
    lon = (l0 + c + dpsi - 0.00569) % 360           # aberration + nutation
    ra, dec = self._ecl_vers_eq(lon, 0.0, eps)
    return dict(ra=ra, dec=dec, illum=1.0, dist_km=r * AU_KM, elong=0.0,
                croissante=False, ecl_lon=lon, ecl_lat=0.0)

def _lune(self, t, dpsi, eps):
    lp = 218.3164477 + 481267.88123421 * t - 0.0015786 * t * t + t ** 3 / 538841 - t ** 4 / 65194000
    d = 297.8501921 + 445267.1114034 * t - 0.0018819 * t * t + t ** 3 / 545868 - t ** 4 / 113065000
    m = 357.5291092 + 35999.0502909 * t - 0.0001536 * t * t + t ** 3 / 24490000
    mp = 134.9633964 + 477198.8675055 * t + 0.0087414 * t * t + t ** 3 / 69699 - t ** 4 / 14712000
    f = 93.2720950 + 483202.0175233 * t - 0.0036539 * t * t - t ** 3 / 3526000 + t ** 4 / 863310000
    a1, a2, a3 = 119.75 + 131.849 * t, 53.09 + 479264.290 * t, 313.45 + 481266.484 * t
    e = 1 - 0.002516 * t - 0.0000074 * t * t
    sl = sr = sb = 0.0
    for kd, km, kmp, kf, cl, cr in self.LR:
        arg = (kd * d + km * m + kmp * mp + kf * f) * DEG
        ee = e ** abs(km)
        sl += cl * ee * math.sin(arg)
        sr += cr * ee * math.cos(arg)
    for kd, km, kmp, kf, cb in self.B:
```

```
            arg = (kd * d + km * m + kmp * mp + kf * f) * DEG
            sb += cb * e ** abs(km) * math.sin(arg)
        sl += 3958 * math.sin(a1 * DEG) + 1962 * math.sin((lp - f) * DEG) + 318 * math.sin(a2 * DEG)
        sb += (-2235 * math.sin(lp * DEG) + 382 * math.sin(a3 * DEG) + 175 * math.sin((a1 - f) * DEG)
               + 175 * math.sin((a1 + f) * DEG) + 127 * math.sin((lp - mp) * DEG)
               - 115 * math.sin((lp + mp) * DEG))
        lon = (lp + sl / 1e6 + dpsi) % 360
        lat = sb / 1e6
        dist = 385000.56 + sr / 1000
        ra, dec = self._ecl_vers_eq(lon, lat, eps)
        return dict(ra=ra, dec=dec, dist_km=dist, ecl_lon=lon, ecl_lat=lat)

    def at(self, jd):
        t = (jd - J2000) / 36525
        dpsi, eps = self._nutation_obliquite(t)
        s = self._soleil(t, dpsi, eps)
        m = self._lune(t, dpsi, eps)
        # elongation, phase (Meeus ch. 48)
        psi = math.acos(max(-1, min(1, math.sin(s["dec"] * DEG) * math.sin(m["dec"] * DEG)
                                    + math.cos(s["dec"] * DEG) * math.cos(m["dec"] * DEG)
                                    * math.cos((s["ra"] - m["ra"]) * DEG))))
        i = math.atan2(s["dist_km"] * math.sin(psi), m["dist_km"] - s["dist_km"] * math.cos(psi))
        m["illum"] = (1 + math.cos(i)) / 2
        m["elong"] = math.degrees(psi)
        m["croissante"] = 0 < (m["ecl_lon"] - s["ecl_lon"]) % 360 < 180
        return s, m


# ------------------------------------------------------------------ le dino
def phase_nom(illum, croissante):
    if illum < 0.03:
        return "nouvelle lune"
    if illum > 0.97:
        return "pleine lune"
    if illum < 0.47:
        return "croissant" if croissante else "dernier croissant"
    if illum <= 0.53:
        return "premier quartier" if croissante else "dernier quartier"
    return "gibbeuse croissante" if croissante else "gibbeuse decroissante"


def cote(x):
    """x en degres, positif = a droite du dino (sud, il court vers l'est)."""
    if abs(x) < 0.5:
        return "droit devant"
    return f"{abs(x):.1f} deg a {'droite (sud)' if x > 0 else 'gauche (nord)'}"


def calcul(eph, jd, lon):
    s, m = eph.at(jd)
    ds, est, nord, zen = topocentrique(s["ra"], s["dec"], s["dist_km"], jd, lon)
    dm, _, _, _ = topocentrique(m["ra"], m["dec"], m["dist_km"], jd, lon)
    sud = (-nord[0], -nord[1], -nord[2])
    out = {}
    for nom, d, geo in (("soleil", ds, s), ("lune", dm, m)):
        az, el = az_el(d, est, nord, zen)
        devant, droite = dot(d, est), dot(d, sud)
        lateral = math.degrees(math.atan2(droite, devant))       # 0 devant, +90 a droite, 180 derriere
        out[nom] = dict(az=round(az, 3), hauteur=round(el, 3), lateral=round(lateral, 2),
                        devant=devant > 0, ra=round(geo["ra"], 4), dec=round(geo["dec"], 4),
                        ecl_lon=round(geo["ecl_lon"], 4), ecl_lat=round(geo["ecl_lat"], 4),
                        dist_km=round(geo["dist_km"], 1),
                        # sur l'equateur, un astre se leve a l'azimut 90 - dec :
                        # dec > 0 => a gauche (nord) de la route plein est.
                        lever_lateral=round(-geo["dec"], 2))
    # limbe eclaire de la Lune : direction du Soleil vue depuis la Lune,
```

```
    # projetee sur le ciel autour d'elle. 0 = eclaire par le haut, 90 = par la
    # droite du dino, -90 = par la gauche, 180 = par le bas.
    haut = norm(tuple(zen[i] - dot(zen, dm) * dm[i] for i in range(3)))
    droite = cross(dm, haut)          # a droite quand on regarde la Lune (regard x haut)
    vers_soleil = norm(tuple(ds[i] * s["dist_km"] - dm[i] * m["dist_km"] for i in range(3)))
    limbe = math.degrees(math.atan2(dot(vers_soleil, droite), dot(vers_soleil, haut)))
    out["lune"].update(illum=round(m["illum"], 4), elong=round(m["elong"], 3),
                       croissante=bool(m["croissante"]),
                       phase=phase_nom(m["illum"], m["croissante"]), limbe=round(limbe, 1))
    out["nuit"] = out["soleil"]["hauteur"] < -6
    return out


def texte(jd, lon, km, accel, src, c):
    d, h = date_fr(jd)
    hl = (jd + lon / 360 + 0.5) % 1 * 24                          # heure solaire moyenne locale
    s, m = c["soleil"], c["lune"]
    lines = [
        f"{d}, {h} UT  ·  heure locale {int(hl):02d} h {int(hl % 1 * 60):02d}  ·  x{accel:g}",
        f"le dino : equateur, longitude {lon:.2f} deg E, {km:.1f} km parcourus, route plein est",
        f"Soleil : hauteur {s['hauteur']:+6.1f}  azimut {s['az']:6.1f}  {'devant' if s['devant'] else 'derriere'},
         ↪  {cote(s['lateral'] if s['devant'] else wrap180(180 - s['lateral']))}"
        f"  ·  se leve {cote(s['lever_lateral'])} (declinaison {s['dec']:+.1f})",
        f"Lune   : hauteur {m['hauteur']:+6.1f}  azimut {m['az']:6.1f}  {'devant' if m['devant'] else 'derriere'},
         ↪  {cote(m['lateral'] if m['devant'] else wrap180(180 - m['lateral']))}"
        f"  ·  se leve {cote(m['lever_lateral'])} (declinaison {m['dec']:+.1f})",
        f"         {m['phase']}, eclairee a {m['illum'] * 100:.0f} %, elongation {m['elong']:.0f} deg,"
        f" limbe eclaire vers {m['limbe']:+.0f} deg (0 = haut, +90 = droite)",
        f"         latitude ecliptique {m['ecl_lat']:+.2f} deg (inclinaison de l'orbite : jusqu'a 5,1), distance
         ↪  {m['dist_km'] / 1000:.0f} mille km",
        f"{'nuit' if c['nuit'] else 'jour'}  ·  source : {src}",
    ]
    return "\n".join(lines)


def main():
    ap = argparse.ArgumentParser(description="le ciel du dino, aux valeurs de la NASA (JPL Horizons)")
    ap.add_argument("--date", default="BC 9999-03-16 04:00",
                    help="depart : 'BC 3000-06-21 04:00', '-2999-06-21', 'JD 625844.5', ISO (defaut : le plus vieux
                      ↪  ciel qu'Horizons connaisse)")
    ap.add_argument("--lon", type=float, default=0.0, help="longitude de depart en deg est (defaut 0)")
    ap.add_argument("--vitesse", type=float, default=10.0, help="vitesse du dino en m/s, vers l'est (defaut 10)")
    ap.add_argument("--accel", type=float, default=3600.0, help="acceleration du temps (defaut 3600 : une heure
↪  par seconde, la journee de 24 s du jeu)")
    ap.add_argument("--tick", type=float, default=1.0, help="secondes reelles entre deux affichages (defaut 1)")
    ap.add_argument("--duree", type=float, default=0.0, help="secondes reelles avant de s'arreter (0 = sans fin)")
    ap.add_argument("--une-fois", action="store_true", help="un seul instant, puis quitte")
    ap.add_argument("--json", action="store_true", help="une ligne JSON par tick (pour la page du dino)")
    ap.add_argument("--source", choices=("auto", "horizons", "meeus"), default="auto")
    a = ap.parse_args()

    jd0 = parse_date(a.date)
    pas_min = a.accel * a.tick / 60
    eph = None
    if a.source in ("auto", "horizons"):
        try:
            eph = Horizons(pas_min)
            eph.at(jd0)
        except Exception as e:                                    # reseau, date hors DE441...
            if a.source == "horizons":
                sys.exit(f"[ciel] {e}")
            print(f"[ciel] Horizons indisponible ({e}) : repli sur Meeus", file=sys.stderr)
            eph = None
    if eph is None:
        eph = Meeus()
```

```
    tty = sys.stdout.isatty() and not a.json
    t0 = time.monotonic()
    n = 0
    try:
        while True:
            reel = time.monotonic() - t0
            jd = jd0 + reel * a.accel / 86400
            m_parc = a.vitesse * reel * a.accel
            lon = (a.lon + m_parc / 111319.49 + 180) % 360 - 180   # 1 deg = 111,3 km sur l'equateur
            c = calcul(eph, jd, lon)
            if a.json:
                d, h = date_fr(jd)
                c.update(jd=round(jd, 6), date=f"{d}, {h} UT", lon=round(lon, 4), km=round(m_parc / 1000, 2),
                         accel=a.accel, source=eph.NOM)
                print(json.dumps(c, ensure_ascii=False), flush=True)
            else:
                out = texte(jd, lon, m_parc / 1000, a.accel, eph.NOM, c)
                if tty and n:
                    out = "\x1b[7A\x1b[J" + out
                print(out, flush=True)
            n += 1
            if a.une_fois or (a.duree and reel >= a.duree):
                break
            time.sleep(max(0.0, a.tick - ((time.monotonic() - t0) % a.tick)))
    except KeyboardInterrupt:
        pass


if __name__ == "__main__":
    main()
```

3.403 /opt/GSM/osmo-operator/tools/osmo-dino.py

11008 octets, 214 lignes → 214 lignes

```
#!/usr/bin/env python3
# osmo-dino.py - le DINO (jeu "Coureur d'ombres") ANIME, pose sous le Conky.
#
# Remplace osmo-moon.py (le cavalier/lune en gif) : meme fenetre de type BUREAU
# (Gdk.WindowTypeHint.DESKTOP - sous toutes les fenetres, au-dessus du fond, non
# decoree, collante, transparente), meme colonne haut-droite, meme largeur 400,
# collee EN DESSOUS du Conky. Mais le contenu est une page HTML/canvas
# (configs/conky/dino.html) rendue par WebKit2 : le jeu tourne en mode "attract"
# tout seul (le dino court, Soleil/Lune, ombres calculees) - aucune interaction
# requise, comme une animation. On masque header/pied de page par une feuille de
# style injectee pour ne garder que le cadre du jeu.
#
# Reglage par variables d'environnement :
#   OSMO_DINO_HTML   chemin du html    (defaut: configs/conky/dino.html)
#   OSMO_DINO_W      largeur en px     (defaut: 400, la largeur du Conky)
#   OSMO_DINO_H      hauteur en px     (defaut: 260 ~ ratio 8/5 du cadre)
#   OSMO_DINO_GAP_X  marge a droite    (defaut: 24, le gap_x du Conky)
#   OSMO_DINO_GAP_Y  y du haut du jeu  (defaut: 680 = sous le Conky, qui descend
#                    jusque vers 580 ; 600 le laissait colle a lui)
#
# Lance par /usr/local/bin/osmo-desktop-panel (comme osmo-panel.py + conky).
import os
import subprocess
import sys

# [2026-09-06] LA FENETRE DE BUREAU EXIGE X11, PAS WAYLAND.
# Sous Wayland, GTK3 ignore window.move() ET Gdk.WindowTypeHint.DESKTOP : le
# compositeur place la fenetre ou il veut (au milieu, au premier plan) - d ou
# l encart et le dino qui « demarrent n importe ou ». Le protocole ne donne au
```

```
# client aucun moyen de se poser a des coordonnees absolues (il faudrait
# gtk-layer-shell, absent sur mutter). On passe donc par XWayland, ou le
# positionnement X11 est honore, exactement comme le Conky voisin (own_window
# type desktop) qui, lui, se cale bien. On ne force que si un DISPLAY existe :
# sans serveur X, mieux vaut le comportement par defaut qu un echec au demarrage.
X11_FORCE = os.environ.get("XDG_SESSION_TYPE") == "wayland" and bool(os.environ.get("DISPLAY"))
if X11_FORCE:
    os.environ["GDK_BACKEND"] = "x11"

import gi

gi.require_version("Gtk", "3.0")
gi.require_version("Gdk", "3.0")
gi.require_version("WebKit2", "4.1")
from gi.repository import Gdk, GLib, Gtk, WebKit2  # noqa: E402

# [2026-09-06] ET ALORS LE TYPE BUREAU NE VA PLUS. Sur XWayland, une fenetre
# _NET_WM_WINDOW_TYPE_DESKTOP est rangee par GNOME Shell dans la couche du
# bureau, SOUS la fenetre plein ecran des icones (extension DING) - qui, elle,
# avale tous les clics : l encart s affichait mais ses boutons ne repondaient
# plus. On garde donc une fenetre NORMALE, simplement maintenue en dessous des
# autres (keep_below + stick + hors barre des taches) : meme rendu, et les
# clics arrivent. Sur X11 natif (la cle live) le type BUREAU marchait : on n y
# touche pas.
# [2026-09-13] MODE APPLICATION (--app, ou OSMO_DINO_APP=1).
# Le dino n'est plus pose sur le bureau : il est dans le MENU DES APPLICATIONS
# (data/desktop/osmo-dino.desktop), et s'ouvre comme n'importe quel programme.
# Ce qui change, et c'est tout : une fenetre ORDINAIRE - decoree, redimension-
# nable, dans la barre des taches, centree, ni collante ni maintenue dessous -
# et la page rendue telle quelle, sans la feuille de style qui rognait le cadre
# pour tenir dans un coin de 400x260. Le bouton plein ecran de la page, lui,
# n'a plus a deleguer a osmo-dino-play : cette fenetre-ci PEUT s'agrandir.
APP_MODE = os.environ.get("OSMO_DINO_APP") == "1" or "--app" in sys.argv[1:]

TYPE_HINT = Gdk.WindowTypeHint.NORMAL if (X11_FORCE or APP_MODE) else Gdk.WindowTypeHint.DESKTOP
# [2026-09-06] ... ET « EN DESSOUS » NON PLUS. Deuxieme moitie du meme piege :
# avec _NET_WM_STATE_BELOW, GNOME Shell range la fenetre XWayland sous sa
# propre couche de bureau (native Wayland), qui est plein ecran et prend tous
# les clics - la fenetre restait visible mais totalement inerte. Sous XWayland
# on renonce donc a « toujours dessous » : fenetre ordinaire, collante et hors
# barre des taches. Elle passera au premier plan quand on clique dedans, ce qui
# est le prix a payer pour qu on puisse justement cliquer dedans. Mettre
# OSMO_DESKTOP_BELOW=1 pour retrouver l ancien comportement (fenetre sous tout,
# clics perdus) ; sur X11 natif rien ne change.
KEEP_BELOW = (not APP_MODE) and ((not X11_FORCE) or os.environ.get("OSMO_DESKTOP_BELOW") == "1")

REPO = os.environ.get("OSMO_REPO", "/opt/GSM/osmo-operator")
HTML = os.environ.get("OSMO_DINO_HTML", os.path.join(REPO, "configs/conky/dino.html"))
WIN_W = int(os.environ.get("OSMO_DINO_W", "880" if APP_MODE else "400"))  # widget : la largeur du Conky
WIN_H = int(os.environ.get("OSMO_DINO_H", "560" if APP_MODE else "260"))
GAP_X = int(os.environ.get("OSMO_DINO_GAP_X", "24"))  # meme gap_x que le Conky
GAP_Y = int(os.environ.get("OSMO_DINO_GAP_Y", "720"))  # sous le Conky, qui descend jusque vers 700

# On ne garde que le cadre du jeu : ni titre ni pied de page, cadre a ras bord.
INJECT_CSS = (
    "header,footer{display:none!important}"
    "body{padding:0!important;margin:0!important;min-height:auto!important;"
    "gap:0!important;background:var(--ink)!important}"
    ".frame{width:100%!important;max-width:none!important;border:0!important;"
    "border-radius:0!important;box-shadow:none!important;aspect-ratio:auto!important;"
    "height:100vh!important}"
)


class Dino(Gtk.Window):
    def __init__(self):
```

```
super().__init__(title="osmo-dino")
if not os.path.exists(HTML):
    print(f"[dino] html introuvable: {HTML}", file=sys.stderr)
    sys.exit(1)
self.w, self.h = WIN_W, WIN_H
self._plein = False

if APP_MODE:
    # Une fenetre comme les autres : le gestionnaire de fenetres la
    # place, la redimensionne et la range dans la barre des taches.
    # Pas de move(), donc pas de _pin non plus - rien ne cherche a la
    # remettre a un coin qu'elle n'a plus.
    self.x = self.y = 0
    print(f"[dino] application {self.w}x{self.h} ({HTML})", flush=True)
    self.set_type_hint(TYPE_HINT)
    self.set_default_size(self.w, self.h)
    self.set_position(Gtk.WindowPosition.CENTER)
    self.set_icon_name("osmo-dino")
else:
    # colonne haut-droite du moniteur primaire, comme le Conky (top_right)
    disp = Gdk.Display.get_default()
    mon = disp.get_primary_monitor() or disp.get_monitor(0)
    g = mon.get_geometry()
    self.x = g.x + g.width - GAP_X - self.w
    self.y = g.y + GAP_Y
    print(f"[dino] {self.w}x{self.h} @ {self.x},{self.y} ({HTML})", flush=True)

    # fenetre de type BUREAU (cf. osmo-panel.py / osmo-moon.py)
    self.set_type_hint(TYPE_HINT)
    self.set_decorated(False)
    self.set_resizable(False)
    self.set_keep_below(KEEP_BELOW)
    self.set_skip_taskbar_hint(True)
    self.set_skip_pager_hint(True)
    self.stick()
    self.set_default_size(self.w, self.h)
    self.set_size_request(self.w, self.h)
    self.move(self.x, self.y)

screen = self.get_screen()
visual = screen.get_rgba_visual()
if visual and screen.is_composited():
    self.set_visual(visual)
self.set_app_paintable(True)
self.connect("destroy", Gtk.main_quit)

# WebView : feuille de style injectee (masque header/footer) + fond
# transparent (le corps de la page reste sombre, c'est voulu).
ucm = WebKit2.UserContentManager()
# La feuille de style ne sert qu'au widget : elle masque header et pied
# de page pour ne garder que le cadre du jeu dans 400x260. En
# application, la page s'affiche entiere, comme dans un navigateur.
if not APP_MODE:
    ucm.add_style_sheet(WebKit2.UserStyleSheet(
        INJECT_CSS, WebKit2.UserContentInjectedFrames.ALL_FRAMES,
        WebKit2.UserStyleLevel.USER, None, None))
# [2026-09-07] LE BOUTON PLEIN ECRAN DU WIDGET. Une page ne peut pas
# agrandir une fenetre GTK, et ce widget de 400x260 n a de toute facon
# rien a agrandir : c est un coin du bureau. La page se sait donc
# « widget » (OSMO_HOST) et, au clic, nous envoie un message WebKit -
# on ouvre alors le VRAI jeu, en plein ecran, dans sa propre fenetre.
# 'widget' fait deleguer le plein ecran (la page le sait : dino.html
# l.519). En application, l'hote PEUT s'agrandir : on annonce 'app', et
# le message {full:...} est traite plus bas par _on_message.
ucm.add_script(WebKit2.UserScript(
    "window.OSMO_HOST='app';" if APP_MODE else "window.OSMO_HOST='widget';",
```

```
            WebKit2.UserContentInjectedFrames.ALL_FRAMES,
            WebKit2.UserScriptInjectionTime.START, None, None))
        ucm.connect("script-message-received::osmo", self._on_message)
        ucm.register_script_message_handler("osmo")
        self.view = WebKit2.WebView.new_with_user_content_manager(ucm)
        self.view.set_background_color(Gdk.RGBA(0, 0, 0, 0))
        self.add(self.view)
        self.view.load_uri(GLib.filename_to_uri(HTML, None))

        self.show_all()
        self._pos = (self.x, self.y)
        if not APP_MODE:
            self._pin()
            self.connect("map-event", self._pin)
            GLib.timeout_add(500, self._pin)

    # [2026-09-06] SE REPOSER APRES COUP. Meme sous X11, le gestionnaire de
    # fenetres peut deplacer la fenetre au moment ou il la mappe (mutter le
    # fait pour les fenetres non decorees qu il ne reconnait pas comme du
    # bureau). Un seul move() avant show_all() ne tient donc pas : on recale
    # au map-event, puis une derniere fois une demi-seconde plus tard.
    def _pin(self, *_a):
        self.move(*self._pos)
        return False

    def _on_message(self, _ucm, result):
        """{full:...} : en application, CETTE fenetre ; en widget, celle du jeu."""
        if APP_MODE:
            # La page envoie {full:true} ou {full:false} - on suit.
            try:
                val = result.get_js_value()
                plein = bool(val.object_get_property("full").to_boolean())
            except Exception:
                plein = not self._plein
            self._plein = plein
            self.fullscreen() if plein else self.unfullscreen()
            return
        for cand in ("/usr/local/bin/osmo-dino-play",):
            if os.path.exists(cand):
                subprocess.Popen([cand, "--fullscreen"], stdin=subprocess.DEVNULL,
                                 stdout=subprocess.DEVNULL, stderr=subprocess.DEVNULL,
                                 start_new_session=True)
                return
        print("[dino] osmo-dino-play absent (tools/osmo-extras-install.sh le pose)", flush=True)


def main():
    Dino()
    Gtk.main()


if __name__ == "__main__":
    main()
```

3.404 /opt/GSM/osmo-operator/tools/osmo-drivers.sh

6054 octets, 115 lignes → 115 lignes

```
#!/bin/bash
# ============================================================================
#  osmo-drivers - le pilote graphique NVIDIA : etat, installation, mise a jour
# ============================================================================
#  Le pendant, sur le systeme qui tourne (live ou installe), de la page
#  "Pilotes graphiques" de l installeur : montre la carte, le module charge et
```

```
#  ce qui est installe, et laisse installer ou mettre a jour.
#
#  [2026-09-04] LE PILOTE DU BANC EST FIGE : nvidia-driver-610. `ubuntu-drivers`
#  n arbitre plus rien - il interrogeait les depots et decidait seul, si bien
#  qu on n installait pas le meme pilote d une machine a l autre, et rien du
#  tout quand ubuntu-drivers-common manquait. Ici c est un paquet, nomme :
#      sudo apt install nvidia-driver-610
#  ubuntu-drivers reste utilise s il est la, mais UNIQUEMENT pour informer
#  (la liste et le "recommande" affiches par --status). OSMO_NVIDIA_DRIVER
#  permet d en viser un autre.
#
#      sudo ./tools/osmo-drivers.sh            menu whiptail
#      sudo ./tools/osmo-drivers.sh --status   l etat seul, en texte
#      sudo ./tools/osmo-drivers.sh --install [pilote]   (defaut : nvidia-driver-610)
#      sudo ./tools/osmo-drivers.sh --update
# ============================================================================
set -u
RED='\033[0;31m'; GREEN='\033[0;32m'; YELLOW='\033[1;33m'; CYAN='\033[0;36m'; NC='\033[0m'
export DEBIAN_FRONTEND=noninteractive

[ "$(id -u)" -eq 0 ] || exec pkexec --disable-internal-agent "$0" "$@" 2>/dev/null || exec sudo "$0" "$@"

need() { command -v "$1" >/dev/null 2>&1; }
# NVDRV : le pilote que ce script installe. Un nom de paquet, pas une enquete.
NVDRV="${OSMO_NVIDIA_DRIVER:-nvidia-driver-610}"
# ubuntu-drivers n est plus une dependance : on ne l installe pas pour lui-meme.
# S il est deja la, --status s en sert pour afficher le "recommande" d Ubuntu -
# a titre d information, a cote du pilote que le banc installe reellement.

gpu()       { lspci -d 10de: 2>/dev/null | head -1 | cut -d: -f3- | sed 's/^ //; s/ (rev.*//'; }
reco()      { ubuntu-drivers devices 2>/dev/null | awk '/^driver *:/ && /recommended/{print $3; exit}'; }
avail()     { { echo "$NVDRV"; ubuntu-drivers list 2>/dev/null | awk '/^nvidia-driver-/{print $1}' | sed
 ↪  's/,.*//'; } | sort -u; }
installed() { dpkg -l 'nvidia-driver-*' 2>/dev/null | awk '/^ii/{print $2}'; }
loaded()    { lsmod 2>/dev/null | awk '$1=="nvidia"||$1=="nouveau"{print $1}' | paste -sd, -; }

status_text() {
    local g r
    g="$(gpu)"; r="$(reco)"
    echo "Carte NVIDIA : ${g:-aucune (lspci vendor 10de)}"
    echo "Module charge : $(loaded)"
    echo "Pilote du banc: $NVDRV  ($(apt-cache policy "$NVDRV" 2>/dev/null | awk '/Candidate:|Candidat/{print $2;
     ↪  exit}'))"
    echo "Recommande Ubuntu : ${r:-inconnu (ubuntu-drivers absent ou pas de reseau)}"
    echo "Installe      : $(installed | paste -sd' ' -)"
    [ -z "$(installed)" ] && echo "                (aucun pilote proprietaire)"
    echo "Disponibles   :"
    local d st
    for d in $(avail); do
        st=""
        [ "$d" = "$NVDRV" ] && st="$st pilote du banc"
        [ "$d" = "$r" ] && st="$st recommande Ubuntu"
        dpkg -s "$d" >/dev/null 2>&1 && st="$st installe"
        echo "   $d${st:+  [$st ]}"
    done
    if need nvidia-smi; then echo; nvidia-smi --query-gpu=name,driver_version,memory.used,memory.total
     ↪  --format=csv,noheader 2>/dev/null | sed 's/^/nvidia-smi : /'; fi
}

do_install() {
    local what="${1:-$NVDRV}"
    [ "$what" = recommended ] && what="$NVDRV"   # ancien nom d option, meme resultat
    if ! lspci -n 2>/dev/null | grep -qi 10de:; then
        echo -e "${RED}Aucune carte NVIDIA sur cette machine : rien a installer.${NC}"; return 1
    fi
```

```
    apt-get update -qq || { echo -e "${RED}Pas de reseau : les pilotes viennent des depots Ubuntu.${NC}"; return
     ↪  1; }
    echo -e "${CYAN}apt install $what...${NC}"
    if ! apt-get install -y "$what"; then
        echo -e "${RED}$what non installe.${NC} Il vit dans multiverse :"
        echo -e "   ${CYAN}sudo add-apt-repository multiverse && sudo apt install $what${NC}"
        return 1
    fi
    update-initramfs -u >/dev/null 2>&1 || true
    echo -e "${GREEN}Termine. Redemarrez pour charger le module nvidia.${NC}"
}

do_update() {
    local pk; pk="$(dpkg -l 'nvidia-*' 'libnvidia-*' 'linux-modules-nvidia-*' 2>/dev/null | awk '/^ii/{print $2}'
     ↪  | paste -sd' ' -)"
    [ -n "$pk" ] || { echo -e "${YELLOW}Aucun paquet NVIDIA installe : rien a mettre a jour (--install d
     ↪  abord).${NC}"; return 1; }
    apt-get update -qq || { echo -e "${RED}Pas de reseau.${NC}"; return 1; }
    echo -e "${CYAN}Mise a jour de : $pk${NC}"
    # shellcheck disable=SC2086
    apt-get install -y --only-upgrade $pk
    update-initramfs -u >/dev/null 2>&1 || true
    echo -e "${GREEN}Termine.${NC}"
}

case "${1:-}" in
    --status)  status_text; exit 0 ;;
    --install) do_install "${2:-$NVDRV}"; exit $? ;;
    --update)  do_update; exit $? ;;
esac

need whiptail || { status_text; echo; echo "whiptail absent : --install [pilote|recommended] / --update"; exit 0; }
while :; do
    items=(status "Etat : carte, module charge, pilote du banc, installes")
    for d in $(avail); do
        st="installer"; dpkg -s "$d" >/dev/null 2>&1 && st="deja installe - reinstaller"
        [ "$d" = "$NVDRV" ] && st="$st (pilote du banc)"
        items+=("$d" "$st")
    done
    items+=(update "Mettre a jour les paquets NVIDIA installes")
    items+=(quit "Quitter")
    choice=$(whiptail --title "Pilotes graphiques - $(gpu | cut -c1-50)" \
        --menu "Pilote du banc : $NVDRV. Machine : $(hostname). Module charge : $(loaded)" 22 78 12 "${items[@]}"
         ↪  3>&1 1>&2 2>&3) || exit 0
    case "$choice" in
        status)      status_text | whiptail --title "Etat des pilotes" --textbox /dev/stdin 24 78 ;;
        update)      do_update; read -rp "Entree pour continuer..." _ ;;
        quit|"")     exit 0 ;;
        *)           do_install "$choice"; read -rp "Entree pour continuer..." _ ;;
    esac
done
```

3.405 /opt/GSM/osmo-operator/tools/osmo-epc.sh

11466 octets, 224 lignes → 224 lignes

```
#!/bin/bash
# ============================================================================
# osmo-epc.sh - LE COEUR 4G DU BANC : Open5GS (EPC), en remplacement de srsepc.
#
# [2026-09-07] srsepc etait un coeur de demonstration : mono-UE, sans vraie
# gestion de session et sans interface SGs. Open5GS apporte un EPC complet,
# et surtout le SGs vers OsmoMSC qui permet le CSFB (voix et SMS depuis le
# LTE). Plan d adressage : les UE sont en 10.45.0.0/16 sur ogstun (PAS le
```

```
# 172.17.0.0/24 de srsepc, qui recouvrait docker0).
#
# [2026-09-08] OPEN5GS VIT DANS /opt/LTE, PAS DANS /root. Il avait ete compile
# dans /root/open5gs/install : un coeur de reseau dans le home de root n est
# ni dans l ISO, ni dans un .deb, ni relisible par le compte de session. Le
# prefixe est maintenant /opt/LTE/open5gs/install (le Dockerfile, le .deb
# osmo-build-open5gs et tools/osmo-lte-install.sh le posent la) ; l ancien
# chemin reste accepte, en dernier recours, pour une machine pas encore
# migree. Les configs FONT PARTIE DU DEPOT : configs/open5gs/*.yaml, posees
# par osmo-lte-install.sh dans $PREFIX/etc/open5gs.
#
#   osmo-epc start        mongod, ogstun + NAT, puis les demons dans l ordre
#   osmo-epc stop         arrete les demons (ogstun reste)
#   osmo-epc restart
#   osmo-epc status       un etat par demon, ogstun, SGs
#   osmo-epc subscribers  pose les abonnes du depot (configs/open5gs/
#                         subscribers.json) dans MongoDB s ils n y sont pas
#   osmo-epc log [nom]    le journal d un demon (mme par defaut), en direct
#
# Variables : OPEN5GS_PREFIX, SRS_WAN (interface de sortie du NAT ; par defaut
# celle de la route par defaut), OPEN5GS_UESUBNET, OPEN5GS_TUNADDR.
# ============================================================================
set -u

REPO="${OSMO_REPO:-/opt/GSM/osmo-operator}"
# Le prefixe : la variable, sinon /opt/LTE, sinon l ancien /root.
_prefix() {
    local p
    for p in "${OPEN5GS_PREFIX:-}" /opt/LTE/open5gs/install /root/open5gs/install; do
        [ -n "$p" ] && [ -x "$p/bin/open5gs-mmed" ] && { echo "$p"; return 0; }
    done
    echo "${OPEN5GS_PREFIX:-/opt/LTE/open5gs/install}"
}
PREFIX="$(_prefix)"
BIN="$PREFIX/bin"
ETC="$PREFIX/etc/open5gs"
LOG="$PREFIX/var/log/open5gs"
UESUBNET="${OPEN5GS_UESUBNET:-10.45.0.0/16}"
TUNADDR="${OPEN5GS_TUNADDR:-10.45.0.1/16}"
# L interface de sortie du NAT : SRS_WAN, sinon celle de la route par defaut.
# (Elle etait ecrite en dur - enxa0cec8a98b06 - : une autre machine n a pas
# cette carte-la.)
WAN="${SRS_WAN:-$(ip -4 route show default 2>/dev/null | awk '/default/ {print $5; exit}')}"
# Ordre de demarrage : le NRF (les SBI du SMF le cherchent), le HSS avant le
# MME (Diameter), les plans usager avant les plans controle qui s y rattachent.
DAEMONS="open5gs-nrfd open5gs-hssd open5gs-sgwcd open5gs-sgwud open5gs-smfd open5gs-upfd open5gs-pcrfd
 ↪  open5gs-mmed"

RED='\033[0;31m'; GREEN='\033[0;32m'; YELLOW='\033[1;33m'; CYAN='\033[0;36m'; NC='\033[0m'
_ok()   { echo -e "  ${GREEN}✓${NC} $*"; }
_warn() { echo -e "  ${YELLOW}!${NC} $*"; }
_err()  { echo -e "  ${RED}✗${NC} $*" >&2; }
_root() { [ "$(id -u)" -eq 0 ] && return 0; _err "il faut root (tun, NAT, mongod) : sudo $0 $*"; return 1; }

[ -x "$BIN/open5gs-mmed" ] || { _err "Open5GS introuvable ($BIN/open5gs-mmed) - « osmo-lte-install --build » ou
 ↪  dpkg -i osmo-build-open5gs"; [ "${1:-}" = status ] || exit 1; }

# [2026-09-08] MONGOD D ABORD. Le HSS et le PCRF lisent les abonnes dans
# MongoDB et MEURENT a l initialisation s il ne tourne pas (« Failed to
# connect to server [mongodb://localhost/open5gs] ») : le MME repond alors
# 3002 sur S6a et rejette tout attach (EMM cause 15). Le service peut etre
# « disabled » : on le lance ici, a chaque fois.
_mongod() {
    systemctl is-active --quiet mongod && return 0
    if systemctl start mongod 2>/dev/null; then
        _ok "mongod demarre"; sleep 2
```

```
    else
        _err "mongod ne demarre pas (HSS et PCRF ne tiendront pas) : apt install mongodb-org"
        return 1
    fi
}

# Les abonnes du depot, poses s ils manquent. Le champ slice[].session[].smf
# (127.0.0.200, valeur par defaut du WebUI) est RETIRE : il a priorite sur
# gtpc.client.smf de mme.yaml, le SGW-C envoyait le Create Session vers
# 127.0.0.200, personne ne repondait, GTP Timeout, RRC Release.
epc_subscribers() {
    _root subscribers || return 1
    _mongod || return 1
    local dump="$REPO/configs/open5gs/dump/open5gs" j="$REPO/configs/open5gs/subscribers.json" n
    command -v mongosh >/dev/null 2>&1 || { _warn "mongosh absent : abonnes non poses"; return 0; }
    n="$(mongosh --quiet open5gs --eval 'print(db.subscribers.countDocuments())' 2>/dev/null | tail -1)"
    # Base vide et un mongodump dans le depot (configs/open5gs/dump : abonnes,
    # comptes du WebUI, sessions) : on restaure TOUT, tel quel (BSON : les
    # Long du sqn et les ObjectId restent ce qu ils sont).
    if [ "${n:-0}" = "0" ] && [ -f "$dump/subscribers.bson" ] && command -v mongorestore >/dev/null 2>&1; then
        mongorestore --quiet --db open5gs "$dump" >/dev/null 2>&1 \
            && _ok "base open5gs restauree depuis le depot (mongodump)" \
            || _warn "mongorestore a echoue - on essaie le JSON"
    fi
    [ -s "$j" ] || return 0
    mongosh --quiet open5gs --eval "
        const subs = JSON.parse(require('fs').readFileSync('$j', 'utf8'));
        let n = 0;
        for (const s of subs) {
            // JSON.stringify d un Long donne {high, low, unsigned} : on le refait Long.
            if (s.security && s.security.sqn && typeof s.security.sqn === 'object')
                s.security.sqn = Long.fromBits(s.security.sqn.low, s.security.sqn.high);
            delete s.mme_timestamp;
            if (db.subscribers.countDocuments({ imsi: s.imsi }) === 0) { db.subscribers.insertOne(s); n++; }
        }
        db.subscribers.updateMany({ 'slice.session.smf': { \$exists: true } }, { \$unset: {
↪   'slice.$[].session.$[].smf': '' } });
        print('abonnes en base : ' + db.subscribers.countDocuments() + ' (' + n + ' ajoute(s) depuis le depot)');
    " 2>&1 | sed 's/^/  /'
}

setup_tun() {
    if ! ip link show ogstun >/dev/null 2>&1; then
        ip tuntap add name ogstun mode tun && _ok "ogstun cree"
    fi
    ip addr replace "$TUNADDR" dev ogstun
    ip link set ogstun up
    sysctl -q -w net.ipv4.ip_forward=1
    if [ -n "$WAN" ]; then
        iptables -t nat -C POSTROUTING -s "$UESUBNET" -o "$WAN" -j MASQUERADE 2>/dev/null \
            || iptables -t nat -A POSTROUTING -s "$UESUBNET" -o "$WAN" -j MASQUERADE
        _ok "ogstun $TUNADDR, NAT $UESUBNET -> $WAN"
    else
        _warn "ogstun $TUNADDR, pas de route par defaut : pas de NAT (SRS_WAN=<iface>)"
    fi
    setup_forward
}

# [2026-09-16] LE FORWARD, SANS QUOI LE NAT NE SERT A RIEN. Docker pose une
# politique FORWARD a DROP sur l hote : un paquet de l UE (10.45.0.2, via
# ogstun) vers Internet etait NATe puis jete, sans refus ni trace. Vu depuis
# le telephone pmOS : PPP monte, 10.99.0.2 attribuee, ping de la passerelle
# OK, et rien au-dela - NetworkManager restait « connecte (local) ». Meme
# constat que start-docker.sh pour le WAN : DOCKER-USER est consultee avant les
# regles de docker et il ne la reecrit pas ; on y autorise tout ce qui entre
# ou sort par ogstun. Rejoue a chaque « osmo-epc start » (donc a chaque
```

```
# « osmo-lte start »), et par osmo-pmos-qemu a chaque lancement de la VM.
setup_forward() {
    local spec
    iptables -N DOCKER-USER 2>/dev/null || true
    for spec in "-i ogstun" "-o ogstun"; do
        # shellcheck disable=SC2086
        iptables -C DOCKER-USER $spec -j ACCEPT 2>/dev/null \
            || iptables -I DOCKER-USER 1 $spec -j ACCEPT 2>/dev/null \
            || { _warn "FORWARD ($spec) : iptables refuse - la data des UE n ira pas plus loin que l hote"; return
             ↪  0; }
    done
    # Les regles DOCKER-USER ne comptent que si FORWARD y passe : docker le
    # fait (-A FORWARD -j DOCKER-USER) ; sans docker, on saute l etape.
    iptables -C FORWARD -j DOCKER-USER 2>/dev/null || iptables -I FORWARD 1 -j DOCKER-USER 2>/dev/null || true
    _ok "FORWARD ogstun <-> $WAN autorise (DOCKER-USER)"
}

epc_start() {
    _root start || return 1
    _mongod || true
    epc_subscribers >/dev/null 2>&1 || true
    mkdir -p "$LOG"
    setup_tun
    local d y
    for d in $DAEMONS; do
        if pgrep -x "$d" >/dev/null; then _ok "$d deja en cours"; continue; fi
        # -D : demon. La config est celle de $ETC (le prefixe), posee depuis
        # le depot par osmo-lte-install.sh. [2026-09-09] open5gs-nrfd lit
        # nrf.yaml, pas nrfd.yaml : le -c echouait TOUJOURS et c est le
        # repli sans -c (le chemin compile dans le binaire) qui demarrait
        # tout - meme prefixe, donc meme fichier, mais un OPEN5GS_PREFIX
        # different n aurait jamais lu ses configs.
        # Et stdout AUSSI vers /dev/null : -D ne detache pas les descripteurs,
        # le demon gardait le tube « osmo-epc start | sed » d osmo-lte.sh
        # ouvert, et sed n en voyait jamais la fin.
        y="${d#open5gs-}"; y="$ETC/${y%d}.yaml"
        if "$BIN/$d" -D -c "$y" >/dev/null 2>&1; then :; else
            "$BIN/$d" -D >/dev/null 2>&1
        fi || _err "ECHEC $d (voir $LOG/$(basename "$y" .yaml).log)"
        sleep 0.5
        # [2026-09-09] « demarre » NE SUFFISAIT PAS : -D rend la main avant que
        # le demon ait lu sa config. Sur l ISO, un MME sans freeDiameter/
        # mme.conf (rien de freeDiameter dans le .deb : voir configs/open5gs/
        # freeDiameter/mme.conf) donnait « ✓ open5gs-mmed demarre » puis
        # mourait aussitot - le S1 n ecoutait pas, srsUE recevait
        # ConnectionReject, et personne ne disait pourquoi. On regarde donc
        # s il est TOUJOURS la ; sinon on le rejoue 3 s au premier plan pour
        # rapporter la raison (ses lignes FATAL / ERROR).
        if pgrep -x "$d" >/dev/null; then _ok "$d demarre"
        else
            _err "$d est mort a l init :"
            timeout 3 "$BIN/$d" -c "$y" 2>&1 | grep -aE 'FATAL|ERROR' | grep -v backtrace | head -3 | sed 's/^/
             ↪  /'
        fi
    done
    sleep 1
    ss -np 2>/dev/null | grep -q 29118 && _ok "SGs : association vers OsmoMSC etablie" \
        || _warn "SGs : pas encore d association (osmo-msc lance ?)"
}

epc_stop() {
    _root stop || return 1
    local d n=0
    for d in $DAEMONS; do pgrep -x "$d" >/dev/null && { pkill -TERM -x "$d"; n=$((n + 1)); }; done
    sleep 1
    for d in $DAEMONS; do pgrep -x "$d" >/dev/null && pkill -KILL -x "$d"; done
```

```
    [ "$n" -gt 0 ] && _ok "$n demon(s) Open5GS arrete(s)" || _ok "rien ne tournait"
}

epc_status() {
    echo -e "\033[1mOpen5GS - le coeur 4G du banc\033[0m  ($PREFIX)"
    local d
    for d in $DAEMONS; do
        pgrep -x "$d" >/dev/null && _ok "$d" || _warn "$d : arrete"
    done
    systemctl is-active --quiet mongod && _ok "mongod actif" || _warn "mongod arrete"
    ip -br addr show ogstun 2>/dev/null | sed 's/^/  /' || _warn "ogstun absent"
    ss -San 2>/dev/null | grep -q "ESTAB.*:36412" && _ok "S1 : un eNB est associe" || _warn "S1 : aucun eNB"
}

case "${1:-status}" in
    start)       epc_start ;;
    stop)        epc_stop ;;
    restart)     epc_stop; sleep 1; epc_start ;;
    status)      epc_status ;;
    subscribers) epc_subscribers ;;
    log)         exec tail -F "$LOG/${2:-mme}.log" ;;
    *) sed -n '/^#   osmo-epc start/,/^#   osmo-epc log/p' "$0" | sed 's/^# \{0,2\}//'; exit 2 ;;
esac
```

3.406 /opt/GSM/osmo-operator/tools/osmo-extras-install.sh

21776 octets, 473 lignes → 473 lignes

```
#!/bin/bash
# ============================================================================
# osmo-extras-install.sh - JEUX + MEDIA, poses une seule fois pour DEUX chemins.
#
# Meme logique que le reste du depot : ce qui doit exister a la fois dans le
# SUPPLEMENT (addition.sh, sur une machine deja installee) et dans le NATIF
# (l ISO, iso_modules/80-chroot.sh) est ecrit UNE fois ici, et les deux chemins
# sourcent ce script et appellent osmo_extras_install. Rien n est fatal : un
# paquet absent du miroir apt ne doit pas arreter le reste.
#
# Ce qu il pose :
#   - [addition.sh seulement, pas l ISO : OSMO_EXTRAS_JEUX=0 les saute, ~470 Mo]
#     deux jeux LIBRES, donnees comprises, verifies sur Ubuntu noble (conteneur
#     + Xvfb, 2026-09-25) :
#       Doom  : freedoom (WAD libres) + chocolate-doom (moteur que le lanceur
#               freedoom2 sait trouver ; dsda-doom, lui, n est pas reconnu) ;
#       Quake : openarena (moteur ioquake3, Quake III, ~410 Mo de donnees).
#     Retires : gzdoom et openra (absents de noble), quakespasm (sans le
#     pak0.pak proprietaire il s arrete au demarrage), Unreal (aucune version
#     libre) ;
#   - kodi (mediacenter) ;
#   - wmctrl : l encart (osmo-panel.py) en a besoin pour CALER la fenetre lancee
#     sur le cadre FFT ;
#   - le DINO jouable (osmo-dino-play, plein ecran ou cale) a partir du meme
#     configs/conky/dino.html que l animation du bureau ;
#   - YOUTUBE dans un vrai Firefox (connexion Google possible) avec uBlock
#     Origin force par POLITIQUE d entreprise (policies.json) - pas un client
#     tiers, pour garder le login Google ;
#   - les .desktop qui manquent (youtube, dino) et deux DOSSIERS d applications
#     GNOME : « Jeux » et « Media ».
#
# Idempotent : relancable sans degat. Utilisable en session (gsettings dispo) ET
# en chroot ISO (pas de session : on passe alors par le gschema override, lu par
# la session live au boot).
# ============================================================================
```

```
# Ces couleurs peuvent deja exister (addition.sh les definit) : on ne rale pas.
: "${GREEN:=}"; : "${YELLOW:=}"; : "${CYAN:=}"; : "${BOLD:=}"; : "${NC:=}"

# Le depot, pour retrouver dino.html. addition.sh pose DIR ; l ISO pose REPO ou
# rien (le depot est copie dans l image a un chemin fixe).
OSMO_EXTRAS_REPO="${DIR:-${REPO:-/opt/GSM/osmo-operator}}"

_ex_say()  { echo -e "  ${CYAN}\xe2\x86\x92${NC} $*"; }
_ex_ok()   { echo -e "      ${GREEN}\xe2\x9c\x93${NC} $*"; }
_ex_warn() { echo -e "      ${YELLOW}!${NC} $*"; }

# -- LES PAQUETS ------------------------------------------------------------
# Non fatal paquet par paquet : sur un miroir ou l un manque (universe pas
# active, version), les autres passent quand meme.
_osmo_extras_apt() {
    export DEBIAN_FRONTEND=noninteractive
    apt-get update -y >/dev/null 2>&1 || true
    # wireshark demande en debconf s il faut autoriser la capture sans root : on
    # tranche NON (l encart le lance en root de toute facon, cf. osmo-wireshark-
    # root) pour que l install passe sans question.
    echo "wireshark-common wireshark-common/install-setuid boolean false" \
        | debconf-set-selections 2>/dev/null || true
    local p
    for p in wmctrl xdotool \
             qemu-system-x86 qemu-system-gui qemu-system-modules-opengl ovmf sshpass kpartx \
             kodi \
             wireshark \
             gir1.2-webkit2-4.1; do
        if dpkg -s "$p" >/dev/null 2>&1; then
            _ex_ok "$p deja present"
        elif apt-get install -y --no-install-recommends "$p" >/dev/null 2>&1; then
            _ex_ok "$p installe"
        else
            _ex_warn "$p indisponible (apt) - ignore"
        fi
    done
    # linphone : le paquet a change de nom selon la version (linphone-desktop
    # recent, linphone ancien). On prend le premier qui s installe.
    if dpkg -s linphone-desktop >/dev/null 2>&1 || dpkg -s linphone >/dev/null 2>&1; then
        _ex_ok "linphone deja present"
    elif apt-get install -y --no-install-recommends linphone-desktop >/dev/null 2>&1 \
      || apt-get install -y --no-install-recommends linphone >/dev/null 2>&1; then
        _ex_ok "linphone installe"
    else
        _ex_warn "linphone indisponible (apt) - ignore"
    fi
    # ofono : pile telephonie (daemon + outils de test)
    if dpkg -s ofono >/dev/null 2>&1; then
        _ex_ok "ofono deja present"
    elif apt-get install -y --no-install-recommends ofono >/dev/null 2>&1; then
        _ex_ok "ofono installe"
    else
        _ex_warn "ofono indisponible (apt) - ignore"
    fi
}

# -- LES JEUX : SUPPLEMENT SEULEMENT ----------------------------------------
# [2026-09-25] Hors de l ISO : ~470 Mo pour deux jeux. addition.sh --extras les
# pose (OSMO_EXTRAS_JEUX=1, le defaut) ; iso_modules/80-chroot.sh passe 0.
_osmo_extras_jeux() {
    export DEBIAN_FRONTEND=noninteractive
    local p
    for p in freedoom chocolate-doom openarena; do
        if dpkg -s "$p" >/dev/null 2>&1; then
            _ex_ok "$p deja present"
        elif apt-get install -y --no-install-recommends "$p" >/dev/null 2>&1; then
```

```
            _ex_ok "$p installe"
        else
            _ex_warn "$p indisponible (apt) - ignore"
        fi
    done
}

# ── YOUTUBE : uBLOCK ORIGIN FORCE PAR POLITIQUE FIREFOX ──────────────────
# On ne veut PAS d un client tiers (il perdrait la connexion Google) : c est le
# vrai Firefox, avec uBlock Origin installe d office pour TOUS les profils via la
# politique d entreprise. `Install` pointe l XPI signe d AMO ; `installation_mode
# force_installed` l impose sans clic. Le login Google reste celui de Firefox.
#
# Firefox du .deb Mozilla lit /etc/firefox/policies/policies.json ET
# <install>/distribution/policies.json - on pose les deux pour ne pas dependre du
# chemin d installation.
_osmo_extras_firefox_ublock() {
    local pol
    read -r -d '' pol <<'POLICIES'
{
  "policies": {
    "ExtensionSettings": {
      "uBlock0@raymondhill.net": {
        "installation_mode": "force_installed",
        "install_url": "https://addons.mozilla.org/firefox/downloads/latest/ublock-origin/latest.xpi"
      }
    },
    "DisableAppUpdate": false
  }
}
POLICIES
    local d
    for d in /etc/firefox/policies \
             /usr/lib/firefox/distribution \
             /opt/firefox/distribution \
             /usr/lib/firefox-esr/distribution; do
        # /etc/firefox : toujours ; les <install>/distribution : seulement si
        # le repertoire parent (donc Firefox) existe.
        case "$d" in
            /etc/firefox/policies) mkdir -p "$d" ;;
            *) [ -d "$(dirname "$d")" ] || continue; mkdir -p "$d" ;;
        esac
        printf '%s\n' "$pol" > "$d/policies.json"
    done
    _ex_ok "uBlock Origin force par politique Firefox (login Google conserve)"
}

# ── LES LANCEURS /usr/local/bin ──────────────────────────────────────────
# Chaque lanceur accepte --fullscreen. Sans lui : fenetre normale (l encart la
# calera sur le cadre via wmctrl). Avec lui : plein ecran direct.
_osmo_extras_lanceurs() {
    # YOUTUBE : Firefox, profil dedie, sur youtube.com. Login Google OK, uBlock
    # deja force par la politique ci-dessus.
    cat > /usr/local/bin/osmo-youtube <<'YT'
#!/bin/bash
# osmo-youtube - Firefox (login Google possible) + uBlock Origin (politique),
# sur YouTube. --fullscreen => kiosque plein ecran.
set -u
URL="https://www.youtube.com"
PROFDIR="${HOME:-/root}/.mozilla/osmo-youtube"
mkdir -p "$PROFDIR"
FF="$(command -v firefox || command -v firefox-esr || true)"
[ -n "$FF" ] || { command -v zenity >/dev/null 2>&1 && \
    zenity --error --text="Firefox introuvable" 2>/dev/null; exit 1; }
if [ "${1:-}" = "--fullscreen" ]; then
    exec "$FF" --profile "$PROFDIR" --no-remote --kiosk "$URL"
```

```
else
    exec "$FF" --profile "$PROFDIR" --no-remote --new-window "$URL"
fi
YT
    chmod 755 /usr/local/bin/osmo-youtube

    # DINO jouable : le meme dino.html, mais en fenetre WebKit interactive (et
    # non l animation "attract" de osmo-dino.py). --fullscreen => plein ecran.
    cat > /usr/local/bin/osmo-dino-play <<'DINO'
#!/usr/bin/env python3
# osmo-dino-play - le jeu du dino (configs/conky/dino.html) jouable.
# Sans argument : fenetre 900x560. Avec --fullscreen : plein ecran.
import os, sys, gi
gi.require_version("Gtk", "3.0"); gi.require_version("WebKit2", "4.1")
from gi.repository import Gtk, GLib, WebKit2
REPO = os.environ.get("OSMO_REPO", "/opt/GSM/osmo-operator")
HTML = os.environ.get("OSMO_DINO_HTML", os.path.join(REPO, "configs/conky/dino.html"))
full = "--fullscreen" in sys.argv
class W(Gtk.Window):
    def __init__(self):
        super().__init__(title="Cavalier d'ombres")
        self.set_default_size(900, 560)
        self.connect("destroy", Gtk.main_quit)
        # Le bouton plein ecran de la page nous parle par un message WebKit
        # ({full:true|false}) : c est NOUS qui agrandissons la fenetre, ou
        # la rendons - une page ne peut pas le faire seule.
        ucm = WebKit2.UserContentManager()
        ucm.connect("script-message-received::osmo", self.on_message)
        ucm.register_script_message_handler("osmo")
        self.view = WebKit2.WebView.new_with_user_content_manager(ucm)
        self.add(self.view)
        if os.path.exists(HTML):
            # #full : la page part en plein ecran (le bouton de la page ramene
            # dans la fenetre). Sans --fullscreen on est dans la fenetre, point.
            self.view.load_uri(GLib.filename_to_uri(HTML, None) + ("#full" if full else ""))
        else:
            self.view.load_html("<h2 style='font-family:sans-serif'>dino.html introuvable</h2>", None)
        self.show_all()
        if full:
            self.fullscreen()
    def on_message(self, _ucm, result):
        try:
            on = result.get_js_value().to_string().find("true") >= 0 \
                 if not hasattr(result.get_js_value(), "object_get_property") \
                 else result.get_js_value().object_get_property("full").to_boolean()
        except Exception:
            on = True
        (self.fullscreen if on else self.unfullscreen)()
W(); Gtk.main()
DINO
    chmod 755 /usr/local/bin/osmo-dino-play

    # WIRESHARK EN ROOT : la capture demande les privileges (l install a mis
    # setuid a false). [2026-09-08] Le lanceur est un fichier du depot,
    # tools/osmo-wireshark-root.sh : root par pkexec (invite graphique), et
    # capture immediate avec le filtre LTE + SCTP + GSM du banc, filtre d
    # affichage 2G + 4G + telephone (appel, SMS, data) et boutons de filtre. Et
    # /usr/local/bin/wireshark pointe dessus : « wireshark » au clavier, c est
    # lui (le binaire reste /usr/bin/wireshark, qu il appelle).
    [ -f "$OSMO_EXTRAS_REPO/tools/osmo-wireshark-root.sh" ] \
        && install -m 755 "$OSMO_EXTRAS_REPO/tools/osmo-wireshark-root.sh" /usr/local/bin/osmo-wireshark-root
    ln -sfn /usr/local/bin/osmo-wireshark-root /usr/local/bin/wireshark

    # OFONO : pile telephonie. Pas d IHM native ; on ouvre un terminal qui
    # s assure que le daemon tourne (root) et liste les modems.
    cat > /usr/local/bin/osmo-ofono <<'OF'
```

```
#!/bin/bash
# osmo-ofono - demarre ofonod si besoin et liste les modems (terminal).
set -u
SUDO=""; [ "$(id -u)" -ne 0 ] && SUDO="sudo -E"
$SUDO systemctl start ofono 2>/dev/null || $SUDO ofonod 2>/dev/null &
sleep 1
if [ -x /usr/share/ofono/scripts/list-modems ]; then
    /usr/share/ofono/scripts/list-modems
else
    command -v ofonoctl >/dev/null 2>&1 && ofonoctl list || \
        echo "ofono demarre (aucun outil de liste trouve)"
fi
echo; read -n1 -rsp 'ofono - une touche pour fermer...'
OF
    chmod 755 /usr/local/bin/osmo-ofono

    # LE TELEPHONE DU BANC : la VM postmarketOS / Phosh.
    # [2026-09-07] C etait Android (Waydroid), abandonne : sans RIL, Android
    # n avait pas de pile radio et tout passait par un pont qui posait des
    # NOTIFICATIONS - une imitation d appel. postmarketOS arrive avec
    # ModemManager, Calls et Chatty, et se branche sur le VRAI modem du banc
    # (tools/osmo-phonesim-banc.py). tools/osmo-pmos.sh fait le reste.
    cat > /usr/local/bin/osmo-pmos <<'PM'
#!/bin/bash
# osmo-pmos - le telephone postmarketOS du banc (voir tools/osmo-pmos.sh).
exec "${OSMO_REPO:-/opt/GSM/osmo-operator}/tools/osmo-pmos.sh" "$@"
PM
    chmod 755 /usr/local/bin/osmo-pmos

    cat > /usr/local/bin/osmo-sms-send <<'SM'
#!/bin/bash
# osmo-sms-send <numero> <texte...> - envoie un SMS par oFono via le pont.
set -u
[ $# -ge 2 ] || { echo "usage: osmo-sms-send <numero> <texte...>"; exit 2; }
NUM="$1"; shift
# /usr/bin/python3 en dur : sous le PATH du banc, « env python3 » tombe sur
# le venv, qui n a pas les liaisons GObject (cf. start-native.sh).
exec /usr/bin/python3 "${OSMO_REPO:-/opt/GSM/osmo-operator}/tools/osmo-ofono-bridge.py" --once "sms $NUM $*"
SM
    chmod 755 /usr/local/bin/osmo-sms-send

    cat > /usr/local/bin/osmo-call <<'CA'
#!/bin/bash
# osmo-call <numero> | answer | hangup - les appels par oFono via le pont.
set -u
B="${OSMO_REPO:-/opt/GSM/osmo-operator}/tools/osmo-ofono-bridge.py"
P=/usr/bin/python3        # et pas le venv : il n a pas les liaisons GObject
case "${1:-}" in
    answer) exec "$P" "$B" --once "answer" ;;
    hangup) exec "$P" "$B" --once "hangup" ;;
    "")     echo "usage: osmo-call <numero> | answer | hangup"; exit 2 ;;
    *)      exec "$P" "$B" --once "call $1" ;;
esac
CA
    chmod 755 /usr/local/bin/osmo-call

    _ex_ok "lanceurs poses : osmo-youtube, osmo-dino-play, osmo-wireshark-root, osmo-ofono,"
    _ex_ok "                 osmo-pmos, osmo-sms-send, osmo-call"
}

# ── LES .desktop QUI MANQUENT + LES DOSSIERS « Jeux » / « Media » ──────────
# Les jeux et kodi posent DEJA leur .desktop (io.github.freedoom.Phase1/2,
# openarena.desktop, kodi.desktop). On n ajoute que ceux qui
# manquent (youtube, dino-jeu), puis on RANGE tout dans deux dossiers GNOME.
_osmo_extras_desktops() {
    install -d /usr/share/applications
```

```
    cat > /usr/share/applications/osmo-youtube.desktop <<'YTD'
[Desktop Entry]
Type=Application
Name=YouTube
Comment=YouTube dans Firefox (login Google) avec uBlock Origin
Exec=/usr/local/bin/osmo-youtube
Icon=firefox
Terminal=false
Categories=AudioVideo;Video;Network;
Keywords=youtube;video;media;
YTD

    cat > /usr/share/applications/osmo-dino-jeu.desktop <<'DIND'
[Desktop Entry]
Type=Application
Name=Cavalier d'ombres
Comment=Le jeu du dino, jouable
Exec=/usr/local/bin/osmo-dino-play
Icon=applications-games
Terminal=false
Categories=Game;ArcadeGame;
Keywords=dino;jeu;game;
DIND

    cat > /usr/share/applications/osmo-wireshark-root.desktop <<'WSD'
[Desktop Entry]
Type=Application
Name=Wireshark (banc)
GenericName=Analyseur reseau
Comment=Wireshark en root, capture immediate du banc, filtre 2G + 4G + telephone (signalisation, appel, SMS, data)
 ↪  sans les battements de coeur ; boutons Banc / 2G / 4G / Appel / SMS / Data / Attach
Exec=/usr/local/bin/osmo-wireshark-root %f
Icon=org.wireshark.Wireshark
Terminal=false
StartupNotify=true
Categories=Network;Monitor;
Keywords=wireshark;capture;pcap;reseau;lte;sctp;gsm;gsmtap;2g;4g;sms;appel;
WSD

    cat > /usr/share/applications/osmo-ofono.desktop <<'OFD'
[Desktop Entry]
Type=Application
Name=oFono
Comment=Pile telephonie oFono - demarre le daemon et liste les modems
Exec=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py --tty --title "oFono" -- /usr/local/bin/osmo-ofono
Icon=phone
Terminal=false
Categories=Network;Telephony;
Keywords=ofono;telephonie;modem;
OFD

    cat > /usr/share/applications/osmo-pmos.desktop <<'PMD'
[Desktop Entry]
Type=Application
Name=Telephone (postmarketOS)
Comment=Le telephone du banc - Phosh, ModemManager, appels et SMS reels
Exec=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py --progress --root --title "Telephone (postmarketOS)" --
 ↪  /usr/local/bin/osmo-pmos up
Icon=phone
Terminal=false
Categories=Network;Telephony;
Keywords=postmarketos;phosh;telephone;sms;appel;modem;
Actions=Etat;Arreter;

[Desktop Action Etat]
```

```
Name=Etat du telephone
Exec=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py --root --keep-open --title "Etat du telephone" --
 ↪  /usr/local/bin/osmo-pmos status

[Desktop Action Arreter]
Name=Arreter le telephone
Exec=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py --root --close-on-exit --title "Arret du telephone" --
 ↪  /usr/local/bin/osmo-pmos stop
PMD

    # [2026-09-07] Les deux lanceurs par pmbootstrap (voir update.sh,
    # osmo_poser_pmos) : la VM et son format, puis le modem et la voix.
    local ps
    for ps in qemu setup; do
        [ -f "$OSMO_EXTRAS_REPO/tools/osmo-pmos-$ps.sh" ] \
            && install -m 755 "$OSMO_EXTRAS_REPO/tools/osmo-pmos-$ps.sh" "/usr/local/bin/osmo-pmos-$ps"
    done
    # [2026-09-07] DEUX FORMATS, DEUX ICONES. Phosh se met en page d apres
    # l ecran qu on lui donne : haut et etroit, c est un telephone ; large,
    # c est une tablette - meme image, meme session. C est la carte graphique
    # de QEMU qui porte cette taille (OSMO_PMOS_RES, lu par le patch
    # pmbootstrap), et le systeme demarre dedans : le format se choisit donc
    # AU LANCEMENT et ne change pas sous une session deja ouverte. D ou deux
    # entrees plutot qu un reglage.
    local pf pn pr
    for pf in smartphone:720x1440 tablette:1280x800; do
        pn="${pf%%:*}"; pr="${pf#*:}"
        cat > "/usr/share/applications/osmo-pmos-$pn.desktop" <<PMD
[Desktop Entry]
Type=Application
Name=postmarketOS - $pn (banc)
Comment=Le telephone du banc, modem et voix branches tout seuls, en format $pn ($pr) - modem branche sur le banc
 ↪  GSM
Exec=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py --progress --title "Telephone du banc (VM)" -- env
 ↪  OSMO_PMOS_RES=$pr /usr/local/bin/osmo-pmos-qemu
Icon=$([ "$pn" = tablette ] && echo video-display || echo phone)
Terminal=false
Categories=Network;Telephony;System;
Keywords=postmarketos;pmos;qemu;modem;gsm;telephone;$pn;
Actions=Setup;SansModem;

[Desktop Action Setup]
Name=Rebrancher le modem et la voix (osmo-pmos-setup)
Exec=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py --progress --title "Modem et voix du telephone" --
 ↪  /usr/local/bin/osmo-pmos-setup

[Desktop Action SansModem]
Name=Demarrer sans modem (VM nue)
Exec=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py --progress --title "Telephone du banc (VM)" -- env
 ↪  OSMO_PMOS_RES=$pr OSMO_PMOS_MODEM=0 /usr/local/bin/osmo-pmos-qemu
PMD
        chmod 644 "/usr/share/applications/osmo-pmos-$pn.desktop"
    done

    chmod 644 /usr/share/applications/osmo-youtube.desktop \
              /usr/share/applications/osmo-dino-jeu.desktop \
              /usr/share/applications/osmo-wireshark-root.desktop \
              /usr/share/applications/osmo-ofono.desktop \
              /usr/share/applications/osmo-pmos.desktop

    # ── DOSSIERS D APPLICATIONS GNOME : « Jeux » et « Media » ──────────────
    # Un gschema override est lu a la fois par la session live (ISO) et par une
    # session installee ; c est le seul moyen qui marche AUSSI en chroot (pas de
    # dbus). On liste les .desktop attendus - un absent est ignore par GNOME,
    # comme pour les favoris du dock.
    local ov=/usr/share/glib-2.0/schemas/99-osmo-extras.gschema.override
```

```
    cat > "$ov" <<'OVR'
[org.gnome.desktop.app-folders]
folder-children=['Jeux', 'Media', 'Telephone', 'Outils']

[org.gnome.desktop.app-folders.folders:/org/gnome/desktop/app-folders/folders/Jeux/]
name='Jeux'
apps=['io.github.freedoom.Phase2.desktop', 'io.github.freedoom.Phase1.desktop', 'openarena.desktop',
 ↪  'osmo-dino-jeu.desktop']

[org.gnome.desktop.app-folders.folders:/org/gnome/desktop/app-folders/folders/Media/]
name='Media'
apps=['kodi.desktop', 'osmo-youtube.desktop']

[org.gnome.desktop.app-folders.folders:/org/gnome/desktop/app-folders/folders/Telephone/]
name='Telephone'
apps=['linphone.desktop', 'org.linphone.desktop', 'osmo-ofono.desktop', 'osmo-pmos.desktop']

[org.gnome.desktop.app-folders.folders:/org/gnome/desktop/app-folders/folders/Outils/]
name='Outils'
apps=['osmo-wireshark-root.desktop', 'org.wireshark.Wireshark.desktop', 'wireshark.desktop']
OVR
    glib-compile-schemas /usr/share/glib-2.0/schemas 2>/dev/null || true

    command -v gtk-update-icon-cache >/dev/null 2>&1 && \
        gtk-update-icon-cache -f -q /usr/share/icons/hicolor 2>/dev/null || true
    update-desktop-database /usr/share/applications 2>/dev/null || true
    _ex_ok "menus poses ; dossiers d applications « Jeux » et « Media »"
}

# ── L ENTREE ──────────────────────────────────────────────────────────
osmo_extras_install() {
    [ "$(id -u)" -eq 0 ] || { _ex_warn "osmo_extras_install demande root"; return 0; }
    if [ "${OSMO_EXTRAS_JEUX:-1}" = "1" ]; then
        _ex_say "jeux + media (Doom, Quake III, Kodi, YouTube, dino) ..."
    else
        _ex_say "media (Kodi, YouTube, dino) - jeux Doom/Quake sautes (OSMO_EXTRAS_JEUX=0) ..."
    fi
    _osmo_extras_apt
    [ "${OSMO_EXTRAS_JEUX:-1}" = "1" ] && _osmo_extras_jeux
    _osmo_extras_firefox_ublock
    _osmo_extras_lanceurs
    _osmo_extras_desktops
    echo -e "  ${GREEN}\xe2\x9c\x93${NC} jeux + media poses (dossiers ${BOLD}Jeux${NC} / ${BOLD}Media${NC}, et
 ↪  dans l encart du bureau)"
    return 0
}

# Lance directement (hors source) : ./osmo-extras-install.sh
if [ "${BASH_SOURCE[0]}" = "${0}" ]; then
    osmo_extras_install
fi
```

3.407 /opt/GSM/osmo-operator/tools/osmo-fft-snap.py

53127 octets, 1081 lignes → 1081 lignes

```
#!/usr/bin/env python3
# osmo-fft-snap.py - l ENCART VIVANT du bureau : le cadre du fond d ecran
# (tools/wallpaper-render.py, boite 510,220-1410,1010 en 1920x1080), en DEUX
# MOITIES, chacune FONDANT vers son banc une fois qu il est pret :
#
#   +-----------------------------------------------------------+
#   |  Spectre I/Q du UE srsRAN (montant) |  console srsUE      |   4G
#   |  spectre + chute d eau (ZMQ 2001)   |  (tmux srsran:ue)   |
```

```
#   +-----------------------------------------------------------+
#   |  Spectre I/Q du mobile (montant)    |  mobile.log (QEMU)  |   2G
#   |  spectre + chute d eau              |  dernieres lignes   |
#   +-----------------------------------------------------------+
#
# [2026-09-08] DEUX MOITIES, DEUX BANCS, DEUX IMAGES. Le cadre ne portait que
# la 2G ; la 4G du banc (srsENB / srsUE par ZeroMQ) n avait aucune place a
# l ecran. Le cadre a donc grandi vers le haut (410 -> 790 px) et s est coupe
# en deux : le haut est a la 4G, le bas a la 2G. Chaque moitie a SON fondu :
# tant que son banc n est pas la, elle montre la bande dessinee que le fond
# d ecran porte a cet endroit (deux images : une BD geek au hasard en haut,
# Calvin & Hobbes en bas - tools/osmo-wallpaper.sh). Un banc allume, une
# moitie vivante ; les deux, tout l encart.
#
# LA 4G S ECOUTE SANS RIEN LUI PRENDRE. La radio virtuelle de srsRAN est du
# ZeroMQ en REQ/REP : celui qui recoit DEMANDE des echantillons, et celui qui
# emet les lui REPOND - une fois. Un troisieme qui viendrait demander sur le
# port de l UE (2001) VOLERAIT des reponses a l eNB, qui n aurait plus l
# uplink entier : attach rate, et avec fail_on_disconnect sa radio s arrete.
# On n ouvre donc AUCUNE socket ZeroMQ. On lit le trafic TCP de l interface
# de bouclage en prise brute (AF_PACKET, d ou l unite en root), on suit le
# flux du port 2001 et on y decoupe les trames ZMTP : chaque reponse de l UE
# est [trame vide][trame longue = N echantillons complex64]. Rien n est
# renvoye, rien n est consomme : srsue et srsenb ne peuvent pas s en
# apercevoir. Le UE n emet que par rafales (PRACH, PUSCH...), et 11,52 MS/s
# font 92 Mo/s sur le bouclage : on ne lit qu une fenetre de 250 ms par image
# et on garde la CRETE de chaque case du spectre sur cette fenetre, sinon on
# ne verrait que du silence entre deux rafales.
#
# Source FFT 2G : le dashboard, http://127.0.0.1:8080/psd?src=ms - le meme
# JSON que l onglet FFT (vue fft1) : freqs, psd (dB), dr, arfcn. Le journal :
# le mobile.log de la pile (QEMU MS#1), lu par la queue. Cote 4G, le journal
# est la console de srsue, capturee dans le tmux « srsran » qui le porte
# (tools/osmo-lte.sh), et /tmp/ue.log a defaut.
#
#   /run/osmo-fft/panel.png   900x790   (coordonnees du fond ; osmo-panel.py
#                                        le met a l echelle de l ecran)
#
# Lance par osmo-fft-snap.service. Ne depend que de Pillow et des polices
# DejaVu, presents sur l image comme sur l hote ; numpy pour la FFT 4G (sans
# lui, la moitie haute le dit et reste sur sa bande dessinee).
import glob
import json
import os
import re
import socket
import struct
import subprocess
import sys
import threading
import time
import urllib.request

from PIL import Image, ImageDraw, ImageFont

URL = os.environ.get("OSMO_FFT_URL", "http://127.0.0.1:8080/psd")
# [2026-09-23] LE DESCENDANT EN FOND, LE MONTANT PAR-DESSUS. Le montant seul
# (src=ms) n a de signal que quand le mobile emet (RACH, canal dedie) : entre
# deux, l encart montrait un plancher plat, et la FFT paraissait sporadique. Le
# descendant (src=bts, /tmp/iq_fft.fifo que pont.py remplit) porte en continu
# la BCCH, la CCCH et donc le paging. On le trace en permanence, et le montant
# se superpose en orange des qu il porte une rafale.
# OSMO_FFT_2G=ms retablit l ancien affichage (montant seul).
FFT_2G = os.environ.get("OSMO_FFT_2G", "bts+ms")
UL_ACTIF_DB = float(os.environ.get("OSMO_FFT_UL_ACTIF_DB", "20"))
OUT = os.environ.get("OSMO_FFT_DIR", "/run/osmo-fft")
```

```
PERIOD = float(os.environ.get("OSMO_FFT_PERIOD", "1"))
FADE_S = float(os.environ.get("OSMO_FFT_FADE", "3"))
# Opacite du banc PAR-DESSUS l image du fond, une fois le fondu termine.
#
# [2026-09-04] ESSAYEE A 0.86, REMISE A 1.0. L idee etait de laisser l image du
# jour transparaitre sous le spectre. A l ecran, la bande dessinee traversait le
# journal et le spectre : les traits noirs du dessin passaient entre les lignes
# de mobile.log, et le spectre se lisait sur un fond a motifs. On perdait les
# deux - l image, hachee par le panneau, et le banc, illisible.
# Le fondu, lui, reste : c est LUI qui fait passer de l image au banc et
# retour (FADE_S), et pendant ces trois secondes les deux se superposent.
# OSMO_FFT_OPACITY=0.86 rend l ancien comportement a qui veut l essayer.
OPACITY = max(0.0, min(1.0, float(os.environ.get("OSMO_FFT_OPACITY", "1.0"))))
# Etat ecrit par tools/osmo-wallpaper.sh : y a-t-il un strip dans le fond ?
# Sans reseau, gocomics ne rend pas l image et le fond part SANS strip - le
# cadre est alors vide, et composer le banc en transparence par-dessus du vide
# ne fait que ternir le spectre et le journal. Dans ce cas l encart passe en
# opacite pleine : pas de png, mais le mobile.log et la FFT restent nets.
STRIP_STATE = os.environ.get("OSMO_WP_STATE", "/var/cache/osmo-wallpaper/strip.state")
WALLPAPER = os.environ.get("OSMO_WP_FILE", "/usr/share/backgrounds/gsm-lab-wallpaper.jpg")
# ── CE QU IL Y A SOUS L ENCART : L ILLUSTRATION ENTIERE ──────────────────
# [2026-10-01] L encart DECOUPAIT sa boite (510,220 .. 1410,1010) dans le fond
# d ecran, parce que le fond ETAIT le rendu compose (tools/wallpaper-render.py)
# avec ses deux cadres BD exactement la. Depuis le 2026-09-25 le fond par
# defaut est une image sans cadre, et le bureau GNOME porte ce que l operateur
# y met (configs/gsm-lab-fond.jpg) : decouper une boite dedans donnait un
# morceau de ciel et de colline agrandi, sans aucun sens. L encart montre donc
# l illustration du banc ENTIERE (pylones, soleil, colline), ajustee dans sa
# boite - le haut et le bas que le format laisse libres sont remplis par la
# meme image, floutee et assombrie, pour ne pas peindre deux bandes noires.
#
# Le rendu compose garde son chemin : quand osmo-wallpaper.sh a tourne et
# ecrit FOND= dans strip.state, c est CE fichier qu on decoupe, comme avant -
# ses cadres sont bien la ou la boite les attend. OSMO_FFT_FOND=decoupe force
# l ancien comportement sur $OSMO_WP_FILE.
# [2026-10-11] UNE SEULE SCENE. L illustration et le fond GNOME sortent du meme
# dessin (tools/paysage-render.py) : le fond est le paysage nu, l illustration
# est la decoupe de BOX dans ce paysage AVEC les pylones. Elle a donc deja le
# format W x H (900/790) : « entier » l ajuste sans bande, et les antennes
# semblent plantees dans le bureau par l encart. Les fichiers livres sont une
# version retouchee de ce rendu, choisie sur le banc.
ILLUSTRATION = os.environ.get("OSMO_FFT_IMAGE", "/usr/share/backgrounds/gsm-lab-wallpaper.jpg")
# [2026-10-11] PLUS D IMAGE AU CENTRE. Par defaut (« aucun ») le fond de l encart
# est TRANSPARENT : banc a l arret, on voit le bureau ; banc allume, les deux
# panneaux (spectre, journal) se posent dessus, a PANEL_ALPHA d opacite - pas
# moins, un spectre sur un paysage clair ne se lit pas. panel.png est en RGBA,
# osmo-panel.py l affiche sur une fenetre elle-meme transparente.
# OSMO_FFT_FOND=entier (l illustration) ou decoupe rendent les anciens fonds.
FOND_MODE = os.environ.get("OSMO_FFT_FOND", "aucun")
PANEL_ALPHA = max(0.0, min(1.0, float(os.environ.get("OSMO_FFT_PANEL_ALPHA", "0.8"))))
PANEL = (8, 10, 14, int(round(255 * PANEL_ALPHA)))
# ── OU EST LE JOURNAL DU MOBILE ──────────────────────────────────────────
# [2026-10-19] UN SEUL CHEMIN, PUBLIE PAR start-native.sh. mobile.log vit sous
# LOG_DIR (= RUN_DIR/logs : /run/user/<uid>/osmo-nitb/logs, ou /tmp/osmo-nitb
# sans XDG_RUNTIME_DIR - environment/paths.env), et en --dsp sous le RUNDIR de
# c54x_exe (/tmp/c54x-pont). L encart tapait /run/user/0 en dur : banc lance
# par un autre uid, et le cadre restait sur « pas encore de journal ».
# start-native.sh ecrit le chemin reel dans /run/osmo-banc/mobile-log (comme
# ms1-vty pour le port VTY) et dans l environnement systemd (OSMO_MOBILE_LOG).
# Ordre : OSMO_MOBILE_LOG force ; le fichier publie ; sinon le plus recent des
# emplacements connus ; sinon l ancien defaut. Resolu a chaque rendu : le
# journal apparait APRES le lancement de l encart.
MOBILE_LOG_CANDIDATS = ("/run/user/*/osmo-nitb/logs/mobile.log",
                        "/tmp/osmo-nitb/logs/mobile.log",
                        "/tmp/c54x-pont/mobile.log")
MOBILE_LOG_DEFAUT = "/run/user/0/osmo-nitb/logs/mobile.log"
```

```
def mobile_log_path():
    # [2026-10-24] LE FICHIER PUBLIE D ABORD. OSMO_MOBILE_LOG est fige au demarrage du service : lance
    # pendant un --ms, l encart restait sur mobile.log (vide) apres un --ms-magnetite, alors que
    # /run/osmo-banc/mobile-log, reecrit a chaque lancement, disait deja rvinterf.log.
    try:
        with open("/run/osmo-banc/mobile-log") as f:
            publie = f.read().strip()
        if publie:
            return publie
    except OSError:
        pass
    force = os.environ.get("OSMO_MOBILE_LOG")
    if force:
        return force
    vus = []
    for motif in MOBILE_LOG_CANDIDATS:
        for p in glob.glob(motif):
            try:
                vus.append((os.stat(p).st_mtime, p))
            except OSError:
                pass
    return max(vus)[1] if vus else MOBILE_LOG_DEFAUT


# L operateur choisi dans l encart (tools/osmo-panel.py, fleches) : OP=, MODE=,
# IP=, NAME=, DASH=. Absent ou natif : le dashboard et le journal locaux.
OP_FILE = os.path.join(OUT, "operator")
# La boite de l encart dans le fond d ecran (wallpaper-render.py, main()) :
# LES QUATRE FICHIERS QUI LA CONNAISSENT DOIVENT RESTER D ACCORD -
# wallpaper-render.py, osmo-fft-snap.py, osmo-panel.py, osmo-launcher.py.
BOX = (510, 220, 1410, 1010)
W, H = BOX[2] - BOX[0], BOX[3] - BOX[1]
# Les deux moities, en coordonnees de l image : (y haut, y bas). Le fond
# d ecran dessine deux cadres separes par ECART px (les deux images), et
# chaque moitie vivante recouvre exactement le sien.
ECART = 8
HH = (H - ECART) // 2
HAUT = (0, HH)
BAS = (H - HH, H)
PAD = 18
# [2026-09-04] FOOT = 0 : PLUS DE BANDE DE PIED.
# Il y avait en bas de l encart un bandeau de 30 px portant « Operateur N ·
# spectre I/Q du mobile (http://...) », et la barre de boutons se posait dessus.
# C est le « rectangle » qu on voyait en travers du bas : une bande pleine
# largeur, distincte du contenu, pour une ligne de texte qui ne disait rien que
# les boutons ne disent deja (ils portent le numero de l operateur depuis
# qu ils suivent la selection). Le contenu - spectre, chute d eau, journal -
# prend donc toute la hauteur, et il ne reste QUE les boutons, poses par-dessus
# dans le coin. FOOT reste nomme : la geometrie s exprime avec, et le remettre
# a 30 rend l ancien pied.
FOOT = 0
FONT_DIR = "/usr/share/fonts/truetype/dejavu"
ANSI = re.compile(r"\x1b\[[0-9;]*[A-Za-z]|\x07")
# ── LES COULEURS DU JOURNAL SONT DEJA DANS LE JOURNAL ──────────────────────
# mobile.log est ecrit AVEC ses codes ANSI (osmocom colorie par sous-systeme :
# « \033[0;31m<0004> gsm322.c:... \033[0;m »). L encart les jetait et
# repeignait les lignes au petit bonheur, sur des mots-cles (« error »,
# « call »...) : deux lignes du meme sous-systeme sortaient de couleurs
# differentes, et la lecture ne ressemblait plus du tout a celle du terminal.
# On rend donc la couleur que le journal porte, avec la palette Tango - celle
# du profil gnome-terminal de l image (85-installeur-bureau.sh).
SGR = re.compile(r"\x1b\[([0-9;]*)m")
TANGO = {
    "30": (85, 87, 83),    "31": (204, 0, 0),     "32": (78, 154, 6),
```

```
    "33": (196, 160, 0),   "34": (52, 101, 164),  "35": (117, 80, 123),
    "36": (6, 152, 154),   "37": (211, 215, 207),
}
TANGO_VIF = {
    "30": (128, 130, 126), "31": (239, 41, 41),   "32": (138, 226, 52),
    "33": (252, 233, 79),  "34": (114, 159, 207), "35": (173, 127, 168),
    "36": (52, 226, 226),  "37": (238, 238, 236),
}
LOG_DEFAUT = (200, 210, 225)


def couleur_ansi(brut):
    """La couleur que ce terminal donnerait a cette ligne. None si elle n en a pas."""
    for m in SGR.finditer(brut):
        codes = [c for c in m.group(1).split(";") if c]
        if not codes or codes == ["0"]:
            continue                      # remise a zero : pas une couleur
        table = TANGO_VIF if "1" in codes else TANGO
        for c in codes:
            if c in table:
                return table[c]
            if c.startswith("9") and len(c) == 2 and c[1] in "01234567":
                return TANGO_VIF["3" + c[1]]   # 90-97 : les vifs d ECMA-48
    return None


def font(name, size):
    try:
        return ImageFont.truetype(os.path.join(FONT_DIR, name), size)
    except OSError:
        return ImageFont.load_default()


F_TITLE = font("DejaVuSansMono-Bold.ttf", 15)
F_SMALL = font("DejaVuSansMono.ttf", 12)
F_LOG = font("DejaVuSansMono.ttf", 12)


def palette_sdr(v):
    """Palette SDR, 0..1 -> RGB : marine, bleu, cyan, vert, jaune, rouge (la meme que l onglet FFT, sdr).
    [2026-10-24] remplace inferno."""
    ancres = [(0,0,40), (0,20,150), (0,90,255), (0,200,230), (0,210,80), (170,235,0), (255,230,0), (255,130,0),
 ↪  (230,0,0)]
    stops = [(i / (len(ancres) - 1), c) for i, c in enumerate(ancres)]
    v = 0.0 if v < 0 else 1.0 if v > 1 else v
    for i in range(len(stops) - 1):
        a, ca = stops[i]
        b, cb = stops[i + 1]
        if v <= b:
            t = (v - a) / (b - a) if b > a else 0
            return tuple(int(ca[j] + (cb[j] - ca[j]) * t) for j in range(3))
    return stops[-1][1]


def operator():
    """{'OP','MODE','IP','NAME','DASH'} de l operateur choisi, natif par defaut."""
    op = {"OP": "1", "MODE": "native", "IP": "", "NAME": "", "DASH": URL.rsplit("/", 1)[0]}
    try:
        with open(OP_FILE) as f:
            for line in f:
                k, _, v = line.strip().partition("=")
                if k in op:
                    op[k] = v
    except OSError:
        pass
    return op
```

```
def fetch(src):
    op = operator()
    base = URL if op["MODE"] != "docker" else op["DASH"].rstrip("/") + "/psd"
    req = urllib.request.Request(f"{base}?src={src}&t={int(time.time() * 1000)}",
                                 headers={"Cache-Control": "no-store"})
    with urllib.request.urlopen(req, timeout=2) as r:
        return json.loads(r.read().decode("utf-8", "replace"))


def normalize(psd, dr):
    top = max(psd)
    floor = top - (dr if dr and dr > 0 else 40)
    span = top - floor if top > floor else 1
    return [min(1.0, max(0.0, (p - floor) / span)) for p in psd]


def resample(vals, n):
    """Ramene une liste a n colonnes (max par groupe : un pic reste visible)."""
    if len(vals) <= n:
        return vals + [vals[-1]] * (n - len(vals)) if vals else [0.0] * n
    out = []
    step = len(vals) / n
    for i in range(n):
        a, b = int(i * step), max(int((i + 1) * step), int(i * step) + 1)
        out.append(max(vals[a:b]))
    return out


_strip = {"mtime": None, "present": True, "fond": ""}


def _lire_strip_state():
    """Relit strip.state (osmo-wallpaper.sh) quand il change : STRIP= et FOND=."""
    try:
        mt = os.stat(STRIP_STATE).st_mtime
    except OSError:
        _strip.update(mtime=None, present=True, fond="")
        return
    if mt == _strip["mtime"]:
        return
    present, fond = True, ""
    try:
        with open(STRIP_STATE) as fh:
            for line in fh:
                k, _, v = line.strip().partition("=")
                if k == "STRIP":
                    present = v.strip().lower() != "non"
                elif k == "FOND":
                    fond = v.strip()
    except OSError:
        pass
    _strip.update(mtime=mt, present=present, fond=fond)


def strip_present():
    """Le fond du jour porte-t-il un strip ? (defaut : oui, on ne devine pas)"""
    _lire_strip_state()
    return _strip["present"]


def _source_fond():
    """(fichier, mode) : le rendu compose a DECOUPER, ou l illustration a AJUSTER."""
    _lire_strip_state()
    f = _strip["fond"]
```

```
    if _strip["present"] and f and os.path.isfile(f):
        return f, "decoupe"
    if FOND_MODE == "decoupe":
        return WALLPAPER, "decoupe"
    if FOND_MODE == "aucun":
        return None, "aucun"
    return (ILLUSTRATION if os.path.isfile(ILLUSTRATION) else WALLPAPER), "entier"


def _ajuster(im):
    """L image ENTIERE dans W x H : ajustee et centree ; le reste de la boite est
    la meme image qui la remplit, floutee et assombrie (pas de bandes noires)."""
    from PIL import ImageEnhance, ImageFilter
    im.thumbnail((2 * W, 2 * H), Image.LANCZOS)       # 1800x1580 (ou plus) -> de quoi travailler vite
    sw, sh = im.size
    s = max(W / sw, H / sh)
    fond = im.resize((max(W, int(sw * s + 0.5)), max(H, int(sh * s + 0.5))), Image.LANCZOS)
    x0, y0 = (fond.width - W) // 2, (fond.height - H) // 2
    fond = fond.crop((x0, y0, x0 + W, y0 + H)).filter(ImageFilter.GaussianBlur(18))
    fond = ImageEnhance.Brightness(fond).enhance(0.5)
    s = min(W / sw, H / sh)
    dev = im.resize((max(1, int(sw * s)), max(1, int(sh * s))), Image.LANCZOS)
    fond.paste(dev, ((W - dev.width) // 2, (H - dev.height) // 2))
    return fond


# ── LE FOND DE L ENCART, RELU QUAND IL CHANGE ──────────────────────────
# On suit le mtime du fichier retenu (et le fichier lui-meme : osmo-wallpaper.sh
# peut en changer le jour ou il tourne).
_base = {"cle": None, "img": None}


def base_image():
    src, mode = _source_fond()
    try:
        mt = os.stat(src).st_mtime if src else None
    except OSError:
        mt = None
    cle = (src, mode, mt)
    if cle != _base["cle"] or _base["img"] is None:
        img = None
        if mt is not None:
            try:
                wp = Image.open(src).convert("RGB")
                if mode == "decoupe":
                    if wp.size != (1920, 1080):
                        wp = wp.resize((1920, 1080), Image.LANCZOS)
                    img = wp.crop(BOX)
                else:
                    img = _ajuster(wp)
                print(f"[fft-snap] fond de l encart : {src} ({mode})", flush=True)
            except Exception as e:  # fond illisible : cadre sombre, sans strip
                print(f"[fft-snap] fond {src} : {e}", file=sys.stderr, flush=True)
        if img is None and mode == "aucun":
            img = Image.new("RGBA", (W, H), (0, 0, 0, 0))     # rien : le bureau
        elif img is None:
            img = Image.new("RGB", (W, H), (20, 24, 36))
            for m in (HAUT, BAS):     # les deux cadres, comme le fond les dessine
                ImageDraw.Draw(img).rounded_rectangle((0, m[0], W - 1, m[1] - 1), radius=26,
                                                      outline=(200, 200, 210), width=2)
        _base.update(cle=cle, img=img.convert("RGBA"))
    return _base["img"]


def base_moitie(m):
    """Le morceau de fond sous une moitie de l encart (HAUT ou BAS)."""
```

```
    return base_image().crop((0, m[0], W, m[1]))


# ── LE JOURNAL DU MOBILE ──────────────────────────────────────────────
def tail_lines(path, n, width):
    op = operator()
    if op["MODE"] == "docker":
        # [2026-09-05] `docker logs` NE MONTRE RIEN D UN OPERATEUR, exactement
        # comme pour le hub (voir HUB_LOGS plus bas, corrige le 2026-09-04) :
        # la pile est lancee par `docker exec` dans un tmux et sa sortie part
        # dans des FICHIERS, pas sur le flux standard du conteneur. Celui-ci ne
        # porte que les deux lignes de l entrypoint ("Initialisation du
        # peripherique TUN...", "Created symlink..."), et le cadre restait donc
        # fige sur elles depuis le demarrage - sur un operateur dont la radio
        # tournait parfaitement. On lit le fichier DANS le conteneur ; le chemin
        # est celui du RUN_DIR de la pile, et `docker logs` reste en dernier
        # recours pour une image qui lancerait l operateur autrement.
        data = ""
        for chemin in (path, "/tmp/osmo-nitb/logs/mobile.log"):
            try:
                out = subprocess.run(["docker", "exec", op["NAME"], "tail", "-n", str(n + 5), chemin],
                                     capture_output=True, text=True, timeout=4)
            except Exception as e:
                return [(f"{op['NAME']} : journal inaccessible ({type(e).__name__})", None)]
            if out.returncode == 0 and out.stdout.strip():
                data = out.stdout
                break
            if "permission denied" in (out.stderr or "").lower():
                return [("journal indisponible (docker : droit refuse -", None),
                        ("  le compte n est pas dans le groupe docker)", None)]
        if not data:
            try:
                r = subprocess.run(["docker", "logs", "--tail", str(n + 5), op["NAME"]],
                                   capture_output=True, text=True, timeout=3)
                data = (r.stdout + r.stderr)
            except Exception as e:
                return [(f"{op['NAME']} : journal inaccessible ({type(e).__name__})", None)]
    else:
        try:
            with open(path, "rb") as f:
                f.seek(0, 2)
                size = f.tell()
                f.seek(max(0, size - 16384))
                data = f.read().decode("utf-8", "replace")
        except OSError:
            return [(f"{os.path.basename(path)} : pas encore de journal", None)]
    # On garde la ligne BRUTE le temps d en lire la couleur, puis on la nettoie.
    brutes = [l for l in data.splitlines() if ANSI.sub("", l).strip()]
    out = []
    for brut in brutes[-n:]:
        col = couleur_ansi(brut)
        l = ANSI.sub("", brut).rstrip()
        # L horodatage osmocom (20260904135215721) mange 17 colonnes pour rien.
        l = re.sub(r"^\d{17}\s+", "", l)
        out.append((l[:width], col))
    return out or [("(journal vide)", None)]


# ── LE BANC : SPECTRE A GAUCHE, JOURNAL A DROITE ──────────────────────
history = []


# ── LA VUE DU HUB : DU SS7, PAS UNE FFT ───────────────────────────────
# [2026-09-04] Quand les fleches se posent sur l inter-STP, l encart affichait
# le meme spectre I/Q qu un operateur - c est-a-dire RIEN, et pour une raison
# de fond : un hub M3UA n a pas de radio. Il n a pas de /psd a interroger, pas
```

```
# d ARFCN, pas de mobile.log. Une FFT vide sur le noeud central du banc, c est
# l ecran le plus inutile qu on puisse afficher au moment ou l on cherche
# justement pourquoi le SS7 ne passe pas.
#
# Ce qu un hub a, en revanche : une matrice de connectivite (qui parle a qui),
# ses ASP, ses routes, et un journal. C est ce qu on montre.
#
# La matrice vient du CACHE que remplit tools/conky-osmo-status.sh
# (--refresh-matrix, depuis checks/ss7_check.sh) : la mesure prend une minute,
# elle n a rien a faire dans une boucle de rendu a 1 Hz. On lit le meme fichier
# que le Conky - une mesure, deux affichages. Les balises ${colorN} du Conky
# sont retirees ici et rendues en couleurs Pillow.
MATRIX_FILE = os.path.join(OUT, "ss7-matrix")
CONKY_TAG = re.compile(r"\$\{color[0-9]?\}")
COUL_SS7 = {"self": (88, 166, 255), "via": (63, 185, 80), "FAIL": (248, 81, 73)}


def matrice_lignes():
    """Le cache de la matrice, sans les balises Conky. [] s il n existe pas."""
    try:
        with open(MATRIX_FILE) as f:
            return [CONKY_TAG.sub("", l.rstrip("\n")) for l in f if l.strip()]
    except OSError:
        return []


# Les endroits ou le hub ecrit, dans l ordre ou on les essaie.
#
# [2026-09-04] `docker logs osmo-inter-stp` rend VIDE, et c est normal : le
# conteneur a pour commande `sleep infinity`, et osmo-stp est lance APRES par
# `docker exec` dans un tmux, sa sortie passant par un `tee` vers un fichier.
# Rien de tout cela ne traverse le flux standard du conteneur - le cadre
# « journal » restait donc desesperement vide sur un hub qui tournait tres bien.
# On lit donc le fichier, DANS le conteneur ; docker logs reste en dernier
# recours, pour une image qui lancerait le STP autrement.
HUB_LOGS = ("/tmp/osmo-stp.log", "/var/log/osmocom/osmo-stp.log")


def hub_journal(nom, n, width):
    """Les dernieres lignes du STP du hub. Sans docker (ou sans droit), on le
    dit plutot que de laisser un cadre vide - un cadre vide se lit comme une
    panne du hub, alors que c est la sonde qui n a pas le bras assez long."""
    for chemin in HUB_LOGS:
        try:
            out = subprocess.run(["docker", "exec", nom, "tail", "-n", str(n), chemin],
                                 capture_output=True, text=True, timeout=4)
        except Exception as e:
            return [(f"journal indisponible : {e}", (212, 153, 34))]
        if out.returncode == 0 and out.stdout.strip():
            return [(ANSI.sub("", l)[:width], None) for l in out.stdout.splitlines()[-n:]]
        if "permission denied" in (out.stderr or "").lower():
            return [("journal indisponible (docker : droit refuse -",
                     (212, 153, 34)), ("  le compte n est pas dans le groupe docker)", (212, 153, 34))]
    try:
        out = subprocess.run(["docker", "logs", "--tail", str(n), nom],
                             capture_output=True, text=True, timeout=4)
        lignes = (out.stdout + out.stderr).splitlines()[-n:]
        if lignes:
            return [(ANSI.sub("", l)[:width], None) for l in lignes]
    except Exception:
        pass
    return [("pas de journal : " + " ni ".join(HUB_LOGS), (139, 148, 158)),
            ("le hub tourne-t-il ?  docker ps --filter name=" + nom, (139, 148, 158))]


def render_interstp(op):
```

```
    """L encart quand la selection est l inter-STP : matrice a gauche, journal
    du hub a droite. Meme cadre, memes marges et meme pied que la vue radio -
    seul le contenu change, pour que l oeil ne perde pas ses reperes en
    passant d un operateur au hub."""
    img = base_moitie(BAS).copy()
    d = ImageDraw.Draw(img)
    # ── LE PANNEAU COUVRE TOUT LE CADRE, BORDURE COMPRISE ──────────────────
    # [2026-09-04] Le rendu peignait son contenu EN RETRAIT de PAD (18 px), et
    # ces 18 px laissaient voir base_image() - c est-a-dire le morceau de fond
    # d ecran, donc le `glass_panel` que wallpaper-render.py y a peint : un
    # remplissage plus clair (20,24,36) et surtout une BORDURE (200,200,210).
    # A l ecran, cela dessinait un cadre clair arrondi tout autour de l encart -
    # « le rectangle ». Il ne venait ni de la barre GTK (dont le fond avait deja
    # ete supprime) ni du pied (dont la couleur avait deja ete alignee) : il
    # etait dans le fond d ecran, et l encart le laissait passer par ses marges.
    #
    # Quand le banc tourne, l encart est un panneau plein : il couvre TOUT le
    # cadre, meme rayon que celui du fond (26) pour tomber exactement dessus.
    # Le cadre du fond reste visible quand le banc est a l arret - c est alors
    # l image du jour qu il encadre, et la il a un sens.
    d.rounded_rectangle((0, 0, W - 1, HH - 1), radius=26, fill=PANEL)
    x0, y0, x1, y1 = PAD, PAD, W - PAD, HH - PAD - FOOT
    d.rounded_rectangle((x0 - 6, y0 - 6, x1 + 6, y1 + 6), radius=8, fill=PANEL)
    split = x0 + int((x1 - x0) * 0.56)

    d.text((x0, y0), "Matrice de connectivite  ·  via inter-STP", font=F_TITLE, fill=(88, 166, 255))
    tag = f"PC {os.environ.get('MULTI_HUB_PC', '0.0.0')}"
    d.text((split - 12 - d.textlength(tag, font=F_SMALL), y0 + 2), tag, font=F_SMALL, fill=(63, 185, 80))

    lignes = matrice_lignes()
    yy = y0 + 30
    if not lignes:
        d.text((x0, yy), "premiere mesure en cours (checks/ss7_check.sh)...",
               font=F_SMALL, fill=(139, 148, 158))
    for ligne in lignes:
        xx = x0
        # Chaque mot est peint a sa couleur : self/via/FAIL portent le sens.
        for mot in re.split(r"(\s+)", ligne):
            if not mot:
                continue
            col = COUL_SS7.get(mot.strip(), (200, 208, 220))
            if mot.strip().startswith("SS7"):
                col = (63, 185, 80) if "OK" in ligne else (212, 153, 34)
            d.text((xx, yy), mot, font=F_LOG, fill=col)
            xx += d.textlength(mot, font=F_LOG)
        yy += 17
        if yy > y1 - 14:
            break

    # Le journal du hub, a droite, exactement ou vit celui du mobile.
    lx0 = split
    d.line((lx0 - 6, y0, lx0 - 6, y1), fill=(30, 36, 48))
    d.text((lx0, y0), f"journal  ·  {op['NAME']}", font=F_TITLE, fill=(88, 166, 255))
    line_h = 15
    n = max(1, (y1 - (y0 + 26)) // line_h)
    cols = max(10, int((x1 - lx0) / 7.3))
    for i, (l, col) in enumerate(hub_journal(op["NAME"], n, cols)):
        if col is None:
            col = LOG_DEFAUT
            if re.search(r"error|fail|reject|lost|down", l, re.I):
                col = TANGO_VIF["31"]
            elif re.search(r"ASP|AS |active|established|route", l, re.I):
                col = TANGO_VIF["32"]
        d.text((lx0, y0 + 26 + i * line_h), l, font=F_LOG, fill=col)

    return img
```

```
def render_live(data, data_ul=None):
    img = base_moitie(BAS).copy()
    d = ImageDraw.Draw(img)
    # ── LE PANNEAU COUVRE TOUT LE CADRE, BORDURE COMPRISE ──────────────────
    # [2026-09-04] Le rendu peignait son contenu EN RETRAIT de PAD (18 px), et
    # ces 18 px laissaient voir base_image() - c est-a-dire le morceau de fond
    # d ecran, donc le `glass_panel` que wallpaper-render.py y a peint : un
    # remplissage plus clair (20,24,36) et surtout une BORDURE (200,200,210).
    # A l ecran, cela dessinait un cadre clair arrondi tout autour de l encart -
    # « le rectangle ». Il ne venait ni de la barre GTK (dont le fond avait deja
    # ete supprime) ni du pied (dont la couleur avait deja ete alignee) : il
    # etait dans le fond d ecran, et l encart le laissait passer par ses marges.
    #
    # Quand le banc tourne, l encart est un panneau plein : il couvre TOUT le
    # cadre, meme rayon que celui du fond (26) pour tomber exactement dessus.
    # Le cadre du fond reste visible quand le banc est a l arret - c est alors
    # l image du jour qu il encadre, et la il a un sens.
    d.rounded_rectangle((0, 0, W - 1, HH - 1), radius=26, fill=PANEL)
    x0, y0, x1, y1 = PAD, PAD, W - PAD, HH - PAD - FOOT
    d.rounded_rectangle((x0 - 6, y0 - 6, x1 + 6, y1 + 6), radius=8, fill=PANEL)
    split = x0 + int((x1 - x0) * 0.56)
    # Spectre
    titre = ("Spectre I/Q du mobile  ·  montant (UL)" if FFT_2G == "ms"
             else "Spectre I/Q  ·  DL + UL")
    d.text((x0, y0), titre, font=F_TITLE, fill=(88, 166, 255))
    arfcn = data.get("arfcn", "?") if data else "?"
    tag = f"ARFCN {arfcn}"
    d.text((split - 12 - d.textlength(tag, font=F_SMALL), y0 + 2), tag, font=F_SMALL, fill=(63, 185, 80))
    sx0, sx1 = x0, split - 12
    sw = sx1 - sx0
    py0 = y0 + 24
    psd_h = 110
    py1 = py0 + psd_h
    d.rectangle((sx0, py0, sx1, py1), fill=(12, 16, 24))
    for gy in range(1, 4):
        yy = py0 + gy * psd_h // 4
        d.line((sx0, yy, sx1, yy), fill=(24, 30, 40))
    if data and "psd" in data:
        vals = resample(normalize(data["psd"], data.get("dr", 40)), sw)
        pts = [(sx0 + x, py1 - 1 - int(v * (psd_h - 4))) for x, v in enumerate(vals)]
        for x, y in pts:
            d.line((x, y, x, py1 - 1), fill=(30, 70, 110))
        d.line(pts, fill=(88, 210, 255), width=1)
        ul = data_ul.get("psd") if data_ul else None
        if ul and max(ul) - min(ul) >= UL_ACTIF_DB:
            # Rafale montante : sa courbe par-dessus, et dans la chute d eau le
            # max des deux, pour qu un RACH ou un bloc SDCCH y laisse sa trace.
            vul = resample(normalize(ul, data_ul.get("dr", 40)), sw)
            d.line([(sx0 + x, py1 - 1 - int(v * (psd_h - 4))) for x, v in enumerate(vul)],
                   fill=(255, 166, 60), width=1)
            vals = [max(a, b) for a, b in zip(vals, vul)]
        history.insert(0, vals)
    else:
        d.text((sx0 + 8, py0 + 8), "pas de flux", font=F_SMALL, fill=(248, 81, 73))
    # Chute d eau : une ligne par image, la plus recente en haut
    wy0 = py1 + 4
    wf_h = y1 - wy0
    del history[wf_h:]
    wf = Image.new("RGB", (sw, wf_h), (8, 10, 14))
    px = wf.load()
    for row, line in enumerate(history):
        for x in range(min(sw, len(line))):
            px[x, row] = palette_sdr(line[x])
    img.paste(wf, (sx0, wy0))
```

```
    # Journal
    d = ImageDraw.Draw(img)
    lx0 = split
    d.line((lx0 - 6, y0, lx0 - 6, y1), fill=(30, 36, 48))
    op = operator()
    # [2026-10-23] le nom du fichier lu : rvinterf.log (traces TI) avec --ms-magnetite
    d.text((lx0, y0), (f"{os.path.basename(mobile_log_path())}  ·  QEMU MS#1" if op["MODE"] != "docker"
                       else f"journal  ·  {op['NAME']}"), font=F_TITLE, fill=(88, 166, 255))
    line_h = 15
    n = max(1, (y1 - (y0 + 26)) // line_h)
    cols = max(10, int((x1 - lx0) / 7.3))
    for i, (l, col) in enumerate(tail_lines(mobile_log_path(), n, cols)):
        if col is None:
            # Journal sans couleurs (docker logs, redirection qui les a
            # mangees) : on retombe sur la lecture par mots-cles.
            col = LOG_DEFAUT
            if re.search(r"error|fail|reject|lost", l, re.I):
                col = TANGO_VIF["31"]
            elif re.search(r"attach|answer|call|SMS|proceed", l, re.I):
                col = TANGO_VIF["32"]
        d.text((lx0, y0 + 26 + i * line_h), l, font=F_LOG, fill=col)
    # ── LE PIED COUVRE JUSQU AU BAS, SANS RIEN LAISSER DEPASSER ──────────────
    # [2026-09-04] La barre s arretait a H - PAD + 4, et le commentaire d alors
    # l assumait : « la legende du strip est dessous ». Elle l etait, et ELLE SE
    # VOYAIT - une quinzaine de pixels de « NASA · Astronomy Picture of the Day
    # · 2026-09-04 · apod.nasa.gov », coupee en deux par le bord de l encart,
    # sous le spectre et le mobile.log. Une ligne orpheline en travers du
    # bureau, que rien n expliquait.
    #
    # Le panneau est construit SUR une copie du fond d ecran (base_image) : tout
    # ce que le rendu ne peint pas laisse voir le fond. On peint donc jusqu au
    # bord. Le rayon reste, mais le bas du rectangle sort du cadre (H + 8) :
    # l arrondi tombe hors de l image et le bas est franc, comme le bord de
    # l encart lui-meme.
    # [2026-09-04] LE PIED N EST PLUS D UNE AUTRE COULEUR QUE LE CORPS.
    # Il etait peint en (20,24,36) sous un corps en (8,10,14) : mesure sur
    # panel.png, c est une marche de douze niveaux, et elle dessinait une bande
    # claire en travers du bas de l encart - le « rectangle » qu on voyait
    # derriere les boutons, et qu on prenait pour un fond de la barre GTK. Ce
    # n en etait pas un : le fond de la barre avait deja ete supprime, la marche
    # etait dans l image. Meme couleur des deux cotes : un seul panneau.
    # Plus de legende : elle vivait dans la bande de pied, supprimee (FOOT = 0).
    # Ce qu elle disait - quel operateur, quel journal - est porte par les
    # boutons eux-memes (« Dashboard op2 », « VTY 4247 op2 ») et par le titre
    # de la colonne de droite.
    return img


# ── LA 4G : LE UE srsRAN, ECOUTE EN PRISE BRUTE SUR LE BOUCLAGE ──────────────
# Voir l entete : pas de socket ZeroMQ, on lit le TCP du port de l UE tel
# qu il passe sur lo et on y decoupe les trames ZMTP. Reglages :
#   OSMO_LTE_TAP_PORT    le port ZMQ « tx » de srsue (2001 ; 2000 = l eNB,
#                        donc le descendant, pour qui prefere le voir)
#   OSMO_LTE_TAP_IFACE   l interface (lo)
#   OSMO_LTE_TAP_WINDOW  la fenetre d ecoute par image, en s (0.25)
#   OSMO_SRSRAN_UE_CONF  le ue.conf a lire pour l EARFCN et la cadence
#   OSMO_SRSRAN_TMUX     le tmux (-L) qui porte srsue, fenetre « ue »
#   OSMO_SRSRAN_UE_LOG   le journal de srsue, a defaut du tmux
LTE_PORT = int(os.environ.get("OSMO_LTE_TAP_PORT", "2001"))
LTE_IFACE = os.environ.get("OSMO_LTE_TAP_IFACE", "lo")
LTE_FENETRE = float(os.environ.get("OSMO_LTE_TAP_WINDOW", "0.25"))
LTE_NFFT = 1024
LTE_UE_CONF = os.environ.get("OSMO_SRSRAN_UE_CONF", "/root/.config/srsran/ue.conf")
LTE_TMUX = os.environ.get("OSMO_SRSRAN_TMUX", "srsran")
# [2026-09-09] tools/osmo-lte.sh lance srsue avec --log.filename=/tmp/osmo-lte-ue.log
# (qui prime sur le « filename = /tmp/ue.log » du ue.conf) et sa SORTIE STANDARD
```

```
# dans /tmp/osmo-lte-ue.console - c est la que sont « Found PLMN », « RRC
# Connected », « Network attach successful ». Le tmux -L srsran n existe que
# si on l a ouvert a la main. Chercher /tmp/ue.log donnait donc toujours
# « console srsue introuvable » alors que le UE etait attache.
LTE_UE_LOG = os.environ.get("OSMO_SRSRAN_UE_LOG", "/tmp/osmo-lte-ue.log")
LTE_UE_CONSOLE = os.environ.get("OSMO_SRSRAN_UE_CONSOLE", "/tmp/osmo-lte-ue.console")
LTE_UE_SOURCES = [LTE_UE_CONSOLE, LTE_UE_LOG, "/tmp/ue.log"]
# Le marqueur d une reponse ZMTP de l UE : la trame vide du REQ/REP (0x01 =
# « il y en a une autre », taille 0), puis l en-tete d une trame longue (0x02,
# taille sur 8 octets grand-boutiste, dont les cinq premiers sont nuls tant
# qu elle fait moins de 2^24 octets - une trame d une seconde en ferait 92 M).
ZMTP_MARQUE = b"\x01\x00\x02\x00\x00\x00\x00\x00"
try:
    import numpy as np
except ImportError:      # la moitie 4G le dira ; la 2G n en a pas besoin
    np = None


def _tcp_ecoute(port):
    """Quelqu un ecoute-t-il ce port TCP ? (lecture de /proc/net/tcp, sans privilege)"""
    for f in ("/proc/net/tcp", "/proc/net/tcp6"):
        try:
            with open(f) as fh:
                next(fh)
                for l in fh:
                    c = l.split()
                    if c[3] == "0A" and int(c[1].rsplit(":", 1)[1], 16) == port:
                        return True
        except (OSError, StopIteration, ValueError, IndexError):
            pass
    return False


def srsran_ue_conf():
    """{'earfcn', 'srate'} d apres ue.conf ; des defauts raisonnables sinon."""
    conf = {"earfcn": "?", "srate": 11.52e6}
    try:
        section = ""
        with open(LTE_UE_CONF) as fh:
            for l in fh:
                l = l.strip()
                if l.startswith("["):
                    section = l
                elif l.startswith("dl_earfcn") and section == "[rat.eutra]":
                    conf["earfcn"] = l.split("=", 1)[1].strip()
                elif l.startswith("device_args"):
                    m = re.search(r"base_srate=([0-9.eE+]+)", l)
                    if m:
                        conf["srate"] = float(m.group(1))
    except (OSError, ValueError):
        pass
    return conf


class SondeLte(threading.Thread):
    """Ecoute la moitie 4G : etat = {psd (dB, LTE_NFFT cases, centre au milieu),
    crete (amplitude max vue), trames, erreur, t}. Ne tourne que si le port
    de l UE est ouvert - sinon rien a ecouter, et rien a ouvrir."""

    def __init__(self):
        super().__init__(daemon=True)
        self.etat = {"psd": None, "crete": 0.0, "trames": 0, "erreur": None, "t": 0.0}
        self.hann = (np.hanning(LTE_NFFT).astype(np.float32) if np is not None else None)

    def run(self):
        while True:
```

```
        if np is None:
            self.etat = dict(self.etat, psd=None, erreur="numpy absent : pas de FFT 4G")
            time.sleep(10)
            continue
        if not _tcp_ecoute(LTE_PORT):
            self.etat = dict(self.etat, psd=None, trames=0, erreur=None)
            time.sleep(PERIOD)
            continue
        try:
            self.capture()
        except PermissionError:
            self.etat = dict(self.etat, psd=None, erreur="prise brute refusee : root (CAP_NET_RAW) requis")
            time.sleep(10)
        except Exception as e:
            self.etat = dict(self.etat, psd=None, erreur=f"sonde : {type(e).__name__}: {e}")
            time.sleep(2)
        time.sleep(max(0.1, PERIOD - LTE_FENETRE))

def capture(self):
    s = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.htons(0x0003))
    try:
        s.bind((LTE_IFACE, 0))
        s.settimeout(0.2)
        s.setsockopt(socket.SOL_SOCKET, socket.SO_RCVBUF, 4 << 20)
        buf, attendu, crete_psd, crete, trames = b"", None, None, 0.0, 0
        fin = time.time() + LTE_FENETRE
        while time.time() < fin:
            try:
                pkt, adr = s.recvfrom(70000)
            except socket.timeout:
                continue
            # Sur lo chaque paquet passe deux fois (sortant puis entrant) :
            # on ne garde que l entrant (pkttype 4 = PACKET_OUTGOING).
            if adr[2] == 4 or len(pkt) < 34 or pkt[12:14] != b"\x08\x00" or pkt[23] != 6:
                continue
            tcp = 14 + (pkt[14] & 0xF) * 4
            sport, _dport, seq = struct.unpack("!HHI", pkt[tcp:tcp + 8])
            if sport != LTE_PORT:
                continue
            charge = pkt[tcp + (pkt[tcp + 12] >> 4) * 4:]
            if not charge:
                continue
            if attendu is not None and seq != attendu:
                buf = b""            # trou dans le flux : on resynchronise
            attendu = (seq + len(charge)) & 0xFFFFFFFF
            buf += charge
            while True:
                i = buf.find(ZMTP_MARQUE)
                if i < 0:
                    buf = buf[-len(ZMTP_MARQUE):]
                    break
                taille = int.from_bytes(buf[i + 3:i + 11], "big")
                if taille % 8 or taille > (64 << 20):
                    buf = buf[i + 1:]            # faux marqueur dans les donnees
                    continue
                if len(buf) < i + 11 + taille:
                    buf = buf[i:]
                    break
                x = np.frombuffer(buf[i + 11:i + 11 + taille], dtype=np.complex64)
                buf = buf[i + 11 + taille:]
                n = len(x) // LTE_NFFT
                if n == 0:
                    continue
                trames += 1
                blocs = x[:n * LTE_NFFT].reshape(n, LTE_NFFT) * self.hann
                p = np.abs(np.fft.fftshift(np.fft.fft(blocs, axis=1), axes=1)) ** 2
```

```
                    p = p.max(axis=0) / (LTE_NFFT * LTE_NFFT)
                    crete_psd = p if crete_psd is None else np.maximum(crete_psd, p)
                    crete = max(crete, float(np.max(np.abs(x))))
        finally:
            s.close()
        if trames == 0:
            self.etat = dict(self.etat, psd=None, trames=0, erreur=None, t=time.time())
        else:
            self.etat = {"psd": (10 * np.log10(crete_psd + 1e-20)).tolist(), "crete": crete,
                         "trames": trames, "erreur": None, "t": time.time()}


def _queue_fichier(chemin, n):
    """Les n dernieres lignes non vides d un fichier texte, sans couleurs ANSI ;
    [] s il est absent ou vide."""
    try:
        with open(chemin, "rb") as f:
            f.seek(0, 2)
            f.seek(max(0, f.tell() - 262144))
            data = f.read().decode("utf-8", "replace")
    except OSError:
        return []
    # Les mesures RRC (« MEAS: New measurement serving cell ») tombent toutes
    # les 20 ms : elles noieraient l attach. On ne garde que les evenements.
    # Derniere ligne encore en cours d ecriture : on ne la montre pas tronquee.
    if data and not data.endswith("\n"):
        data = data[:data.rfind("\n") + 1]
    # srsue ecrit ses barres de progression au \r : seul le dernier segment compte.
    lignes = [ANSI.sub("", l.split("\r")[-1]).rstrip() for l in data.splitlines()]
    return [l for l in lignes if l.strip() and " MEAS:" not in l][-n:]


def srsran_journal(n, width):
    """Les dernieres lignes de la console de srsue : la fenetre « ue » du tmux
    qui le porte si elle existe, sinon sa sortie standard (osmo-lte-ue.console :
    PLMN trouve, RRC, attach) COMPLETEE par son journal (osmo-lte-ue.log :
    RRC/NAS au niveau info, cf. configs/srsran/ue.conf)."""
    try:
        out = subprocess.run(["tmux", "-L", LTE_TMUX, "capture-pane", "-p", "-t", f"{LTE_TMUX}:ue"],
                             capture_output=True, text=True, timeout=2)
        lignes = [ANSI.sub("", l).rstrip() for l in out.stdout.splitlines()]
        lignes = [l for l in lignes if l.strip()]
        if out.returncode == 0 and lignes:
            return [(l[:width], None) for l in lignes[-n:]]
    except Exception:
        pass
    # Console d abord (l essentiel tient en quelques lignes), le journal prend
    # le reste de la place : les deux tronques par le bas, les plus recents.
    console = _queue_fichier(LTE_UE_CONSOLE, max(1, n // 3))
    journal = _queue_fichier(LTE_UE_LOG, n)
    if not console and not journal:
        journal = _queue_fichier("/tmp/ue.log", n)
    if console or journal:
        lignes = console + journal[-max(0, n - len(console)):] if journal else console
        return [(l[:width], None) for l in lignes[-n:]]
    return [("console srsue introuvable :", (139, 148, 158)),
            (f"  ni tmux -L {LTE_TMUX} (fenetre ue), ni " + ", ni ".join(LTE_UE_SOURCES),
             (139, 148, 158))]


historique_lte = []


def render_lte(etat):
    """La moitie haute : le spectre montant du UE srsRAN a gauche, sa console a
    droite. Meme cadre, memes marges que la 2G : un seul encart, deux bancs."""
```

```
img = base_moitie(HAUT).copy()
d = ImageDraw.Draw(img)
d.rounded_rectangle((0, 0, W - 1, HH - 1), radius=26, fill=PANEL)
x0, y0, x1, y1 = PAD, PAD, W - PAD, HH - PAD - FOOT
d.rounded_rectangle((x0 - 6, y0 - 6, x1 + 6, y1 + 6), radius=8, fill=PANEL)
split = x0 + int((x1 - x0) * 0.56)
conf = srsran_ue_conf()
d.text((x0, y0), "Spectre I/Q du UE srsRAN  ·  UL", font=F_TITLE, fill=(88, 166, 255))
tag = f"EARFCN {conf['earfcn']} · ZMQ {LTE_PORT}"
d.text((split - 12 - d.textlength(tag, font=F_SMALL), y0 + 2), tag, font=F_SMALL, fill=(63, 185, 80))
sx0, sx1 = x0, split - 12
sw = sx1 - sx0
py0 = y0 + 24
psd_h = 110
py1 = py0 + psd_h
d.rectangle((sx0, py0, sx1, py1), fill=(12, 16, 24))
for gy in range(1, 4):
    yy = py0 + gy * psd_h // 4
    d.line((sx0, yy, sx1, yy), fill=(24, 30, 40))
# L axe : la cadence, de -srate/2 a +srate/2 autour de la porteuse.
demi = conf["srate"] / 2e6
for k, frac in ((f"-{demi:.2f}", 0.0), ("0", 0.5), (f"+{demi:.2f} MHz", 1.0)):
    tx = sx0 + int(frac * sw)
    tx = min(max(tx - d.textlength(k, font=F_SMALL) * frac, sx0), sx1 - d.textlength(k, font=F_SMALL))
    d.text((tx, py1 + 2), k, font=F_SMALL, fill=(90, 100, 120))
psd = etat.get("psd")
if etat.get("erreur"):
    d.text((sx0 + 8, py0 + 8), etat["erreur"], font=F_SMALL, fill=(248, 81, 73))
    vals = None
elif psd is None:
    d.text((sx0 + 8, py0 + 8), "pas de flux (l eNB ne demande rien a l UE)", font=F_SMALL,
           fill=(248, 81, 73))
    vals = None
else:
    # Echelle : le haut suit la crete, jamais sous -60 dB - ainsi le
    # silence entre deux rafales reste a plat au lieu d etre etire.
    haut = max(max(psd), -60.0)
    vals = resample([min(1.0, max(0.0, (p - (haut - 50)) / 50)) for p in psd], sw)
    pts = [(sx0 + x, py1 - 1 - int(v * (psd_h - 4))) for x, v in enumerate(vals)]
    for x, y in pts:
        d.line((x, y, x, py1 - 1), fill=(30, 70, 110))
    d.line(pts, fill=(88, 210, 255), width=1)
    if etat.get("crete", 0.0) < 1e-6:
        d.text((sx0 + 8, py0 + 8), "silence : le UE n emet pas (veille)", font=F_SMALL,
               fill=(139, 148, 158))
historique_lte.insert(0, vals if vals is not None else [0.0] * sw)
wy0 = py1 + 18
wf_h = y1 - wy0
del historique_lte[wf_h:]
wf = Image.new("RGB", (sw, wf_h), (8, 10, 14))
px = wf.load()
for row, line in enumerate(historique_lte):
    for x in range(min(sw, len(line))):
        px[x, row] = palette_sdr(line[x])
img.paste(wf, (sx0, wy0))
d = ImageDraw.Draw(img)
lx0 = split
d.line((lx0 - 6, y0, lx0 - 6, y1), fill=(30, 36, 48))
d.text((lx0, y0), "console srsUE  ·  osmo-lte-ue.console + .log", font=F_TITLE, fill=(88, 166, 255))
line_h = 15
n = max(1, (y1 - (y0 + 26)) // line_h)
cols = max(10, int((x1 - lx0) / 7.3))
for i, (l, col) in enumerate(srsran_journal(n, cols)):
    if col is None:
        col = LOG_DEFAUT
        if re.search(r"reject|fail|error|release|lost|timeout", l, re.I):
```

```
                col = TANGO_VIF["31"]
            elif re.search(r"attach|found (plmn|cell)|complete|connected|network attach", l, re.I):
                col = TANGO_VIF["32"]
            d.text((lx0, y0 + 26 + i * line_h), l, font=F_LOG, fill=col)
    return img


def write(img):
    os.makedirs(OUT, exist_ok=True)
    tmp = os.path.join(OUT, ".panel.tmp.png")
    img.save(tmp, "PNG")
    os.replace(tmp, os.path.join(OUT, "panel.png"))


def main():
    print(f"[fft-snap] {URL} + {mobile_log_path()} (2G, bas) + prise brute {LTE_IFACE}:{LTE_PORT} (4G, haut)"
          f" -> {OUT}/panel.png toutes les {PERIOD}s (fondu {FADE_S}s depuis {WALLPAPER})", flush=True)
    sonde = SondeLte()
    sonde.start()
    # Un fondu par moitie : « bas » suit la 2G (ou le hub), « haut » la 4G.
    alpha = {"bas": 0.0, "haut": 0.0}
    step = PERIOD / FADE_S if FADE_S > 0 else 1.0
    last_state = {"bas": None, "haut": None}
    while True:
        t = time.time()
        op_courant = operator()
        # Le hub n a pas de /psd : l interroger ne rendrait jamais rien et
        # l encart resterait sur le strip. Sa vue est « prete » des qu il est
        # selectionne - c est la matrice et son journal qui font le contenu.
        if op_courant.get("MODE") == "interstp":
            data, data_ul, ready = None, None, True
        else:
            data = data_ul = None
            try:
                data = fetch("ms" if FFT_2G == "ms" else "bts")
            except Exception:
                data = None
            if FFT_2G != "ms":
                try:
                    data_ul = fetch("ms")
                except Exception:
                    data_ul = None
                # Pas de descendant (pont sans FIFO DL) : le montant seul, comme avant.
                if not (data and "psd" in data) and data_ul and "psd" in data_ul:
                    data, data_ul = data_ul, None
            ready = bool(data) and "psd" in data
        # La 4G est « la » des que srsue tient son port ZMQ : le spectre peut
        # etre vide (pas de rafale), mais l encart montre alors ce silence,
        # avec la console de l UE a cote - ce qu on veut voir quand on attend
        # l attach, justement.
        prets = {"bas": ready, "haut": _tcp_ecoute(LTE_PORT)}
        for m, nom in (("bas", "banc 2G"), ("haut", "banc 4G (srsue)")):
            if prets[m] != last_state[m]:
                print(f"[fft-snap] {nom} {'pret : fondu vers le spectre' if prets[m] else 'absent : retour a la
                ↪  bande dessinee'}",
                      flush=True)
                last_state[m] = prets[m]
            target = 1.0 if prets[m] else 0.0
            alpha[m] = (min(target, alpha[m] + step) if target > alpha[m]
                        else max(target, alpha[m] - step))
        try:
            img = base_image().copy()
            # TOUJOURS un blend, meme a fondu termine : c est ce qui laisse
            # le strip transparaitre sous le banc. Sans strip derriere, rien
            # a laisser voir : opacite pleine.
            op = OPACITY if strip_present() else 1.0
```

```
            if alpha["bas"] <= 0.0:
                history.clear()
            else:
                live = (render_interstp(op_courant)
                        if op_courant.get("MODE") == "interstp" else render_live(data, data_ul))
                img.paste(Image.blend(base_moitie(BAS), live, alpha["bas"] * op), (0, BAS[0]))
            if alpha["haut"] <= 0.0:
                historique_lte.clear()
            else:
                live = render_lte(sonde.etat)
                img.paste(Image.blend(base_moitie(HAUT), live, alpha["haut"] * op), (0, HAUT[0]))
            write(img)
        except Exception as e:
            print(f"[fft-snap] rendu : {e}", file=sys.stderr, flush=True)
        time.sleep(max(0.2, PERIOD - (time.time() - t)))


if __name__ == "__main__":
    main()
```

3.408 /opt/GSM/osmo-operator/tools/osmo-firmwares-install.sh

23511 octets, 367 lignes → 367 lignes

```
#!/bin/bash
# ============================================================================
# osmo-firmwares-install.sh - LES FIRMWARES TELEPHONE HORS ISO (supplements)
#
# [2026-10-23] Source par addition.sh (--gnuarm, --magnetite, --trx, --fernvale,
# --simcom). Chaque fonction est IDEMPOTENTE (un arbre deja clone n est pas
# reclone, un binaire deja sorti n est pas refait sans OSMO_FW_REFAIRE=1) et
# rien n est fatal : un miroir absent ne doit pas arreter les autres supplements.
#
#   osmo_gnuarm_install     la chaine historique d osmocom-bb : arm-elf-gcc 4.5.2,
#                           TIREE de l image Docker osmocom-bb-trx (pas recompilee)
#                           dans /root/gnuarm, et le lien libmpfr.so.4 -> libmpfr.so.6
#                           que son cc1 reclame
#   osmo_magnetite_install  FreeCalypso Magnetite pour le Motorola C139 (le vrai
#                           firmware TI) : Wine + environnement TI TMS470, sources
#                           git figees, patchs du banc, image flash de 4 MiB
#   osmo_calypso_fw_install le firmware Calypso du banc (layer1 GPRS, rssi...)
#                           recompile avec gnuarm depuis osmocom-bb eac4eed1 + patchs
#   osmo_trx_install        osmocom-bb-trx : l image Docker publiee (cote hote,
#                           ubuntu 16.04) et le firmware trx (TX actif) compile
#                           avec gnuarm
#   osmo_fernvale_install   MediaTek MT6260 (Fernvale) : fernly, fernvale-nuttx,
#                           osmocom-bb-mtk-layer1, avec arm-none-eabi
#   osmo_simcom_install     la documentation et les SDK SIMCom (~3,4 Go)
# ============================================================================

: "${GREEN:=}"; : "${YELLOW:=}"; : "${RED:=}"; : "${CYAN:=}"; : "${BOLD:=}"; : "${NC:=}"
: "${GSM_ROOT:=/opt/GSM}"
: "${REPO:=${DIR:-$(cd "$(dirname "${BASH_SOURCE[0]}")/.." && pwd)}}"
: "${OSMO_FW_JOBS:=$(nproc 2>/dev/null || echo 4)}"
OSMO_FW_LOG="${OSMO_FW_LOG:-/var/log/osmo-firmwares}"

_fw_ok()   { echo -e "  ${GREEN}✓${NC} $*"; }
_fw_warn() { echo -e "  ${YELLOW}!${NC} $*"; }
_fw_err()  { echo -e "  ${RED}✗${NC} $*"; }
_fw_log()  { mkdir -p "$OSMO_FW_LOG" 2>/dev/null; echo "$OSMO_FW_LOG/$1.log"; }

# apt sans question, une seule fois par execution du supplement
_fw_apt() {
    if [ -z "${_FW_APT_UPDATE:-}" ]; then apt-get update -qq >/dev/null 2>&1 || true; _FW_APT_UPDATE=1; fi
```

```
    DEBIAN_FRONTEND=noninteractive apt-get install -y -q "$@" >>"$(_fw_log apt)" 2>&1
}

# git clone si absent ; rien sinon (on ne touche pas a un arbre de travail existant)
_fw_clone() {   # url dest [branche]
    local url="$1" dest="$2" br="${3:-}"
    if [ -d "$dest/.git" ]; then _fw_ok "$(basename "$dest") deja present ($dest)"; return 0; fi
    git clone -q ${br:+-b "$br"} "$url" "$dest" >>"$(_fw_log git)" 2>&1 \
        && _fw_ok "$(basename "$dest") clone ($url${br:+ $br})" \
        || { _fw_err "clone de $url impossible (voir $(_fw_log git))"; return 1; }
}

# [2026-10-23] LE BUG .map D OSMOCOM-BB. Le fichier .map est demande a l edition
# de liens sous deux formes, cassees toutes deux des que le lien passe par le
# pilote gcc ($(CC)) :
#   « -Map board/... »       (jolly/testing, MTK) : gcc lit -Map comme -M -ap ;
#   « -Wl,-Map board/... »   (osmocom-bb amont, eac4eed1) : -Map part a ld mais le
#                            NOM part a gcc, comme fichier d entree : « board/
#                            compal_e88/hello_world.highram.map : No such file ».
# Les deux deviennent -Wl,-Map,<fichier> (virgule). Par $(LD) directement (gnuarm
# sur jolly/testing) « -Map fichier » est juste : on n y touche pas. Le script
# solve_envs.py, lui, est en Python 2 (print sans parentheses) : sans python2 sur
# l hote, aucun environnement n est resolu et le firmware sort sans .bin, en silence.
_fw_bb_corriger() {   # racine/src/target/firmware
    local fw="$1" mk="$1/Makefile.inc"
    if [ -f "$mk" ] && grep -q '^\s*-Wl,-Map board' "$mk"; then
        sed -i 's|^\(\s*\)-Wl,-Map board/|\1-Wl,-Map,board/|' "$mk"
        _fw_ok "Makefile.inc : -Wl,-Map <fichier> -> -Wl,-Map,<fichier>"
    fi
    if [ -f "$mk" ] && grep -q '^\s*-Map board' "$mk" \
       && grep -B2 '^\s*-Map board' "$mk" | grep -q '\$(CC)'; then
        sed -i 's|^\(\s*\)-Map board/\$(2)/\$(1)\.\$(3)\.map |\1-Wl,-Map,board/$(2)/$(1).$(3).map |' "$mk"
        _fw_ok "Makefile.inc : -Map -> -Wl,-Map,<fichier> (lien par gcc)"
    fi
    if [ -f "$fw/solve_envs.py" ] && grep -q "^\s*print ' '" "$fw/solve_envs.py"; then
        sed -i "1s|python\$|python3|; s|^\(\s*\)print ' '.join(envs)|\1print(' '.join(envs))|" "$fw/solve_envs.py"
        _fw_ok "solve_envs.py passe en Python 3"
    fi
}

# ── gnuarm ──────────────────────────────────────────────────────────
# [2026-10-24] LA CHAINE N EST PLUS RECONSTRUITE. gnu-arm-installer (binutils,
# gcc 4.5.2, newlib 1.19 depuis les sources) prenait de longues minutes et
# dependait de miroirs qui disparaissent (newlib a deja quitte sources.redhat.com).
# L image osmocom-bb-trx, deja tiree pour --trx, porte la meme chaine, construite
# une fois pour toutes : /root/gnuarm/install. On la copie hors de l image
# (docker create + docker cp, sans lancer de conteneur) vers $GNUARM_DIR/install.
# Ses binaires viennent d Ubuntu 16.04 : sur noble il ne leur manque que
# libmpfr.so.4, le lien pose plus bas. Teste : compilation et lien d un .c
# avec newlib (-mcpu=arm7tdmi) par arm-elf-gcc 4.5.2.
GNUARM_DIR="${GNUARM_DIR:-/root/gnuarm}"
OSMOCOM_BB_TRX_IMAGE="${OSMOCOM_BB_TRX_IMAGE:-ghcr.io/bbaranoff/osmocom-bb-trx:main}"
osmo_gnuarm_install() {
    echo -e "${BOLD}== gnuarm (arm-elf-gcc 4.5.2, image $OSMOCOM_BB_TRX_IMAGE) ==${NC}"
    if [ ! -x "$GNUARM_DIR/install/bin/arm-elf-gcc" ]; then
        if ! command -v docker >/dev/null 2>&1 || ! docker info >/dev/null 2>&1; then
           _fw_err "docker absent ou arrete : gnuarm vient de l image $OSMOCOM_BB_TRX_IMAGE (addition.sh --docker)"
            return 1
        fi
        local log c; log="$(_fw_log gnuarm)"
        docker pull "$OSMOCOM_BB_TRX_IMAGE" >>"$log" 2>&1 \
            || { _fw_err "docker pull $OSMOCOM_BB_TRX_IMAGE en echec (voir $log)"; return 1; }
        c="$(docker create "$OSMOCOM_BB_TRX_IMAGE" 2>>"$log")" \
            || { _fw_err "docker create en echec (voir $log)"; return 1; }
        mkdir -p "$GNUARM_DIR"
        rm -rf "$GNUARM_DIR/install"
```

```
        docker cp -q "$c:/root/gnuarm/install" "$GNUARM_DIR/install" >>"$log" 2>&1
        local rc=$?
        docker rm -f "$c" >/dev/null 2>&1
        [ "$rc" = 0 ] || { _fw_err "docker cp de /root/gnuarm/install en echec (voir $log)"; return 1; }
        _fw_ok "gnuarm copie depuis l image ($GNUARM_DIR/install, $(du -sh "$GNUARM_DIR/install" | cut -f1))"
    fi
    # [2026-10-23] cc1 de gcc 4.5.2 est lie a libmpfr.so.4, que noble ne fournit plus
    # (libmpfr.so.6, meme ABI pour ce qu il en fait) : sans ce lien, « cc1: error while
    # loading shared libraries: libmpfr.so.4 ».
    if [ ! -e /usr/local/lib/libmpfr.so.4 ]; then
        local m6; m6="$(ls /usr/lib/*-linux-gnu/libmpfr.so.6 2>/dev/null | head -1)"
        if [ -n "$m6" ]; then ln -sf "$m6" /usr/local/lib/libmpfr.so.4 && ldconfig && _fw_ok "lien libmpfr.so.4 ->
         ↪  $m6"
        else _fw_warn "libmpfr.so.6 introuvable : apt install libmpfr6"; fi
    fi
    if echo 'int main(void){return 0;}' | "$GNUARM_DIR/install/bin/arm-elf-gcc" -x c -c -o /dev/null -
     ↪  2>/dev/null; then
        _fw_ok "arm-elf-gcc $("$GNUARM_DIR/install/bin/arm-elf-gcc" -dumpversion) operationnel
         ↪  ($GNUARM_DIR/install/bin)"
    else
        _fw_err "arm-elf-gcc ne compile pas (cc1, libmpfr ?)"; return 1
    fi
    # [2026-10-24] $GNUARM_DIR/install/bin dans le PATH de tout shell de connexion
    # (scripts/osmo-gnuarm-profile.sh -> /etc/profile.d/97-osmo-gnuarm.sh)
    if [ -f "${REPO:-/opt/GSM/osmo-operator}/scripts/osmo-gnuarm-profile.sh" ]; then
        install -m644 "${REPO:-/opt/GSM/osmo-operator}/scripts/osmo-gnuarm-profile.sh"
         ↪  /etc/profile.d/97-osmo-gnuarm.sh \
            && _fw_ok "PATH : $GNUARM_DIR/install/bin (/etc/profile.d/97-osmo-gnuarm.sh, nouveaux shells)"
    fi
}

# ── Magnetite (C139) ──────────────────────────────────────────────────
# [2026-10-24] PLUS RIEN DE freecalypso.org : son FTP plafonne a ~18 Ko/s (52 Mo
# d environnement TI = 45 min). github.com/bbaranoff/fc-magnetite porte l historique
# hg converti (pointe 9394305d4ff5, l amont sous les patchs du banc) et, dans
# freecalypso-env/, l environnement Wine, le bootloader Compal et freecalypso-tools.
FC_MAGNETITE_GIT="${FC_MAGNETITE_GIT:-https://github.com/bbaranoff/fc-magnetite}"
FC_ENV_URL="${FC_ENV_URL:-https://raw.githubusercontent.com/bbaranoff/fc-magnetite/main/freecalypso-env}"
osmo_magnetite_install() {
    echo -e "${BOLD}== FreeCalypso Magnetite (C139) : Wine + TI TMS470 ==${NC}"
    local fcm="$GSM_ROOT/fc-magnetite" fht="$GSM_ROOT/fc-host-tools" wdl="$GSM_ROOT/fc-wine-dl" \
          bdl="$GSM_ROOT/fc-dl" out="$GSM_ROOT/firmware/magnetite" log; log="$(_fw_log magnetite)"
    # Wine 32 bits : le TMS470 (cl470, asm470, lnk470) est un binaire Windows 32 bits
    dpkg --print-foreign-architectures | grep -qx i386 || dpkg --add-architecture i386
    _FW_APT_UPDATE=""
    _fw_apt git wine wine64 wine32:i386 xz-utils wget build-essential \
        || { _fw_err "installation de Wine en echec (voir $(_fw_log apt))"; return 1; }
    _fw_ok "wine $(wine --version 2>/dev/null)"
    # les sources : Magnetite (le depot git, freecalypso-env/ compris) et les outils
    # hote FreeCalypso (rvinterf, fc-shell, fc-fsio : run.sh en a besoin). Un ancien
    # clone hg reste tel quel : l environnement se prend alors par FC_ENV_URL.
    if [ ! -d "$fcm/.hg" ] && [ ! -d "$fcm/.git" ]; then
        git clone -q "$FC_MAGNETITE_GIT" "$fcm" >>"$log" 2>&1 \
            || { _fw_err "git clone fc-magnetite en echec (voir $log)"; return 1; }
        _fw_ok "fc-magnetite clone ($FC_MAGNETITE_GIT)"
    fi
    # _fc_env <fichier> <destination> : la copie du clone, sinon le depot sur GitHub
    _fc_env() {
        [ -s "$2" ] && return 0
        if [ -s "$fcm/freecalypso-env/$1" ]; then cp "$fcm/freecalypso-env/$1" "$2"
        else wget -q -O "$2" "$FC_ENV_URL/$1" >>"$log" 2>&1 || { rm -f "$2"; return 1; }; fi
    }
    if [ ! -f "$fht/Makefile" ]; then
        mkdir -p "$fht" "$wdl"
        _fc_env freecalypso-tools.tar.xz "$wdl/freecalypso-tools.tar.xz" \
            && tar xJf "$wdl/freecalypso-tools.tar.xz" -C "$fht" --strip-components=1 >>"$log" 2>&1 \
```

```
            || _fw_warn "freecalypso-tools : extraction en echec"
    fi
    [ -x "$fht/rvinterf/lowlevel/rvinterf" ] || make -C "$fht" -j"$OSMO_FW_JOBS" >>"$log" 2>&1 \
        || _fw_warn "outils hote FreeCalypso : make en echec (voir $log)"
    # les patchs du banc (ARFCN force, enregistrement auto, SMS, MMI) : un seul fichier,
    # regenere par `hg diff` dans fc-magnetite (patches/fc-magnetite-banc.patch)
    local p="$REPO/patches/fc-magnetite-banc.patch"
    if [ -f "$p" ]; then
        if (cd "$fcm" && patch -p1 -R --dry-run -s -f <"$p" >/dev/null 2>&1); then _fw_ok "patchs du banc deja
         ↪  appliques"
        elif (cd "$fcm" && patch -p1 -s -f <"$p" >>"$log" 2>&1); then _fw_ok "patchs du banc appliques ($(basename
         ↪  "$p"))"
        else _fw_warn "$(basename "$p") ne s applique pas sur cet arbre (voir $log)"; fi
    fi
    # l environnement TI installe (compilateur, cygwin, perl, java) dans ~/.wine/drive_c
    mkdir -p "$wdl" "$bdl" "$out"
    local f
    for f in installed-env.tar.xz nowhine.c winebuild.sh README; do
        _fc_env "$f" "$wdl/$f" || _fw_warn "telechargement $f en echec"
    done
    if [ ! -d "$HOME/.wine/drive_c/Dvpt_tools" ]; then
        WINEDEBUG=-all wineboot -i >>"$log" 2>&1
        tar xJf "$wdl/installed-env.tar.xz" -C "$HOME/.wine/drive_c" >>"$log" 2>&1 \
            && _fw_ok "environnement TI pose dans $HOME/.wine/drive_c" \
            || { _fw_err "environnement TI : extraction en echec"; return 1; }
        WINEDEBUG=-all wineboot -u >>"$log" 2>&1
    fi
    [ -x /usr/local/bin/nowhine ] || gcc -O2 -o /usr/local/bin/nowhine "$wdl/nowhine.c" >>"$log" 2>&1 \
        || _fw_warn "nowhine : compilation en echec"
    # la compilation : c139, hybrid-ui-vo. SANS -j : le TMS470 sous Wine ecrit des
    # fichiers temporaires a noms fixes, deux cl470 en parallele se marchent dessus.
    # [2026-10-23] LA VARIANTE DU BANC : DWNLD 0 (FC_MAGNETITE_DWNLD), dossier -nopatch. Le
    # patch DSP telecharge par la L1 TI (DWNLD 1) echouait sous l emulation (LA_LoadPage :
    # course sur LA_TIMEOUT) ; le banc tourne depuis le 21/10 sur ce build-la, verifie
    # octet pour octet contre c139-flash.img. L1_DYN_DSP_DWNLD reste a 1 (a 0,
    # l1audio_async.c ne compile pas). FC_MAGNETITE_DWNLD=1 : le build d origine.
    local dwnld="${FC_MAGNETITE_DWNLD:-0}" suf="" b
    [ "$dwnld" = 0 ] && suf="-nopatch"
    b="$fcm/build-c139-hybrid-ui-vo$suf"
    if [ ! -s "$b/fwimage.bin" ] || [ "${OSMO_FW_REFAIRE:-0}" = 1 ]; then
        ( cd "$fcm" && (cd helpers && make) && ./configure.sh --clean c139 hybrid-ui-vo ${suf:+SUFFIX=$suf} \
            && sed -i "s/^#define DWNLD [01]\$/#define DWNLD $dwnld/" "$b/config/swconfig.cfg" \
            && cd "$b" && WINEDEBUG=-all make ) \
            >>"$log" 2>&1 || { _fw_err "compilation Magnetite en echec (voir $log)"; return 1; }
    fi
    _fw_ok "fwimage.bin (DWNLD $dwnld) : $(stat -c %s "$b/fwimage.bin") octets, $b"
    # l image flash de 4 MiB : bootloader Compal FreeCalypso a 0, fwimage.bin a 0x10000,
    # FFS a 0x3C0000 (IMEI, afcparams, rfcap ecrits par fc-fsio). Une image EXISTANTE ne
    # change que sa zone firmware : bootloader et FFS du banc restent tels quels (la zone
    # du bootloader de l image du banc differe de 4 octets du fichier FreeCalypso, au
    # vecteur 0x10, depuis le 22/10, d origine inconnue ; le banc demarre avec). Copie
    # de l ancienne image en .avant-supplement.
    _fc_env compal-flash-boot-for-fc.bin "$bdl/compal-flash-boot-for-fc.bin" \
        || _fw_warn "bootloader Compal : telechargement en echec"
    [ -f "$out/c139-flash.img" ] && cp -p "$out/c139-flash.img" "$out/c139-flash.img.avant-supplement"
    python3 - "$bdl/compal-flash-boot-for-fc.bin" "$b/fwimage.bin" "$out/c139-flash.img" <<'EOF' \
        && _fw_ok "image flash $out/c139-flash.img" || _fw_err "assemblage de l image flash en echec"
import os, sys
boot, fw, img_path = sys.argv[1:4]
f = open(fw, "rb").read()
if os.path.exists(img_path) and os.path.getsize(img_path) == 4 << 20:
    img = bytearray(open(img_path, "rb").read())              # bootloader + FFS du banc gardes
    img[0x10000:0x3C0000] = b"\xff" * (0x3C0000 - 0x10000)
else:
    img = bytearray(b"\xff" * (4 << 20))
    b = open(boot, "rb").read(); img[0:len(b)] = b
```

```
img[0x10000:0x10000 + len(f)] = f
open(img_path, "wb").write(img)
EOF
    echo -e "  ${CYAN}→${NC} lancer : ${BOLD}./start.sh --bts-classique --ms-magnetite --gapk${NC} (appels :
     ↪  ENCRYPTION=\"a5 0\" et --regen)"
}

# ── osmocom-bb-trx ──────────────────────────────────────────────────
osmo_trx_install() {
    echo -e "${BOLD}== osmocom-bb-trx (jolly/testing, TX) ==${NC}"
    # cote hote (layer23 --enable-transceiver, libosmocore 1.1.0) : l image publiee
    # (github.com/bbaranoff/osmocom-bb-trx, ubuntu 16.04 + gcc 4.9/5 + gnuarm)
    if command -v docker >/dev/null 2>&1 && docker info >/dev/null 2>&1; then
        if docker pull "$OSMOCOM_BB_TRX_IMAGE" >>"$(_fw_log trx)" 2>&1; then _fw_ok "image $OSMOCOM_BB_TRX_IMAGE"
        else _fw_warn "docker pull $OSMOCOM_BB_TRX_IMAGE en echec (voir $(_fw_log trx))"; fi
    else
        _fw_warn "docker absent ou arrete : l image osmocom-bb-trx n est pas tiree (addition.sh --docker)"
    fi
    # le firmware trx, avec la chaine gnuarm, dans un arbre a part (celui du banc,
    # osmocom-bb-transceiver, garde ses patchs et n est pas touche)
    osmo_gnuarm_install || return 1
    local t="$GSM_ROOT/osmocom-bb-trx" log; log="$(_fw_log trx)"
    _fw_apt autoconf automake libtool pkg-config python3 git   # libosmocore cote cible
    _fw_clone https://gitea.osmocom.org/phone-side/osmocom-bb "$t" jolly/testing || return 1
    sed -i 's/^#CFLAGS += -DCONFIG_TX_ENABLE/CFLAGS += -DCONFIG_TX_ENABLE/' "$t/src/target/firmware/Makefile"
    _fw_bb_corriger "$t/src/target/firmware"
    if [ ! -s "$t/src/target/firmware/board/compal_e88/trx.highram.bin" ] || [ "${OSMO_FW_REFAIRE:-0}" = 1 ]; then
        ( cd "$t/src" && PATH="$GNUARM_DIR/install/bin:$PATH" make -j"$OSMO_FW_JOBS" firmware ) >>"$log" 2>&1 \
            || { _fw_err "firmware trx en echec (voir $log)"; return 1; }
    fi
    if [ -s "$t/src/target/firmware/board/compal_e88/trx.highram.bin" ]; then
        _fw_ok "firmware trx (TX) : $t/src/target/firmware/board/compal_e88/trx.{highram,e88loader,e88flash}.bin"
    else
        _fw_err "trx.highram.bin absent apres make (voir $log)"; return 1
    fi
}

# ── firmware Calypso du banc (layer1, rssi...) avec gnuarm ──────────
# Le firmware livre est celui du depot bbaranoff/firmware (/opt/GSM/firmware,
# board/compal_e88/{layer1,rssi,trx}.highram). Celui-ci le RECOMPILE depuis l arbre
# osmocom-bb du banc : eac4eed1 + les patchs de patches/ (dont
# osmocom-bb-firmware-gprs.patch, le layer1 GPRS), avec arm-elf-gcc 4.5.2.
# layer1.compalram deborde de LRAM (3448 octets) avec le GPRS et gcc 4.5 : il n est
# pas utilise par le banc (highram), d ou make -k. OSMO_FW_INSTALLER=1 recopie les
# .highram.{elf,bin} dans /opt/GSM/firmware (anciens en .avant-<date>).
OSMOCOM_BB_REF="${OSMOCOM_BB_REF:-eac4eed1}"
osmo_calypso_fw_install() {
    echo -e "${BOLD}== firmware Calypso du banc (osmocom-bb, gnuarm) ==${NC}"
    osmo_gnuarm_install || return 1
    local t="$GSM_ROOT/osmocom-bb" fw log p; log="$(_fw_log calypso-fw)"
    _fw_apt autoconf automake libtool pkg-config python3 git
    if [ ! -d "$t/.git" ]; then
        _fw_clone https://gitea.osmocom.org/phone-side/osmocom-bb "$t" || return 1
        git -C "$t" checkout -q "$OSMOCOM_BB_REF" && _fw_ok "osmocom-bb epingle sur $OSMOCOM_BB_REF"
    fi
    for p in "$REPO"/patches/osmocom-bb-*.patch; do
        case "$p" in *transceiver*) continue ;; esac     # ceux-la visent osmocom-bb-transceiver
        if git -C "$t" apply -R --check "$p" 2>/dev/null; then :
        elif git -C "$t" apply --check "$p" 2>/dev/null; then git -C "$t" apply "$p" && _fw_ok "patch $(basename
         ↪  "$p")"
        else _fw_warn "$(basename "$p") ne s applique pas sur cet arbre"; fi
    done
    fw="$t/src/target/firmware"
    _fw_bb_corriger "$fw"
    ( cd "$t/src" && PATH="$GNUARM_DIR/install/bin:$PATH" make libosmocore-target \
        && PATH="$GNUARM_DIR/install/bin:$PATH" make -k -j"$OSMO_FW_JOBS" -C target/firmware \
```

```
        CROSS_COMPILE=arm-elf- BOARDS=compal_e88 ) >>"$log" 2>&1
    local b="$fw/board/compal_e88" ok=1 a
    for a in layer1 rssi; do
        if [ -s "$b/$a.highram.bin" ]; then _fw_ok "$a.highram.bin ($b)"; else _fw_err "$a.highram.bin absent
        ↪ (voir $log)"; ok=0; fi
    done
    if [ "$ok" = 1 ] && [ "${OSMO_FW_INSTALLER:-0}" = 1 ]; then
        local dst="$GSM_ROOT/firmware/board/compal_e88" d; d="$(date +%Y%m%d-%H%M)"
        mkdir -p "$dst"
        for a in layer1 rssi; do
            for x in elf bin; do
                [ -f "$dst/$a.highram.$x" ] && cp -p "$dst/$a.highram.$x" "$dst/$a.highram.$x.avant-$d"
                cp -f "$b/$a.highram.$x" "$dst/"
            done
        done
        _fw_ok "pose dans $dst (anciens en .avant-$d)"
    elif [ "$ok" = 1 ]; then
        echo -e "  ${CYAN}→${NC} le banc garde le firmware livre ; OSMO_FW_INSTALLER=1 pour poser celui-ci dans
        ↪ $GSM_ROOT/firmware"
    fi
    [ "$ok" = 1 ]
}

# ── Fernvale (MT6260) ─────────────────────────────────────────────────
osmo_fernvale_install() {
    echo -e "${BOLD}== Fernvale (MediaTek MT6260) ==${NC}"
    local log; log="$(_fw_log fernvale)"
    _fw_apt gcc-arm-none-eabi binutils-arm-none-eabi libnewlib-arm-none-eabi build-essential \
            libusb-1.0-0-dev genromfs python3 autoconf automake libtool pkg-config git \
        || _fw_warn "paquets arm-none-eabi : installation incomplete (voir $(_fw_log apt))"
    local fl="$GSM_ROOT/fernly" nx="$GSM_ROOT/fernvale-nuttx" mtk="$GSM_ROOT/osmocom-bb-mtk-layer1"
    _fw_clone https://github.com/bbaranoff/fernly "$fl"
    _fw_clone https://github.com/bbaranoff/fernvale-nuttx "$nx"
    _fw_clone https://github.com/bbaranoff/osmocom-bb-mtk-layer1 "$mtk" mtk-layer1
    # fernly : l OS de rétro-ingenierie et son chargeur USB (fernly-usb-loader, natif)
    if [ -d "$fl" ]; then
        if ( cd "$fl" && make CROSS_COMPILE=arm-none-eabi- ) >>"$log" 2>&1; then
            _fw_ok "fernly : $fl/build/firmware.bin + fernly-usb-loader"
        else _fw_err "fernly : make en echec (voir $log)"; fi
        [ -f "$fl/95-fernvale-test.rules" ] && install -D -m644 "$fl/95-fernvale-test.rules"
        ↪ /etc/udev/rules.d/95-fernvale-test.rules \
            && udevadm control --reload 2>/dev/null; true
    fi
    # NuttX (kosagi-fernvale/nsh) : les correctifs de compilation sont deja dans le depot
    if [ -d "$nx/nuttx" ]; then
        if [ ! -s "$nx/nuttx/nuttx.bin" ] || [ "${OSMO_FW_REFAIRE:-0}" = 1 ]; then
            ( cd "$nx/nuttx/tools" && ./configure.sh kosagi-fernvale/nsh && cd .. \
                && make CROSSDEV=arm-none-eabi- ARCROSSDEV=arm-none-eabi- ) >>"$log" 2>&1 \
                || _fw_err "NuttX : make en echec (voir $log)"
        fi
        [ -s "$nx/nuttx/nuttx.bin" ] && _fw_ok "NuttX : $nx/nuttx/nuttx.bin"
    fi
    # osmocom-bb layer1 MTK (bringup + layer1 MT6260)
    if [ -d "$mtk/src" ]; then
        _fw_bb_corriger "$mtk/src/target/firmware"
        if ( cd "$mtk/src" && make mtk-firmware CROSS_HOST=arm-none-eabi ) >>"$log" 2>&1 \
           && [ -s "$mtk/src/target/firmware/board/fernvale/layer1.mt6260.bin" ]; then
            _fw_ok "layer1 MTK : $mtk/src/target/firmware/board/fernvale/{bringup,layer1}.mt6260.bin"
        else _fw_err "layer1 MTK : make mtk-firmware en echec (voir $log)"; fi
    fi
    echo -e "  ${CYAN}→${NC} charger : ${BOLD}$fl/build/fernly-usb-loader -s /dev/fernvale
    ↪ $fl/build/usb-loader.bin <payload.bin>${NC}"
}

# ── SIMCom ────────────────────────────────────────────────────────────
osmo_simcom_install() {
```

```
    echo -e "${BOLD}== SIMCom : documentation et SDK ==${NC}"
    # ~3,4 Go (archives rar des SDK, rapports) : un clone sans historique
    local d="$GSM_ROOT/simcom"
    if [ -d "$d/.git" ]; then _fw_ok "simcom deja present ($d)"; return 0; fi
    git clone -q --depth 1 https://github.com/bbaranoff/simcom "$d" >>"$(_fw_log git)" 2>&1 \
        && _fw_ok "simcom clone dans $d ($(du -sh "$d" 2>/dev/null | cut -f1))" \
        || _fw_err "clone de bbaranoff/simcom en echec"
    _fw_apt p7zip-full unrar 2>/dev/null || true   # les SDK sont en .rar / .7z
}
```

3.409 /opt/GSM/osmo-operator/tools/osmo-fond-antenne.py

3966 octets, 92 lignes → 92 lignes

```
#!/usr/bin/env python3
# Variante « banc en marche » du fond : sans le bandeau ubuntu|osmocom ni le NO RF,
# pour laisser la place à la bannière et au Conky. Seuls les pixels clairs des
# logos / du panneau sont effacés (le fond derrière est sombre) ; on les remplit
# par un flou normalisé des voisins sombres, plus un léger grain.
#
# [2026-10-23] Entree : la sortie de tools/osmo-fond-repos.py (2560x1600) ; les
# zones effacees sont en coordonnees de cette image. Voir l en-tete de
# osmo-fond-repos.py pour la chaine complete.
import sys
import numpy as np
from PIL import Image, ImageFilter

SRC, DST = sys.argv[1], sys.argv[2]
im = np.asarray(Image.open(SRC).convert("RGB"), float)
H, W, _ = im.shape
yy, xx = np.mgrid[0:H, 0:W]


def flou(a, s):
    return np.stack([np.asarray(Image.fromarray(np.clip(a[..., c], 0, 255).astype(np.uint8))
                                .filter(ImageFilter.GaussianBlur(s)), float) for c in range(3)], -1)


def flou_m(m, s):
    return np.asarray(Image.fromarray((np.clip(m, 0, 1) * 255).astype(np.uint8))
                      .filter(ImageFilter.GaussianBlur(s)), float) / 255


def dilate(m, n):
    return np.asarray(Image.fromarray((m * 255).astype(np.uint8)).filter(ImageFilter.MaxFilter(n))) > 0


lum = im.max(axis=2)
clair = lum > 55

# zone 1 : bandeau ubuntu | osmocom | GSM...LTE - seuls les pixels clairs
zone = np.zeros((H, W), bool); zone[115:355, 175:1630] = True
trou = dilate(clair & zone, 23)
# zone 2 : panneau NO RF (disque du cercle barré + texte + trait) - en entier,
# le fond orange derriere le texte ne se distingue pas du trait
cx, cy, r = 2205, 285, 268
norf = (xx - cx) ** 2 + (yy - cy) ** 2 < r ** 2
norf[475:700, 2045:2415] = True
norf = dilate(norf, 9)

# remplissage : moyenne des pixels sombres voisins, à plusieurs échelles
garde = (~trou).astype(float)
rempli = np.zeros_like(im); poids = np.zeros((H, W))
for s, seuil in ((6, .15), (14, .15), (30, .15), (60, .1), (120, .03), (250, 0.003)):
```

```
    num = flou(im * garde[..., None], s)
    den = flou_m(garde, s)
    ok = (den > seuil) & (poids == 0)
    rempli[ok] = num[ok] / den[ok][:, None]
    poids[ok] = 1

# panneau NO RF : membrane lisse (diffusion depuis le bord, a 1/8), sans les
# paliers des echelles de flou, puis les coups de pinceau d une zone 690 px a
# gauche (bandeau deja efface, tour de la ville evitee), estompes vers le bas
def membrane(img, trou, f=8, n=400):
    h, w = H // f, W // f
    p = np.stack([np.asarray(Image.fromarray(img[..., c].astype(np.uint8)).resize((w, h), Image.BOX), float)
                  for c in range(3)], -1)
    t = np.asarray(Image.fromarray((trou * 255).astype(np.uint8)).resize((w, h), Image.BOX)) > 0
    q = p.copy(); q[t] = p[~t].mean(axis=0)
    for _ in range(n):
        v = (np.roll(q, 1, 0) + np.roll(q, -1, 0) + np.roll(q, 1, 1) + np.roll(q, -1, 1)) / 4
        q[t] = v[t]
    return np.stack([np.asarray(Image.fromarray(np.clip(q[..., c], 0, 255).astype(np.uint8))
                                .resize((W, H), Image.BICUBIC), float) for c in range(3)], -1)

# grain : un bruit léger, flouté d un pixel, pour ne pas laisser d aplat lisse
rng = np.random.default_rng(23)
bruit = rng.normal(0, 5, (H, W))
bruit = np.asarray(Image.fromarray(np.clip(bruit + 128, 0, 255).astype(np.uint8))
                   .filter(ImageFilter.GaussianBlur(0.8)), float) - 128
rempli = rempli + bruit[..., None] * np.array([1.0, 0.6, 1.0])

m = flou_m(trou.astype(float), 3)
m = np.maximum(m, trou)[..., None]
out = im * (1 - m) + rempli * m

lisse = membrane(out, norf)
don = np.roll(out, 690, axis=1)
coups = don - flou(don, 22)
attenue = np.clip((660 - yy) / 80, 0, 1)[..., None]
remp2 = lisse + 0.8 * coups * attenue + bruit[..., None] * np.array([1.0, 0.6, 1.0])
m2 = flou_m(dilate(norf, 21).astype(float), 10)
m2 = np.maximum(m2, norf)[..., None]
out = out * (1 - m2) + remp2 * m2
Image.fromarray(np.clip(out, 0, 255).astype(np.uint8)).save(DST, optimize=True)
print(DST)
```

3.410 /opt/GSM/osmo-operator/tools/osmo-fond-bleu.py

9264 octets, 163 lignes → 163 lignes

```
#!/usr/bin/env python3
# Fond « Telecom Operator » en bleu : les coups de pinceau orange du fond passent
# en dégradé violet sombre -> bleu clair (selon la luminosité du pixel).
# Restent orange / rouge : le logo ubuntu, OPERATOR, le panneau NO RF barré et le
# schéma téléphones -> BTS -> BSC -> MSC -> Core.
#
# [2026-10-23] Entree : la sortie de tools/osmo-fond-repos.py (2560x1600) ; les
# zones protegees sont en coordonnees de cette image. La chaine devient :
#   tools/osmo-fond-repos.py configs/gsm-lab-source.png /tmp/repos.png
#   tools/osmo-fond-bleu.py /tmp/repos.png /tmp/bleu.png
#   tools/osmo-fond-antenne.py /tmp/bleu.png /tmp/antenne.png
#
# Sans argument : toute la chaine d un coup, depuis configs/gsm-lab-source.png
# jusqu a configs/gsm-lab-ville.jpg et configs/gsm-lab-antenne.jpg.
#
# [2026-10-24] OSMO_FOND_ORANGE_HAUT=1 : la variante « banc en marche » (celle
# sans le NO RF) melange les deux, l orange d origine en haut qui se fond en
```

```
# degrade dans le bleu vers le bas. La chaine sans argument l applique a
# gsm-lab-antenne.jpg seulement ; gsm-lab-ville.jpg (au repos) reste bleu.
import os
import subprocess
import sys
import tempfile
import numpy as np
from PIL import Image, ImageDraw, ImageFilter, ImageFont

if len(sys.argv) == 1:
    outils = os.path.dirname(os.path.abspath(__file__))
    configs = os.path.join(os.path.dirname(outils), "configs")
    with tempfile.TemporaryDirectory() as tmp:
        repos, bleu, mix, antenne = (os.path.join(tmp, n) for n in ("repos.png", "bleu.png", "mix.png",
 ↪  "antenne.png"))
        for script, src, dst, env in (("osmo-fond-repos.py", os.path.join(configs, "gsm-lab-source.png"), repos,
         ↪  {}),
                                      ("osmo-fond-bleu.py", repos, bleu, {"OSMO_FOND_ORANGE_HAUT": "0"}),
                                      ("osmo-fond-bleu.py", repos, mix, {"OSMO_FOND_ORANGE_HAUT": "1"}),
                                      ("osmo-fond-antenne.py", mix, antenne, {})):
            subprocess.run([sys.executable, os.path.join(outils, script), src, dst], check=True,
                           env={**os.environ, **env})
        for src, nom in ((bleu, "gsm-lab-ville.jpg"), (antenne, "gsm-lab-antenne.jpg")):
            Image.open(src).convert("RGB").save(os.path.join(configs, nom), quality=92)
            print(os.path.join(configs, nom))
    sys.exit(0)
if len(sys.argv) != 3:
    sys.exit("usage : osmo-fond-bleu.py [ENTREE.png SORTIE.png]")

SRC, DST = sys.argv[1], sys.argv[2]
im = Image.open(SRC).convert("RGB")
W, H = im.size
k = W / 2560
a = np.asarray(im, float) / 255
r, g, b = a[..., 0], a[..., 1], a[..., 2]
yy, xx = np.mgrid[0:H, 0:W]

# teinte (-180..180 degres) et saturation
mx, mn = a.max(2), a.min(2)
c = mx - mn
h = np.zeros_like(mx)
nz = c > 1e-6
i = nz & (mx == r); h[i] = ((g - b)[i] / c[i]) % 6
i = nz & (mx == g) & (mx != r); h[i] = (b - r)[i] / c[i] + 2
i = nz & (mx == b) & (mx != r) & (mx != g); h[i] = (r - g)[i] / c[i] + 4
h = h * 60
h = np.where(h > 180, h - 360, h)
sat = np.where(mx > 0, c / np.maximum(mx, 1e-6), 0)

# poids « orange » : du bordeaux (-70) au jaune orangé (60), pixels saturés
w = np.clip((60 - h) / 15, 0, 1) * np.clip((h + 70) / 30, 0, 1) * np.clip((sat - .25) / .25, 0, 1)

# zones protegees : seuls les pixels vifs (logo, lettres, traits) y sont gardés,
# les coups de pinceau plus sombres derriere passent en bleu comme le reste
P = np.zeros((H, W))
for x0, y0, x1, y1 in [(192, 128, 380, 310),      # logo ubuntu
                       (200, 470, 1060, 720),     # OPERATOR
                       (2045, 475, 2415, 700),    # NO RF + trait
                       (1320, 1075, 1660, 1305),  # schema : telephones + BTS
                       (1660, 1130, 2440, 1335)]: # schema : BSC / MSC / Core (pas le reflet au-dessus)
    P[int(y0 * k):int(y1 * k), int(x0 * k):int(x1 * k)] = 1
P[(xx - 2205 * k) ** 2 + (yy - 285 * k) ** 2 < (268 * k) ** 2] = 1   # cercle barré
vif = ((mx > .62) & (sat > .5)).astype(np.uint8) * 255
vif = np.asarray(Image.fromarray(vif).filter(ImageFilter.MaxFilter(5)), float) / 255
P = np.asarray(Image.fromarray((P * vif * 255).astype(np.uint8))
               .filter(ImageFilter.GaussianBlur(2 * k)), float) / 255
```

```
w = w * (1 - P)
if os.environ.get("OSMO_FOND_ORANGE_HAUT", "0") == "1":
    # orange pur sur le premier cinquieme, fondu en smoothstep jusqu aux 3/5 : bleu en dessous
    u = np.clip((yy / H - 0.2) / 0.4, 0, 1)
    w = w * (u * u * (3 - 2 * u))

# dégradé marine -> cyan, suivant la luminosité
# [2026-10-24] etait violet sombre (45, 22, 85) -> bleu clair (120, 195, 255) ; demande : du cyan, un marine plus
 ↪  clair
L = 0.3 * r + 0.59 * g + 0.11 * b
t = np.clip(L / 0.6, 0, 1)[..., None]
MARINE, CYAN = np.array([35, 70, 140]) / 255, np.array([80, 225, 255]) / 255
neuf = (MARINE * (1 - t) + CYAN * t) * (0.55 + 0.45 * np.clip(mx / 0.9, 0, 1))[..., None] * 1.15
out = a * (1 - w[..., None]) + np.clip(neuf, 0, 1) * w[..., None]

# [2026-10-24] TITRE EN UBUNTU SANS. « TELECOM OPERATOR » etait peint au pinceau dans la source : on efface
# les lettres (blanches ou orange, dans le cadre du titre, dilatees pour prendre leur liseré sombre) en
# rebouchant par une convolution normalisee de plus en plus large (pas d OpenCV sur le banc), puis on les
# reecrit plus bas dans la police par defaut d Ubuntu. Le sous-titre « Open Source... » reste tel quel.
X0, Y0, X1, Y1 = (round(v * k) for v in (160, 375, 1065, 708))
L = 0.3 * out[..., 0] + 0.59 * out[..., 1] + 0.11 * out[..., 2]
omx, omn = out.max(2), out.min(2)
osat = np.where(omx > 0, (omx - omn) / np.maximum(omx, 1e-6), 0)
lettre = np.zeros((H, W), bool)
lettre[Y0:Y1, X0:X1] = (((L > .38) & (osat < .45)) |
                        ((out[..., 0] > .45) & (out[..., 0] > out[..., 1] * 1.3) & (osat > .4)))[Y0:Y1, X0:X1]
M = np.asarray(Image.fromarray((lettre * 255).astype(np.uint8))
               .filter(ImageFilter.MaxFilter(9)).filter(ImageFilter.MaxFilter(5))) > 127
connu = (~M).astype(float)
for rayon in (2, 4, 8, 16, 32, 64):
    num = np.asarray(Image.fromarray((np.clip(out * connu[..., None], 0, 1) * 255).astype(np.uint8))
                     .filter(ImageFilter.GaussianBlur(rayon * k)), float) / 255
    den = np.asarray(Image.fromarray((connu * 255).astype(np.uint8))
                     .filter(ImageFilter.GaussianBlur(rayon * k)), float) / 255
    sel = M & (den > .02) & (connu < .5)
    out[sel] = (num / np.maximum(den[..., None], 1e-3))[sel]
    connu[sel] = 1

# [2026-10-24] un peu plus clair : gamma sur tout le fond (les sombres remontent plus que les clairs)
out = np.clip(out, 0, 1) ** float(os.environ.get("OSMO_FOND_GAMMA", "0.82"))
img = Image.fromarray((np.clip(out, 0, 1) * 255).astype(np.uint8))
TITRE_FONTE = "/usr/share/fonts/truetype/ubuntu/UbuntuSans[wdth,wght].ttf"
ft = ImageFont.truetype(TITRE_FONTE, round(150 * k))
ft.set_variation_by_axes([100, 800])
# les couleurs mesurees sur les lettres peintes : TELECOM blanc casse, OPERATOR orange en degrade vertical
def titre_degrade(xy, texte, haut, bas):
    masque = Image.new("L", img.size, 0)
    ImageDraw.Draw(masque).text(xy, texte, font=ft, fill=255, anchor="lt")
    y0_, y1_ = masque.getbbox()[1], masque.getbbox()[3]
    tt = np.clip((np.arange(img.size[1]) - y0_) / max(1, y1_ - y0_), 0, 1)[:, None, None]
    couleur = np.array(haut) * (1 - tt) + np.array(bas) * tt
    couche = Image.fromarray(np.broadcast_to(couleur, (img.size[1], img.size[0], 3)).astype(np.uint8))
    img.paste(couche, (0, 0), masque)
titre_degrade((round(200 * k), round(400 * k)), "TELECOM", (249, 247, 245), (248, 246, 244))
titre_degrade((round(200 * k), round(545 * k)), "OPERATOR", (248, 112, 2), (247, 99, 2))

# cadre autour du schema : tout tourne en local, sans emission radio. Le titre
# est pose dans une coupure du trait du haut (comme une legende de fieldset).
ORANGE = (255, 140, 40)
SANS = "/usr/share/fonts/truetype/ubuntu/Ubuntu[wdth,wght].ttf"
fonte = ImageFont.truetype(SANS, round(30 * k))
fonte.set_variation_by_axes([100, 500])
titre = "localhost  ·  no radio emission"
x0, y0, x1, y1 = (round(v * k) for v in (1300, 1035, 2465, 1345))
calque = Image.new("RGBA", img.size, (0, 0, 0, 0))
d = ImageDraw.Draw(calque)
```

```
d.rounded_rectangle((x0, y0, x1, y1), radius=round(18 * k),
                    outline=ORANGE + (235,), width=max(2, round(3 * k)))
tl = d.textlength(titre, font=fonte)
tx = x0 + round(40 * k)
d.rectangle((tx - round(14 * k), y0 - round(6 * k), tx + tl + round(14 * k), y0 + round(6 * k)), fill=(0, 0, 0, 0))
img = img.convert("RGBA")
# la coupure doit laisser voir le fond : on efface le trait sous le titre
trou = Image.new("L", img.size, 255)
ImageDraw.Draw(trou).rectangle((tx - round(14 * k), y0 - round(6 * k), tx + tl + round(14 * k), y0 + round(6 *
 ↪  k)), fill=0)
# le coin du portable est DEVANT le cadre : pas de trait sur sa coque
ImageDraw.Draw(trou).polygon([(round(x * k), round(y * k)) for x, y in
                              ((1150, 1312), (1482, 1356), (1505, 1385), (1150, 1395))], fill=0)
calque.putalpha(Image.fromarray(np.minimum(np.asarray(calque.getchannel("A")), np.asarray(trou))))
img.alpha_composite(calque)
ImageDraw.Draw(img).text((tx, y0), titre, font=fonte, fill=ORANGE + (255,), anchor="lm")
img.convert("RGB").save(DST, optimize=True)
print(DST)
```

3.411 /opt/GSM/osmo-operator/tools/osmo-fond-repos.py

5757 octets, 121 lignes → 121 lignes

```
#!/usr/bin/env python3
# Fond « Telecom Operator » : agrandi en 2560x1600, terminal et ligne GSM réécrits nets.
#
# [2026-10-23] Image de depart : configs/gsm-lab-source.png (1586x992, generee par
# IA : texte du terminal invente, ligne « GSM | ... | MSC »). Regenerer le fond au
# repos, puis sa variante « banc en marche » :
#   tools/osmo-fond-repos.py configs/gsm-lab-source.png /tmp/repos.png
#   tools/osmo-fond-bleu.py /tmp/repos.png /tmp/bleu.png     (fond orange -> bleu)
#   tools/osmo-fond-antenne.py /tmp/bleu.png /tmp/antenne.png
# et poser les deux en JPEG (qualite 92) sous configs/gsm-lab-ville.jpg et
# configs/gsm-lab-antenne.jpg (les noms que lit tools/osmo-fond.py).
# Les coordonnees (ecran du portable, ligne GSM) sont celles de CETTE image source.
import sys
import numpy as np
from PIL import Image, ImageDraw, ImageFont, ImageFilter

SRC, DST = sys.argv[1], sys.argv[2]
W, H = 2560, 1600
MONO = "/usr/share/fonts/truetype/dejavu/DejaVuSansMono.ttf"
MONO_B = "/usr/share/fonts/truetype/dejavu/DejaVuSansMono-Bold.ttf"
SANS = "/usr/share/fonts/truetype/ubuntu/Ubuntu[wdth,wght].ttf"

src = Image.open(SRC).convert("RGB")
sx, sy = W / src.width, H / src.height
up = src.resize((W, H), Image.LANCZOS).filter(ImageFilter.UnsharpMask(2, 60, 2))


def coeffs(dst_pts, src_pts):
    """Coefficients PERSPECTIVE : point de sortie (dst) -> point d entree (src)."""
    A, b = [], []
    for (x, y), (u, v) in zip(dst_pts, src_pts):
        A.append([x, y, 1, 0, 0, 0, -u * x, -u * y]); b.append(u)
        A.append([0, 0, 0, x, y, 1, -v * x, -v * y]); b.append(v)
    return np.linalg.solve(np.array(A, float), np.array(b, float)).tolist()


def efface(img, masque, sigma):
    """Remplit les pixels du masque par un flou normalisé des pixels voisins."""
    a = np.asarray(img, float)
    m = (np.asarray(masque) > 0)
    garde = Image.fromarray(((~m) * 255).astype(np.uint8))
```

```
    num = np.stack([np.asarray(Image.fromarray((a[..., c] * ~m).astype(np.uint8))
                               .filter(ImageFilter.GaussianBlur(sigma)), float) for c in range(3)], -1)
    den = np.asarray(garde.filter(ImageFilter.GaussianBlur(sigma)), float)[..., None] / 255
    rempli = num / np.maximum(den, 1e-3)
    a[m] = rempli[m]
    return Image.fromarray(np.clip(a, 0, 255).astype(np.uint8))


# ── terminal du portable ──────────────────────────────────────────────
Q = [(184 * sx, 507 * sy), (642 * sx, 521 * sy), (682 * sx, 793 * sy), (218 * sx, 806 * sy)]
FW, FH = 1480, 960
F = [(0, 0), (FW, 0), (FW, FH), (0, FH)]

plat = up.transform((FW, FH), Image.PERSPECTIVE, coeffs(F, Q), Image.BICUBIC)
lum = np.asarray(plat.convert("L"))
masque = (lum > 38).astype(np.uint8) * 255
masque[:80] = 0                                      # garde la barre de titre
masque = Image.fromarray(masque).filter(ImageFilter.MaxFilter(13))
plat = efface(plat, masque, 18)

VERT, BLANC, GRIS = (138, 226, 52), (225, 225, 225), (150, 150, 160)
lignes = [
    [("ubuntu@osmocom:~$ ", VERT, 1), ("systemctl status osmo-bsc", BLANC, 0)],
    [("● ", VERT, 0), ("osmo-bsc.service - Osmocom BSC", BLANC, 0)],
    [("     Loaded: loaded (/lib/systemd/system/osmo-bsc.service; enabled)", BLANC, 0)],
    [("     Active: ", BLANC, 0), ("active (running)", VERT, 1)],
    [("   Main PID: 2483 (osmo-bsc)", BLANC, 0)],
    [("      Tasks: 12 (limit: 2300)", BLANC, 0)],
    [("     Memory: 42.6M", BLANC, 0)],
    [("     CGroup: /system.slice/osmo-bsc.service", BLANC, 0)],
    [("             └─2483 /usr/bin/osmo-bsc", BLANC, 0)],
    [],
    [("ubuntu@osmocom:~$ ", VERT, 1), ("systemctl is-active osmo-bts-virtual", BLANC, 0),
     ("  # NO RF", GRIS, 0)],
    [("active", VERT, 1)],
    [],
    [("ubuntu@osmocom:~$ ", VERT, 1)],
]
fonte = {0: ImageFont.truetype(MONO, 31), 1: ImageFont.truetype(MONO_B, 31)}
calque = Image.new("RGBA", (FW, FH), (0, 0, 0, 0))
d = ImageDraw.Draw(calque)
y = 108
for morceaux in lignes:
    x = 58
    for texte, coul, gras in morceaux:
        d.text((x, y), texte, font=fonte[gras], fill=coul + (255,))
        x += d.textlength(texte, font=fonte[gras])
    if morceaux and morceaux[0][0].startswith("ubuntu@") and len(morceaux) == 1:
        d.rectangle([x + 2, y + 2, x + 18, y + 36], fill=(150, 230, 190, 255))   # curseur
    y += 50
# léger halo, comme un écran éclairé
halo = calque.filter(ImageFilter.GaussianBlur(4))
plat = plat.convert("RGBA")
plat.alpha_composite(Image.eval(halo, lambda v: v // 3))
plat.alpha_composite(calque)

retour = coeffs(Q, F)
colle = plat.transform((W, H), Image.PERSPECTIVE, retour, Image.BICUBIC)
forme = Image.new("L", (FW, FH), 255).transform((W, H), Image.PERSPECTIVE, retour, Image.BICUBIC)
up.paste(colle.convert("RGB"), (0, 0), forme)

# ── ligne « GSM | GPRS | EDGE | LTE » sous le logo osmocom ────────────
bx0, by0, bx1, by1 = int(615 * sx), int(176 * sy), int(995 * sx), int(208 * sy)
zone = up.crop((bx0, by0, bx1, by1))
mz = Image.fromarray(((np.asarray(zone.convert("L")) > 42) *
 ↪  255).astype(np.uint8)).filter(ImageFilter.MaxFilter(11))
```

```
up.paste(efface(zone, mz, 10), (bx0, by0))

ff = ImageFont.truetype(SANS, 33)
ff.set_variation_by_axes([100, 400])
texte = "GSM   |   GPRS   |   EDGE   |   LTE"
d = ImageDraw.Draw(up)
espace = 4                                          # interlettrage comme l original
largeur = sum(d.textlength(c, font=ff) + espace for c in texte) - espace
x, cy = (bx0 + bx1) / 2 - largeur / 2, (by0 + by1) / 2
for c in texte:
    d.text((x, cy), c, font=ff, fill=(215, 215, 220), anchor="lm")
    x += d.textlength(c, font=ff) + espace

up.save(DST, optimize=True)
print(DST, up.size)
```

3.412 /opt/GSM/osmo-operator/tools/osmo-fond.py

9970 octets, 248 lignes → 248 lignes

```
#!/usr/bin/env python3
# osmo-fond.py - LE FOND D ECRAN DU BANC, EN DEUX ETATS, AVEC UN FONDU ENTRE LES DEUX.
#
#   osmo-fond ville        le bureau au repos : la ville au bord de l eau, sans
#                          antenne (configs/gsm-lab-ville.jpg)
#   osmo-fond antenne      un banc tourne : la MEME ville, le pylone 2G/4G
#                          dresse devant, les icones qui montent
#                          (configs/gsm-lab-antenne.jpg)
#   osmo-fond etat         affiche l etat courant (ville / antenne / autre)
#   osmo-fond <fichier>    n importe quelle image, meme fondu
#
#   --duree S     longueur du fondu (defaut 2.4 s) ; --sec : pas de fondu
#   --pas N       nombre d images intermediaires (defaut 12)
#
# [2026-10-12] POURQUOI. Le bureau montrait l antenne en permanence. Choix du
# banc : au repos, la ville seule ; quand on LANCE la 2G (run standalone) ou
# la 4G (run multi), l antenne se dresse dans le paysage - en fondu, pas d un
# coup - et elle repart quand tout s eteint. Deux appelants :
#   - tools/osmo-launcher.py, au CLIC sur un run (immediat, avant meme que le
#     service soit debout) ;
#   - tools/osmo-panel.py, qui sait deja quand un banc monte ou tombe
#     (montrer(actif)) : couvre un lancement depuis un terminal, et le retour
#     a la ville a l arret.
# Les deux peuvent se suivre de pres : l outil est IDEMPOTENT (deja sur la
# cible : rien), et un seul fondu a la fois par session (verrou).
#
# COMMENT. GNOME Shell fait lui-meme un petit fondu croise (~1 s) a chaque
# changement de picture-uri. On s appuie dessus et on l allonge : Pillow rend N
# images intermediaires (melange lineaire des deux fonds) dans
# $XDG_RUNTIME_DIR/osmo-fond/, et on les pousse une a une par gsettings, a
# intervalle regulier ; la derniere valeur posee est le VRAI fichier
# (/usr/share/backgrounds/gsm-lab-*.jpg), pas une image de cache. Les images
# sont gardees (tmpfs, quelques Mo) : le fondu inverse les relit a l envers.
#
# UTILISATEUR. gsettings parle au bus de SESSION : l outil doit tourner comme
# l utilisateur du bureau. Lance en root (start-docker.sh, un service), il se relance
# lui-meme sous le compte qui a un gnome-shell et un /run/user/<uid>/bus.
import argparse
import fcntl
import hashlib
import os
import pwd
import signal
import subprocess
```

```
import sys
import time
from urllib.parse import quote, unquote, urlparse

REPO = os.environ.get("OSMO_REPO", "/opt/GSM/osmo-operator")
BG_DIR = "/usr/share/backgrounds"
# Nom -> fichier pose par l install (80-chroot.sh / 80-bureau.sh), sinon le depot.
ETATS = {
    "ville":   ("gsm-lab-ville.jpg",   "le bureau au repos"),
    "antenne": ("gsm-lab-antenne.jpg", "un banc tourne"),
}
SCHEMA = "org.gnome.desktop.background"
CLES = ("picture-uri", "picture-uri-dark")


def dire(msg):
    print("[fond] " + msg, flush=True)


def fichier_etat(nom):
    f = ETATS[nom][0]
    for d in (BG_DIR, os.path.join(REPO, "configs")):
        p = os.path.join(d, f)
        if os.path.isfile(p):
            return p
    return None


# ── tourner comme l utilisateur du bureau ──────────────────────────
def _session_bus(uid):
    p = "/run/user/%d/bus" % uid
    return "unix:path=" + p if os.path.exists(p) else None


def se_relancer_en_utilisateur():
    """Root sans bus de session : trouver le bureau et se relancer dessous."""
    if os.geteuid() != 0:
        return
    if os.environ.get("DBUS_SESSION_BUS_ADDRESS") and _session_bus(0):
        return                       # le bureau est a root (la cle live)
    candidats = []
    su = os.environ.get("SUDO_USER")
    if su:
        candidats.append(su)
    try:
        out = subprocess.run(["ps", "-eo", "user:32=,comm="], capture_output=True, text=True).stdout
        candidats += [l.split()[0] for l in out.splitlines() if l.strip().endswith("gnome-shell")]
    except OSError:
        pass
    for nom in candidats:
        try:
            pw = pwd.getpwnam(nom)
        except KeyError:
            continue
        bus = _session_bus(pw.pw_uid)
        if not bus:
            continue
        if pw.pw_uid == 0:
            os.environ["DBUS_SESSION_BUS_ADDRESS"] = bus
            os.environ.setdefault("XDG_RUNTIME_DIR", "/run/user/0")
            return
        env = {"DBUS_SESSION_BUS_ADDRESS": bus, "XDG_RUNTIME_DIR": "/run/user/%d" % pw.pw_uid,
               "HOME": pw.pw_dir, "USER": pw.pw_name, "LOGNAME": pw.pw_name,
               "PATH": os.environ.get("PATH", "/usr/local/bin:/usr/bin:/bin"),
               "OSMO_REPO": REPO, "OSMO_FOND_RELANCE": "1"}
        os.execvp("runuser", ["runuser", "-u", pw.pw_name, "--", "env"] +
```

```
                ["%s=%s" % kv for kv in env.items()] +
                [sys.executable, os.path.abspath(__file__)] + sys.argv[1:])
    dire("aucun bureau GNOME trouve : rien a faire")
    sys.exit(0)


# ── gsettings ──────────────────────────────────────────────────────────
def gs_get(cle):
    try:
        v = subprocess.run(["gsettings", "get", SCHEMA, cle], capture_output=True, text=True,
 ↪  timeout=5).stdout.strip()
    except (OSError, subprocess.TimeoutExpired):
        return ""
    return v.strip("'")


def gs_set(uri):
    for cle in CLES:
        subprocess.run(["gsettings", "set", SCHEMA, cle, uri], timeout=5)


def uri_de(path):
    return "file://" + quote(os.path.abspath(path))


def chemin_de(uri):
    if not uri.startswith("file://"):
        return None
    return unquote(urlparse(uri).path)


def etat_courant():
    """(etat, chemin) : ville / antenne / « <cible> en cours » (image d un fondu
    interrompu - un osmo-fond tue en plein vol, cf. start-native.sh
    killall_python) / autre."""
    p = chemin_de(gs_get("picture-uri")) or ""
    for nom, (f, _) in ETATS.items():
        if os.path.basename(p) == f:
            return nom, p
    for nom in ETATS:
        if "/osmo-fond/" in p and os.path.basename(p).startswith(nom + "-"):
            return nom + " en cours", p
    return "autre", p


# ── le fondu ───────────────────────────────────────────────────────────
def _cache_dir():
    base = os.environ.get("XDG_RUNTIME_DIR") or "/run/user/%d" % os.getuid()
    if not os.path.isdir(base):
        base = "/tmp"
    d = os.path.join(base, "osmo-fond")
    os.makedirs(d, mode=0o700, exist_ok=True)
    return d


def rendre_etapes(src, dst, pas, cible):
    """Les images intermediaires src -> dst (sans les deux bouts) ; None si impossible."""
    try:
        from PIL import Image
    except ImportError:
        return None
    try:
        a = Image.open(src).convert("RGB")
        b = Image.open(dst).convert("RGB")
    except OSError:
        return None
```

```
    if a.size != b.size:
        # ramener la source au format de la cible, en « zoom » comme GNOME
        W, H = b.size
        r = max(W / a.width, H / a.height)
        a = a.resize((max(1, round(a.width * r)), max(1, round(a.height * r))), Image.LANCZOS)
        l, t = (a.width - W) // 2, (a.height - H) // 2
        a = a.crop((l, t, l + W, t + H))
    key = hashlib.sha1(("%s:%s:%s:%s:%d" % (src, os.path.getmtime(src), dst, os.path.getmtime(dst),
↪   pas)).encode()).hexdigest()[:10]
    d = _cache_dir()
    etapes = []
    for i in range(1, pas):
        p = os.path.join(d, "%s-%s-%02d.jpg" % (cible, key, i))
        if not os.path.isfile(p):
            Image.blend(a, b, i / pas).save(p, quality=88)
        etapes.append(p)
    return etapes


def poser(cible, duree, pas, sec):
    nom = cible if cible in ETATS else "autre"
    dst = fichier_etat(cible) if cible in ETATS else os.path.abspath(cible)
    if not dst or not os.path.isfile(dst):
        dire("image introuvable : %s" % (dst or cible))
        return 1
    etat, src = etat_courant()
    if (etat == nom and nom != "autre") or (src and os.path.abspath(src) == dst):
        dire("deja « %s » : rien a faire" % nom)
        return 0
    # un fondu interrompu : on repart de l image ou il s est arrete (src)
    # un seul fondu a la fois par session
    verrou = open(os.path.join(_cache_dir(), "verrou"), "w")
    try:
        fcntl.flock(verrou, fcntl.LOCK_EX | fcntl.LOCK_NB)
    except OSError:
        dire("un fondu est deja en cours")
        return 0
    etapes = None
    if not sec and duree > 0 and pas > 1 and src and os.path.isfile(src):
        etapes = rendre_etapes(src, dst, pas, nom)
    try:
        if etapes:
            dire("fondu %s -> %s (%d images, %.1f s)" % (etat, nom, len(etapes), duree))
            dt = duree / (len(etapes) + 1)
            for p in etapes:
                gs_set(uri_de(p))
                time.sleep(dt)
        else:
            dire("%s -> %s (fondu GNOME seul)" % (etat, nom))
    finally:
        # quoi qu il arrive (erreur, SIGTERM), le VRAI fichier est pose
        gs_set(uri_de(dst))
    return 0


def main():
    ap = argparse.ArgumentParser(description="fond d ecran du banc : ville / antenne, en fondu")
    ap.add_argument("cible", help="ville | antenne | etat | <fichier image>")
    ap.add_argument("--duree", type=float, default=2.4, help="duree du fondu en secondes (defaut 2.4)")
    ap.add_argument("--pas", type=int, default=12, help="images intermediaires (defaut 12)")
    ap.add_argument("--sec", action="store_true", help="sans fondu")
    a = ap.parse_args()
    se_relancer_en_utilisateur()
    # SIGTERM pendant le fondu : le finally de poser() pose la cible
    signal.signal(signal.SIGTERM, lambda *_: sys.exit(143))
    if a.cible == "etat":
```

```
        etat, p = etat_courant()
        print("%s\t%s" % (etat, p))
        return 0
    return poser(a.cible, a.duree, a.pas, a.sec)


if __name__ == "__main__":
    sys.exit(main())
```

3.413 /opt/GSM/osmo-operator/tools/osmo-gcc-alternatives.sh

1524 octets, 25 lignes → 25 lignes

```
#!/bin/bash
# osmo-gcc-alternatives.sh - gcc-9 / gcc-11 / gcc-13 dans update-alternatives (g++ en esclave).
#
# [2026-10-24] « update-alternatives --config gcc » repondait « pas d alternatives pour gcc »
# sur l ISO : seul update.sh les posait, en rattrapage, et l ISO n avait que gcc-13. Les
# branches experimentales d osmocom-bb (jolly/testing, burst_ind) ne compilent qu avec gcc-9
# (cf. Dockerfile, stage bb). Ce script est LA definition, appelee par iso_modules/80-chroot.sh
# (dans le chroot, apres apt), install_modules/80-bureau.sh et update.sh.
#
# Priorites : gcc-9 (90), gcc-11 (110), gcc-13 (130). Un choix deja fait
# (update-alternatives --config gcc) est garde ; a la creation, gcc-13 - le compilateur du
# noyau, pour DKMS. Seuls les compilateurs presents entrent : ni apt, ni reseau ici.
[ "$(id -u)" -eq 0 ] || { echo "osmo-gcc-alternatives : il faut root" >&2; exit 1; }
command -v update-alternatives >/dev/null 2>&1 || exit 0
neuf=0
update-alternatives --list gcc >/dev/null 2>&1 || neuf=1
for v in 9 11 13; do
    [ -x "/usr/bin/gcc-$v" ] && [ -x "/usr/bin/g++-$v" ] || continue
    update-alternatives --list gcc 2>/dev/null | grep -qx "/usr/bin/gcc-$v" && continue
    update-alternatives --install /usr/bin/gcc gcc "/usr/bin/gcc-$v" "$(( v * 10 ))" \
        --slave /usr/bin/g++ g++ "/usr/bin/g++-$v" >/dev/null 2>&1 \
        && echo "  [outils] alternative gcc-$v posee"
done
[ "$neuf" = 1 ] && [ -x /usr/bin/gcc-13 ] && update-alternatives --set gcc /usr/bin/gcc-13 >/dev/null 2>&1
exit 0
```

3.414 /opt/GSM/osmo-operator/tools/osmo-glava.sh

4230 octets, 78 lignes → 78 lignes

```
#!/bin/bash
# osmo-glava - lance le visualiseur NCS Spectrum (GLava) sur le bureau.
#
# Pose par tools/glava-build.sh dans /usr/local/bin, lance par le service
# utilisateur glava.service (systemctl --user stop|start glava). Un seul
# GLava NCS par utilisateur.
#
# [2026-10-23] La fenetre est un carre de 26 % de la hauteur de l ecran, place
# au-dessus de la ville du fond (gsm-lab-ville.jpg) : x = 49 %, y = 21 %. Sur
# 2560x1600, c est 420 px en (1250, 330). Calcule ici d apres l ecran
# principal, plutot que fige dans rc.glsl : la meme ISO tourne sur des ecrans
# differents.
#
# [2026-10-24] LE MICRO ET LES HAUT-PARLEURS. GLava ne lit qu UNE source
# PulseAudio. On lui en fabrique une : un sink nul « osmo_glava », dans lequel
# deux module-loopback recopient le moniteur de la sortie par defaut (ce que
# jouent les haut-parleurs) et la source par defaut (le micro - sur le banc,
# osmo_mic_ec, le micro a annulation d echo). GLava ecoute osmo_glava.monitor.
# Le micro ne va QUE dans ce sink nul, jamais vers les haut-parleurs : pas de
```

```
# larsen. Les sources par defaut sont lues au lancement (connexion).
# Le garde ne vise que l instance NCS (-r setgeometry...) : une autre instance
# (glava -m bars, par exemple) ne l empeche pas de demarrer.
MIX=osmo_glava
# --retirer-mix : appele a l arret du service (ExecStopPost de glava.service),
# retire les deux boucles puis le sink nul (dans cet ordre : le sink emporte
# ses boucles, un second retrait echouerait).
if [ "${1:-}" = "--retirer-mix" ]; then
    pactl list short modules 2>/dev/null \
        | awk -v m="$MIX" '$0 ~ ("sink(_name)?=" m "([[:space:]]|$)") {print $1}' \
        | sort -rn | while read -r id; do pactl unload-module "$id"; done
    exit 0
fi
pgrep -u "$(id -u)" -f '^glava -r setgeometry' >/dev/null && exit 0
sleep "${OSMO_GLAVA_DELAI:-6}"      # comme conky : le bureau GNOME doit etre peint

if command -v pactl >/dev/null 2>&1; then
    if ! pactl list short sinks 2>/dev/null | grep -q "^[0-9]*[[:space:]]$MIX[[:space:]]"; then
        pactl load-module module-null-sink sink_name=$MIX \
            sink_properties=device.description=GLava-micro+HP >/dev/null
        # [2026-10-24] Les haut-parleurs = le moniteur de la sortie MATERIELLE.
        # @DEFAULT_MONITOR@ ne suffit pas : a la connexion, le sink par defaut
        # peut etre osmo_hp_ec (l annulation d echo du banc, 8 kHz mono, la seule
        # voix GSM) - GLava ne voyait alors ni le navigateur ni le reste. Le sink
        # ALSA recoit tout, osmo_hp_ec compris (son maitre).
        hp=$(pactl get-default-sink 2>/dev/null)
        case "$hp" in alsa_output.*|bluez_sink.*) ;;
            *) hp=$(pactl list short sinks 2>/dev/null | awk '$2 ~ /^(alsa_output|bluez_sink)\./ {print $2;
             ↪  exit}') ;;
        esac
        hp=${hp:+$hp.monitor}
        pactl load-module module-loopback source="${hp:-@DEFAULT_MONITOR@}" sink=$MIX \
            source_dont_move=true sink_dont_move=true latency_msec=30 >/dev/null
        pactl load-module module-loopback source=@DEFAULT_SOURCE@ sink=$MIX \
            source_dont_move=true sink_dont_move=true latency_msec=30 >/dev/null
    fi
    SOURCE=(-r "setsource \"$MIX.monitor\"")
else
    SOURCE=()
fi

# [2026-10-24] LA BANDE DU BAS : une seconde instance, module bars, sur toute
# la largeur libre (a droite du dock : _NET_WORKAREA) et 8,75 % de la hauteur
# (140 px sur 1600), collee au bord bas. Meme source que le cercle.
lancer_barres() {   # W H
    pgrep -u "$(id -u)" -f '^glava -m bars' >/dev/null && return 0
    local zone x0 h
    zone=$(xprop -root _NET_WORKAREA 2>/dev/null | sed -n 's/.*= \([0-9]*\), .*/\1/p')
    x0=${zone:-0}
    h=$(( $2 * 875 / 10000 ))
    setsid -f glava -m bars -r "setgeometry $x0 $(( $2 - h )) $(( $1 - x0 )) $h" "${SOURCE[@]}" \
        >/dev/null 2>&1
}

ecran=$(xrandr --current 2>/dev/null | sed -n 's/.* connected primary \([0-9]*\)x\([0-9]*\)+.*/\1 \2/p' | head -1)
[ -n "$ecran" ] || ecran=$(xrandr --current 2>/dev/null | sed -n 's/.*current \([0-9]*\) x \([0-9]*\),.*/\1 \2/p'
 ↪  | head -1)
read -r W H <<< "${ecran:-1920 1080}"
cote=$(( H * 26 / 100 ))
lancer_barres "$W" "$H"
exec glava -r "setgeometry $(( W * 49 / 100 )) $(( H * 21 / 100 )) $cote $cote" "${SOURCE[@]}" "$@"
```

3.415 /opt/GSM/osmo-operator/tools/osmo-gtk-run.py

25081 octets, 578 lignes → 578 lignes

```
#!/usr/bin/env python3
# osmo-gtk-run.py - UNE COMMANDE, UNE FENETRE GTK : le terminal devient un journal.
#
#     osmo-gtk-run.py [--title T] [--icon FICHIER] [--root] [--tty] [--progress]
#                     [--keep-open] [--cwd DIR] -- COMMANDE [ARGS...]
#
# [2026-10-01] AUCUN FAVORI N OUVRE PLUS DE TERMINAL. Les icones du dock et du
# bureau (supplements, update, Claude, deka, telephone pmOS, 4G, pilotes, le
# journal du multi, la console tmux, la VTY du mobile...) ouvraient chacune un
# gnome-terminal, avec un « read -n1 » a la fin pour que la fenetre ne se
# ferme pas avant qu on ait lu. Elles passent toutes par ICI : une fenetre GTK
# du banc (meme habillage que la barre de lancement, tools/osmo-launcher.py),
# qui montre la sortie de la commande, dit quand elle est finie et avec quel
# statut, et se ferme.
#
# [2026-10-24] LA FIN DU SCRIPT FERME LA FENETRE. Un favori qui a fini n a
# plus rien a montrer : sortie en 0, la fenetre part apres 1,5 s ; en echec,
# la console est depliee et un compte a rebours de 15 s laisse lire l erreur
# (« garder ouverte » l arrete). --keep-open : l ancien comportement, pour ce
# qu on ouvre POUR LIRE (« Etat 4G », « Etat du telephone »). --close-on-exit
# est accepte et ne change plus rien (c est le defaut).
#
# DEUX FACONS DE MONTRER LA COMMANDE :
#   - journal (defaut) : stdout + stderr de la commande dans une vue texte
#     (couleurs ANSI retirees, defilement automatique, « copier »). Pour tout
#     ce qui ne fait qu ecrire : journalctl -f, un status, un service qu on
#     demarre.
#   - --tty : un VRAI terminal (widget VTE) dans la fenetre. Pour ce qui
#     pose des questions (whiptail, sudo, « Entree pour continuer »), anime
#     la ligne (update.sh teste `[ -t 1 ]` et sort sans tty), ou est
#     interactif de bout en bout (claude, tmux attach, telnet). C est le
#     terminal, mais DANS le GTK, avec le meme bandeau d etat ; sans
#     gir1.2-vte-2.91, on retombe sur le journal et on le dit.
#
# --progress : LA BARRE DE PROGRESSION, PAS LE SHELL. [2026-10-12] Pour les
#   lancements (2G, multi, 4G, telephone pmOS) on ne veut pas lire un defile
#   de shell : une petite fenetre avec une barre, le nom de l etape en cours,
#   la derniere ligne - et un bouton « console » qui DEPLIE la sortie complete
#   (un vrai terminal VTE, ou le journal sans VTE) pour qui veut voir. La
#   commande tourne dans un pseudo-terminal tenu ici : elle se croit dans un
#   terminal (couleurs, questions, pkexec), ce qu elle ecrit est lu au passage
#   pour la barre et transmis a la console. L avancement vient des reperes
#   « [n/m] » que les scripts du banc impriment deja (run.sh : [3/10] Core
#   Osmocom ; osmo-lte, osmo-pmos : poses le 2026-10-12) ; sans repere, la
#   barre pulse et montre la derniere ligne. Fin : barre pleine (vert) ou
#   rouge avec le statut.
#
# --root : la commande tourne en root. Deja root : rien a faire. Sinon pkexec
# (la fenetre de mot de passe du bureau) avec DISPLAY, XAUTHORITY et les
# variables de proxy de la session - pkexec nettoie l environnement, et sans
# ce report les git clone des supplements echouent derriere un proxy. Sans
# pkexec : sudo (qui demande dans le terminal en --tty, et ne peut pas
# demander en journal : sudo -n, l echec se lit dans la fenetre).
#
# Lance par les .desktop du depot (data/desktop/), les lanceurs poses par
# iso_modules/85-installeur-bureau.sh, addition.sh, update.sh, et par les
# barres GTK du bureau (osmo-launcher.py, osmo-panel.py) pour leurs entrees
# interactives.
import argparse
import fcntl
import os
import re
```

```
import shlex
import shutil
import signal
import struct
import subprocess
import sys
import termios
import threading


def _root_prefix(tty):
    """Prefixe argv pour executer en root : rien si on l est deja, pkexec avec
    l environnement utile sinon, sudo en dernier recours."""
    if os.geteuid() == 0:
        return []
    if shutil.which("pkexec") and os.environ.get("DISPLAY"):
        env = ["DISPLAY=" + os.environ.get("DISPLAY", ""),
               "XAUTHORITY=" + os.environ.get("XAUTHORITY", "")]
        for v in ("http_proxy", "https_proxy", "ftp_proxy", "no_proxy",
                  "HTTP_PROXY", "HTTPS_PROXY", "FTP_PROXY", "NO_PROXY"):
            if os.environ.get(v):
                env.append(v + "=" + os.environ[v])
        if not tty:
            env += ["NO_COLOR=1", "TERM=dumb"]
        return ["pkexec", "env"] + env
    if shutil.which("sudo"):
        return ["sudo", "-E"] if tty else ["sudo", "-n", "-E"]
    return []


def _parse():
    ap = argparse.ArgumentParser(description="montre une commande dans une fenetre GTK")
    ap.add_argument("--title", default="osmo-operator")
    ap.add_argument("--icon", default=None, help="fichier d icone (PNG/SVG)")
    ap.add_argument("--root", action="store_true", help="en root (pkexec, ou sudo)")
    ap.add_argument("--tty", action="store_true", help="un terminal (VTE) plutot qu un journal")
    ap.add_argument("--progress", action="store_true",
                    help="une barre de progression, la console repliee derriere un bouton")
    ap.add_argument("--close-on-exit", action="store_true", help="(defaut, garde pour les anciens .desktop)")
    ap.add_argument("--keep-open", action="store_true", help="la fenetre reste ouverte a la fin de la commande")
    ap.add_argument("--cwd", default=None)
    ap.add_argument("--width", type=int, default=980)
    ap.add_argument("--height", type=int, default=600)
    ap.add_argument("cmd", nargs=argparse.REMAINDER)
    a = ap.parse_args()
    if a.cmd and a.cmd[0] == "--":
        a.cmd = a.cmd[1:]
    if not a.cmd:
        ap.error("aucune commande (osmo-gtk-run.py --title T -- commande args)")
    return a


ARGS = _parse()


def _exec_direct(pourquoi):
    # Pas de bureau, ou pas de GTK : on execute tel quel, dans le terminal
    # courant - mieux vaut la commande sans fenetre que pas de commande.
    print("[osmo-gtk-run] %s : execution directe" % pourquoi, file=sys.stderr)
    argv = (_root_prefix(True) if ARGS.root else []) + ARGS.cmd
    if ARGS.cwd:
        os.chdir(ARGS.cwd)
    os.execvp(argv[0], argv)


if not os.environ.get("DISPLAY") and not os.environ.get("WAYLAND_DISPLAY"):
```

```
    _exec_direct("pas de bureau (DISPLAY)")

try:
    import gi
    gi.require_version("Gtk", "3.0")
    gi.require_version("Gdk", "3.0")
    from gi.repository import Gdk, GLib, Gtk  # noqa: E402
except (ImportError, ValueError) as _e:
    _exec_direct("GTK indisponible (%s)" % _e)

try:
    gi.require_version("Vte", "2.91")
    from gi.repository import Vte  # noqa: E402
except (ValueError, ImportError):
    Vte = None

ANSI = re.compile(r"\x1b\[[0-9;?]*[A-Za-z]|\x1b\][^\x07]*\x07|\r")
MAX_CHARS = 400_000          # au-dela, on coupe le debut : le journal n est pas une archive

CSS = b"""
.osmo-run { background-color: #0b0e14; }
.osmo-run label.title { color: #58a6ff; font: bold 10pt "Ubuntu"; }
.osmo-run label.state { color: #8b949e; font: 9pt "Ubuntu"; }
.osmo-run label.state.ok { color: #3fb950; }
.osmo-run label.state.ko { color: #f85149; }
.osmo-run button { background: rgba(22,27,34,0.9); color: #e6edf3;
                   border: 1px solid rgba(88,166,255,0.3); border-radius: 6px;
                   padding: 2px 10px; font: 9pt "Ubuntu"; min-height: 0; }
.osmo-run button:hover { border-color: #58a6ff; }
.osmo-run textview, .osmo-run textview text { background-color: #0b0e14; color: #d8e0ea;
                   font-family: "Ubuntu Mono"; font-size: 10pt; }
.osmo-run .cmd { color: #6e7681; font: 8pt "Ubuntu Mono"; }
.osmo-run progressbar trough { background: rgba(22,27,34,0.95); border: 1px solid rgba(88,166,255,0.25);
                   border-radius: 6px; min-height: 14px; }
.osmo-run progressbar progress { background: linear-gradient(90deg, #58a6ff, #3fe08a); border-radius: 6px;
                   min-height: 14px; }
.osmo-run progressbar.ko progress { background: #f85149; }
.osmo-run progressbar text { color: #e6edf3; font: bold 9pt "Ubuntu"; }
.osmo-run label.etape { color: #e6edf3; font: 10pt "Ubuntu"; }
.osmo-run label.derniere { color: #8b949e; font: 9pt "Ubuntu Mono"; }
"""
ETAPE = re.compile(r"\[\s*(\d+)\s*/\s*(\d+)\s*\]")
COMPACT = (640, 190)


class Run(Gtk.Window):
    def __init__(self, a):
        super().__init__(title=a.title)
        self.a = a
        self.proc = None            # journal : Popen
        self.child_pid = None       # tty : pid du fils VTE
        self.fini = False
        self.progress = bool(a.progress)
        self.master = None          # progress : le pty de la commande
        self.fraction = None        # progress : None tant qu aucun repere [n/m]
        self._lbuf = ""
        self.set_default_size(*(COMPACT if self.progress else (a.width, a.height)))
        if a.icon and os.path.exists(a.icon):
            try:
                self.set_icon_from_file(a.icon)
            except Exception:
                pass
        self.connect("destroy", self._quit)
        prov = Gtk.CssProvider()
        prov.load_from_data(CSS)
        Gtk.StyleContext.add_provider_for_screen(Gdk.Screen.get_default(), prov,
```

```
                                        Gtk.STYLE_PROVIDER_PRIORITY_APPLICATION)
col = Gtk.Box(orientation=Gtk.Orientation.VERTICAL, spacing=4)
col.get_style_context().add_class("osmo-run")
col.set_border_width(6)

head = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=8)
t = Gtk.Label(label=a.title)
t.get_style_context().add_class("title")
self.state = Gtk.Label(label="en cours ...")
self.state.get_style_context().add_class("state")
head.pack_start(t, False, False, 0)
head.pack_start(self.state, False, False, 0)
self.b_close = Gtk.Button(label="fermer")
self.b_close.connect("clicked", lambda *_x: self.destroy())
head.pack_end(self.b_close, False, False, 0)
self.rebours = None         # secondes avant fermeture, None = pas de rebours
self.b_garder = Gtk.Button(label="garder ouverte")
self.b_garder.set_tooltip_text("annule la fermeture automatique")
self.b_garder.connect("clicked", lambda *_x: self._garder())
head.pack_end(self.b_garder, False, False, 0)
if not a.tty:
    b_copy = Gtk.Button(label="copier")
    b_copy.set_tooltip_text("copie tout le journal dans le presse-papier")
    b_copy.connect("clicked", self._copier)
    head.pack_end(b_copy, False, False, 0)
self.b_stop = Gtk.Button(label="■ stop")
self.b_stop.set_tooltip_text("envoie SIGTERM a la commande")
self.b_stop.connect("clicked", lambda *_x: self.stop())
head.pack_end(self.b_stop, False, False, 0)
col.pack_start(head, False, False, 0)

cmdl = Gtk.Label(label="$ " + " ".join(shlex.quote(c) for c in a.cmd))
cmdl.get_style_context().add_class("cmd")
cmdl.set_halign(Gtk.Align.START)
cmdl.set_ellipsize(3)       # Pango.EllipsizeMode.END
col.pack_start(cmdl, False, False, 0)

# progress : la barre, l etape, la derniere ligne, puis la console
# REPLIEE (Gtk.Revealer) derriere le bouton « console »
if self.progress:
    self.bar = Gtk.ProgressBar()
    self.bar.set_show_text(True)
    self.bar.set_text("demarrage...")
    col.pack_start(self.bar, False, False, 2)
    self.l_etape = Gtk.Label(label="")
    self.l_etape.get_style_context().add_class("etape")
    self.l_etape.set_halign(Gtk.Align.START)
    self.l_etape.set_ellipsize(3)
    col.pack_start(self.l_etape, False, False, 0)
    self.l_der = Gtk.Label(label="")
    self.l_der.get_style_context().add_class("derniere")
    self.l_der.set_halign(Gtk.Align.START)
    self.l_der.set_ellipsize(3)
    col.pack_start(self.l_der, False, False, 0)
    self.b_console = Gtk.ToggleButton(label="▾ console")
    self.b_console.set_tooltip_text("montre / cache la sortie complete de la commande")
    self.b_console.set_halign(Gtk.Align.START)
    self.b_console.connect("toggled", self._toggle_console)
    col.pack_start(self.b_console, False, False, 0)
    self.rev = Gtk.Revealer()
    self.rev.set_transition_type(Gtk.RevealerTransitionType.SLIDE_DOWN)
    self.rev.set_reveal_child(False)
    col.pack_start(self.rev, True, True, 0)
    GLib.timeout_add(120, self._pulse)

# en progress, le terminal VTE sert de console des qu il existe (la
```

```
    # commande est dans un pty de toute facon) ; sinon le journal
    self.tty = bool((a.tty or self.progress) and Vte is not None)
    if self.tty:
        self.term = Vte.Terminal()
        self.term.set_scrollback_lines(20000)
        self.term.set_scroll_on_output(False)
        self.term.set_scroll_on_keystroke(True)
        try:
            from gi.repository import Pango
            self.term.set_font(Pango.FontDescription("Ubuntu Mono 10"))
        except Exception:
            pass
        fg, bg = Gdk.RGBA(), Gdk.RGBA()
        fg.parse("#d8e0ea")
        bg.parse("#0b0e14")
        self.term.set_colors(fg, bg, None)
        if self.progress:
            # ce qu on tape dans la console part au pty de la commande
            self.term.connect("commit", self._tty_commit)
            self.term.connect("size-allocate", self._tty_resize)
        else:
            self.term.connect("child-exited", self._tty_exit)
        sc = Gtk.ScrolledWindow()
        sc.set_policy(Gtk.PolicyType.NEVER, Gtk.PolicyType.AUTOMATIC)
        sc.add(self.term)
        (self.rev if self.progress else col).add(sc) if self.progress else col.pack_start(sc, True, True, 0)
    else:
        sc = Gtk.ScrolledWindow()
        sc.set_policy(Gtk.PolicyType.AUTOMATIC, Gtk.PolicyType.AUTOMATIC)
        self.tv = Gtk.TextView()
        self.tv.set_editable(False)
        self.tv.set_cursor_visible(False)
        self.tv.set_monospace(True)
        self.tv.set_wrap_mode(Gtk.WrapMode.WORD_CHAR)
        self.tv.set_left_margin(6)
        self.tv.set_right_margin(6)
        sc.add(self.tv)
        self.sc = sc
        if self.progress:
            self.rev.add(sc)
        else:
            col.pack_start(sc, True, True, 0)
    self.add(col)
    self.show_all()
    self.b_garder.hide()
    if a.tty and Vte is None:
        self._append("[osmo-gtk-run] VTE (gir1.2-vte-2.91) absent : journal au lieu du terminal, "
                     "les questions de la commande ne pourront pas etre repondues.\n")
    GLib.idle_add(self._start)

# ── lancer ──────────────────────────────────────────────────────
def _argv(self):
    return (_root_prefix(self.tty) if self.a.root else []) + list(self.a.cmd)

def _start(self):
    argv = self._argv()
    cwd = self.a.cwd or None
    if self.progress:
        return self._start_pty(argv, cwd)
    if self.tty:
        env = dict(os.environ)
        env.setdefault("TERM", "xterm-256color")
        envv = ["%s=%s" % kv for kv in env.items()]
        try:
            self.term.spawn_async(Vte.PtyFlags.DEFAULT, cwd, argv, envv,
                                  GLib.SpawnFlags.DEFAULT, None, None, -1, None,
```

```
                                    self._tty_spawned)
            except Exception as e:
                self.term.feed(("lancement impossible : %s\r\n" % e).encode())
                self._done(127)
            self.term.grab_focus()
            return False
        env = dict(os.environ, NO_COLOR="1", TERM="dumb")
        try:
            self.proc = subprocess.Popen(argv, cwd=cwd, stdin=subprocess.DEVNULL,
                                         stdout=subprocess.PIPE, stderr=subprocess.STDOUT,
                                         start_new_session=True, env=env)
        except OSError as e:
            self._append("lancement impossible : %s\n" % e)
            self._done(127)
            return False
        threading.Thread(target=self._pump, daemon=True).start()
        return False

    # ── progress : la commande dans un pty tenu ici ──────────────────────
    def _start_pty(self, argv, cwd):
        env = dict(os.environ)
        env.setdefault("TERM", "xterm-256color")
        try:
            master, slave = os.openpty()
            fcntl.ioctl(master, termios.TIOCSWINSZ, struct.pack("HHHH", 40, 120, 0, 0))

            def enfant():
                # son propre groupe + le pty comme terminal de controle
                os.setsid()
                try:
                    fcntl.ioctl(0, termios.TIOCSCTTY, 0)
                except OSError:
                    pass
            self.proc = subprocess.Popen(argv, cwd=cwd, stdin=slave, stdout=slave, stderr=slave,
                                         preexec_fn=enfant, env=env, close_fds=True)
            os.close(slave)
            self.master = master
        except OSError as e:
            self._append("lancement impossible : %s\n" % e)
            self._done(127)
            return False
        threading.Thread(target=self._pump_pty, daemon=True).start()
        return False

    def _pump_pty(self):
        while True:
            try:
                data = os.read(self.master, 4096)
            except OSError:            # EIO : la commande a ferme son cote
                break
            if not data:
                break
            GLib.idle_add(self._feed_pty, data)
        rc = self.proc.wait()
        GLib.idle_add(self._done, rc)

    def _feed_pty(self, data):
        text = data.decode("utf-8", "replace")
        if self.tty:
            self.term.feed(data)       # VTE lit l ANSI lui-meme
        else:
            self._append(text)
        # les reperes d avancement, ligne par ligne
        self._lbuf += ANSI.sub("", text)
        while "\n" in self._lbuf:
            line, self._lbuf = self._lbuf.split("\n", 1)
```

```
        self._ligne(line)
    return False

def _ligne(self, line):
    line = line.strip()
    if not line:
        return
    m = ETAPE.search(line)
    if m:
        n, t = int(m.group(1)), int(m.group(2))
        if t > 0:
            self.fraction = min(1.0, max(0.0, n / t))
            self.bar.set_fraction(self.fraction)
            self.bar.set_text("%d / %d" % (n, t))
        etape = (line[:m.start()] + line[m.end():]).strip(" =:-·.")
        self.l_etape.set_text(etape or line)
    else:
        self.l_der.set_text(line[:200])

def _pulse(self):
    if self.fini:
        return False
    if self.fraction is None:
        self.bar.pulse()
    return True

def _tty_commit(self, _term, text, _size):
    if self.master is not None:
        try:
            os.write(self.master, text.encode("utf-8", "replace"))
        except OSError:
            pass

def _tty_resize(self, term, _alloc):
    if self.master is None:
        return
    try:
        fcntl.ioctl(self.master, termios.TIOCSWINSZ,
                    struct.pack("HHHH", max(1, term.get_row_count()), max(1, term.get_column_count()), 0, 0))
    except OSError:
        pass

def _toggle_console(self, b):
    on = b.get_active()
    b.set_label("▴ console" if on else "▾ console")
    self.rev.set_reveal_child(on)
    if on:
        self.resize(self.a.width, self.a.height)
        if self.tty:
            self.term.grab_focus()
    else:
        GLib.timeout_add(300, lambda: (self.resize(*COMPACT), False)[1])

def _tty_spawned(self, _term, pid, error, *_a):
    if error is not None or not pid or pid < 0:
        self.term.feed(("lancement impossible : %s\r\n" % error).encode())
        self._done(127)
        return
    self.child_pid = pid

def _tty_exit(self, _term, status):
    # status : format waitpid
    rc = (status >> 8) & 0xff if (status & 0xff) == 0 else 128 + (status & 0x7f)
    self._done(rc)

def _pump(self):
```

```
    try:
        for chunk in iter(lambda: self.proc.stdout.read1(4096), b""):
            GLib.idle_add(self._append, chunk.decode("utf-8", "replace"))
    except Exception:
        pass
    rc = self.proc.wait()
    GLib.idle_add(self._done, rc)

# ── montrer ──────────────────────────────────────────────────────
def _append(self, text):
    if self.tty:
        self.term.feed(text.replace("\n", "\r\n").encode())
        return False
    buf = self.tv.get_buffer()
    adj = self.sc.get_vadjustment()
    en_bas = adj.get_value() + adj.get_page_size() >= adj.get_upper() - 40
    buf.insert(buf.get_end_iter(), ANSI.sub("", text))
    n = buf.get_char_count()
    if n > MAX_CHARS:
        buf.delete(buf.get_start_iter(), buf.get_iter_at_offset(n - MAX_CHARS))
    if en_bas:
        self.tv.scroll_to_mark(buf.get_insert(), 0.0, True, 0.0, 1.0)
    return False

def _done(self, rc):
    if self.fini:
        return False
    self.fini = True
    ctx = self.state.get_style_context()
    if rc == 0:
        self.state.set_text("termine")
        ctx.add_class("ok")
    else:
        self.state.set_text("termine (statut %d)" % rc)
        ctx.add_class("ko")
    if self.progress:
        if self._lbuf.strip():
            self._ligne(self._lbuf)
        if rc == 0:
            self.bar.set_fraction(1.0)
            self.bar.set_text("termine")
        else:
            self.bar.set_fraction(self.fraction if self.fraction is not None else 1.0)
            self.bar.set_text("echec (statut %d)" % rc)
            self.bar.get_style_context().add_class("ko")
            # l echec se lit dans la console : on la deplie
            if not self.b_console.get_active():
                self.b_console.set_active(True)
        if self.master is not None:
            try:
                os.close(self.master)
            except OSError:
                pass
            self.master = None
    self.b_stop.set_sensitive(False)
    if not self.a.keep_open:
        if rc == 0:
            GLib.timeout_add(1500, lambda: (self.destroy(), False)[1])
        else:
            self.rebours = 15
            self.b_garder.show()
            self._tic()
            GLib.timeout_add(1000, self._tic)
    return False

def _tic(self):
```

```
        """Echec : compte a rebours dans le bandeau, puis fermeture."""
        if self.rebours is None:
            return False
        if self.rebours <= 0:
            self.destroy()
            return False
        base = self.state.get_text().split(" - ")[0]
        self.state.set_text("%s - fermeture dans %d s" % (base, self.rebours))
        self.rebours -= 1
        return True

    def _garder(self):
        self.rebours = None
        self.state.set_text(self.state.get_text().split(" - ")[0])
        self.b_garder.hide()

    def _copier(self, *_a):
        buf = self.tv.get_buffer()
        texte = buf.get_text(buf.get_start_iter(), buf.get_end_iter(), False)
        Gtk.Clipboard.get(Gdk.SELECTION_CLIPBOARD).set_text(texte, -1)

    # ── arreter ──────────────────────────────────────────────────────
    def stop(self):
        if self.tty and not self.progress:
            if self.child_pid:
                try:
                    os.killpg(os.getpgid(self.child_pid), signal.SIGTERM)
                except OSError:
                    try:
                        os.kill(self.child_pid, signal.SIGTERM)
                    except OSError:
                        pass
                return
            if self.proc is not None and self.proc.poll() is None:
                try:
                    os.killpg(self.proc.pid, signal.SIGTERM)
                except OSError:
                    pass

    def _quit(self, *_a):
        # La fenetre part : la commande aussi. Un journalctl -f ou un tmux
        # attach orphelin n a aucune raison de survivre a ce qu on regardait.
        self.stop()
        Gtk.main_quit()


def main():
    Run(ARGS)
    GLib.unix_signal_add(GLib.PRIORITY_DEFAULT, signal.SIGTERM, lambda *_x: (Gtk.main_quit(), False)[1])
    Gtk.main()


if __name__ == "__main__":
    main()
```

3.416 /opt/GSM/osmo-operator/tools/osmo-info

958 octets, 19 lignes → 19 lignes

```
#!/bin/bash
# osmo-info - ouvre la page d aide du banc (le « ? » du dock).
#
# [2026-10-23] PAR HTTP, PAS PAR file://. Firefox est un snap : son confinement
# ne lit pas /usr/share, et l ancien Exec=xdg-open file:///usr/share/... ouvrait
```

```
# une page « fichier introuvable ». Le dashboard (8080) sert la page ; la copie
# de /usr/share reste en repli si le dashboard ne repond pas, recopiee alors
# dans ~/Documents (le snap ne lit pas les dossiers caches du home).
set -u
URL="${OSMO_INFO_URL:-http://127.0.0.1:8080/info.html}"
PORT="${URL##*:}"; PORT="${PORT%%/*}"
if (exec 3<>"/dev/tcp/127.0.0.1/${PORT}") 2>/dev/null; then
    exec 3>&- 2>/dev/null
    exec xdg-open "$URL"
fi
echo "Dashboard injoignable sur $PORT - repli sur une copie locale" >&2
D="$(xdg-user-dir DOCUMENTS 2>/dev/null || echo "$HOME/Documents")/osmo-operator"
mkdir -p "$D" && cp -f /usr/share/osmo-operator/info.html "$D/help.html" 2>/dev/null
exec xdg-open "file://$D/help.html"
```

3.417 /opt/GSM/osmo-operator/tools/osmo-launcher.py

28370 octets, 616 lignes → 616 lignes

```
#!/usr/bin/env python3
# osmo-launcher.py - LA BARRE DE LANCEMENT DU BAS, cliquable, categorisee.
#
# Meme mecanique de fenetre que osmo-panel.py / osmo-dino.py : une fenetre de
# type BUREAU (Gdk.WindowTypeHint.DESKTOP - sous les fenetres applicatives,
# au-dessus du fond, transparente, collante, non decoree), posee tout en BAS de
# l ecran. Elle rend, en RECTANGLES TRANSPARENTS groupes par categorie, tout ce
# qu on lance depuis le bureau du banc :
#
#   Banc      : run standalone · run multi · run deka · Dashboard · tmux · VTY
#   Jeux      : Doom (freedoom) · Quake (OpenArena) · Dino
#   Media     : Kodi · YouTube
#   Telephone : Linphone
#   Outils    : Wireshark (root)
#
# Chaque appli a TROIS gestes :
#   - clic sur le nom  -> lance l appli CALEE SUR LE CADRE FFT (fenetre normale
#                         deplacee/redimensionnee sur l encart, via wmctrl) ;
#   - bouton □  -> lance en PLEIN ECRAN ;
#   - bouton ⧉  -> lance et ENVOIE EN HAUT (zone LAB GSM) : ecrit une
#                         commande que osmo-topzone.py lit pour faire le fondu de
#                         la banniere et accueillir la fenetre.
#
# ── LES « run » N OUVRENT PLUS DE TERMINAL ──────────────────────────────
# [2026-10-01] run standalone / run multi / run deka ouvraient un gnome-terminal
# qui vivait le temps du lancement et dont la sortie partait avec lui. Ils
# tournent maintenant EN FOND, sans fenetre : leur sortie va dans un fichier
# ($XDG_RUNTIME_DIR/osmo-launcher/<nom>.log), et la barre se DEPLIE (bouton
# « ▴ journal », ou automatiquement au clic sur un run) pour montrer ce
# journal, en direct, au-dessus des boutons. Un bouton par run deja lance
# permet de passer de l un a l autre ; « ▾ » replie. Les privileges passent
# par pkexec (fenetre de mot de passe du bureau), comme launch.sh le fait
# deja pour les actions sans terminal.
#
# Le placement se fait en reperant la NOUVELLE fenetre apparue apres le
# lancement (diff des ids wmctrl) : robuste meme pour les applis qui forkent
# (firefox, kodi). Lance par /usr/local/bin/osmo-desktop-panel.
import os
import re
import shlex
import shutil
import signal
import subprocess
import sys
import time
```

```
# [2026-09-06] MEME PIEGE QUE osmo-panel.py / osmo-dino.py (voir leurs notes) :
# sous Wayland, GTK3 ignore move() et le type BUREAU, et une fois passe par
# XWayland le couple type BUREAU + « toujours dessous » fait manger les clics
# par la couche de bureau de GNOME Shell. Backend X11, fenetre normale.
X11_FORCE = os.environ.get("XDG_SESSION_TYPE") == "wayland" and bool(os.environ.get("DISPLAY"))
if X11_FORCE:
    os.environ["GDK_BACKEND"] = "x11"

import gi

gi.require_version("Gtk", "3.0")
gi.require_version("Gdk", "3.0")
from gi.repository import Gdk, GLib, Gtk  # noqa: E402

TYPE_HINT = Gdk.WindowTypeHint.NORMAL if X11_FORCE else Gdk.WindowTypeHint.DESKTOP
KEEP_BELOW = (not X11_FORCE) or os.environ.get("OSMO_DESKTOP_BELOW") == "1"

FW, FH = 1920, 1080
# L encart FFT (tools/wallpaper-render.py) : c est la que se cale une appli
# lancee "dans le panel". Meme boite que osmo-panel.py.
FFT_BOX = (510, 220, 900, 790)
# La zone haute LAB GSM (carte du fond) : la cible de "envoyer en haut".
TOP_BOX = (320, 60, 1110, 500)

RUN = os.environ.get("XDG_RUNTIME_DIR") or "/run/user/%d" % os.getuid()
RUN = RUN if os.path.isdir(RUN) else "/tmp"
PID_FILE = os.path.join(RUN, "osmo-launcher.pid")
# Le canal vers osmo-topzone.py : une ligne "HOST <winid>" ou "HOST FS <winid>".
TOPZONE_CMD = os.path.join(RUN, "osmo-topzone.cmd")
# Les journaux des « run » : un fichier par action, ecrase a chaque lancement.
LOG_DIR = os.path.join(RUN, "osmo-launcher")
LOG_H = 300                     # hauteur du journal deplie, en px
LOG_W_MAX = 1240
ANSI = re.compile(r"\x1b\[[0-9;?]*[A-Za-z]|\x1b\][^\x07]*\x07|\r")

REPO = os.environ.get("OSMO_REPO", "/opt/GSM/osmo-operator")
INFO_PAGE = "file:///usr/share/osmo-operator/info.html"
if not os.path.exists("/usr/share/osmo-operator/info.html"):
    INFO_PAGE = "file://" + os.path.join(REPO, "configs/info.html")

CSS = b"""
.osmo-launch { background: none; background-color: transparent; border: none;
               box-shadow: none; padding: 0; margin: 0; }
/* [2026-10-11] TRANSPARENCE TOTALE : les cadres de categorie n ont plus de fond ;
   textes et boutons portent une ombre pour rester lisibles sur le paysage. */
.osmo-cat { background: rgba(0,0,0,0); border: 1px solid rgba(239,230,210,0.18);
            border-radius: 10px; padding: 4px 8px; margin: 0 4px; }
.osmo-cat > label.title { color: #c9d4ee; font: 8pt "Ubuntu"; text-shadow: 0 1px 2px rgba(0,0,0,0.9); }
.osmo-app button { background: rgba(22,27,34,0.35); color: #e6edf3; text-shadow: 0 1px 2px rgba(0,0,0,0.9);
                   border: 1px solid rgba(88,166,255,0.25); border-radius: 6px;
                   padding: 1px 8px; font: 9pt "Ubuntu"; min-height: 0; }
.osmo-app button:hover { background: rgba(33,38,45,0.85); border-color: #58a6ff; }
.osmo-app button.mini { padding: 1px 5px; color: #a9b7de; font: 8pt "Ubuntu"; }
.osmo-app button.journal { color: #58a6ff; }
.osmo-help { background: #d62828; color: #ffffff; border: 2px solid #ff6b6b; border-radius: 999px;
             font: bold 22pt "Ubuntu"; min-width: 46px; min-height: 46px; padding: 0; margin: 0 6px; }
.osmo-help:hover { background: #ef3b3b; }
.osmo-log { background: rgba(10,12,18,0.93); border: 1px solid rgba(88,166,255,0.35);
            border-radius: 10px; padding: 6px 8px; margin: 0 0 6px 0; }
.osmo-log label.title { color: #58a6ff; font: bold 9pt "Ubuntu"; }
.osmo-log label.state { color: #8b949e; font: 8pt "Ubuntu"; }
.osmo-log button { background: rgba(22,27,34,0.7); color: #e6edf3;
                   border: 1px solid rgba(88,166,255,0.25); border-radius: 6px;
                   padding: 0 7px; font: 8pt "Ubuntu"; min-height: 0; }
.osmo-log button.actif { border-color: #58a6ff; color: #58a6ff; }
```

```
.osmo-log textview, .osmo-log textview text { background-color: transparent; color: #d8e0ea;
                  font-family: "Ubuntu Mono"; font-size: 9pt; }
"""


# ── LA GEOMETRIE DE L ECRAN (comme osmo-panel.py) ──────────────────────
def screen_geom():
    disp = Gdk.Display.get_default()
    mon = disp.get_primary_monitor() or disp.get_monitor(0)
    g = mon.get_geometry()
    sw, sh = g.width, g.height
    try:
        from gi.repository import Gio
        mode = Gio.Settings.new("org.gnome.desktop.background").get_string("picture-options")
    except Exception:
        mode = "zoom"
    if mode == "stretched":
        sx, sy, ox, oy = sw / FW, sh / FH, 0, 0
    else:
        s = min(sw / FW, sh / FH) if mode == "scaled" else max(sw / FW, sh / FH)
        sx = sy = s
        ox, oy = (FW * s - sw) / 2, (FH * s - sh) / 2
    return (g.x, g.y, sw, sh), (sx, sy, ox, oy)


def box_to_screen(box):
    (gx, gy, sw, sh), (sx, sy, ox, oy) = screen_geom()
    bx, by, bw, bh = box
    return (gx + int(bx * sx - ox), gy + int(by * sy - oy), int(bw * sx), int(bh * sy))


# ── wmctrl : PLACER LA NOUVELLE FENETRE ────────────────────────────────
def _wmctrl_ids():
    try:
        out = subprocess.run(["wmctrl", "-l"], capture_output=True, text=True, timeout=3).stdout
        return {ln.split()[0] for ln in out.splitlines() if ln.strip()}
    except Exception:
        return set()


def _place(winid, geom):
    x, y, w, h = geom
    subprocess.run(["wmctrl", "-i", "-r", winid, "-b",
                    "remove,maximized_vert,maximized_horz,fullscreen"], timeout=3)
    subprocess.run(["wmctrl", "-i", "-r", winid, "-e", f"0,{x},{y},{w},{h}"], timeout=3)


def _fullscreen(winid):
    subprocess.run(["wmctrl", "-i", "-r", winid, "-b", "add,fullscreen"], timeout=3)


def spawn(argv):
    return subprocess.Popen(argv, stdin=subprocess.DEVNULL, stdout=subprocess.DEVNULL,
                            stderr=subprocess.DEVNULL, start_new_session=True)


GTK_RUN = os.path.join(os.path.dirname(os.path.abspath(__file__)), "osmo-gtk-run.py")


def _terminal_argv(cmd, titre="Banc GSM"):
    """argv qui ouvre `cmd` (chaine shell) dans un terminal VTE pose DANS une
    fenetre GTK du banc (osmo-gtk-run.py --tty). [2026-10-01] Plus
    d emulateur de terminal : aucune entree de la barre n en ouvre."""
    return [sys.executable, GTK_RUN, "--tty", "--title", titre, "--", "bash", "-lc", cmd]
```

```
def _root_prefix():
    """Prefixe argv pour executer en root SANS terminal : rien si on l est deja,
    pkexec (fenetre de mot de passe du bureau) sinon, sudo -n en dernier recours
    (il ne peut pas demander : sans NOPASSWD il echoue net, et le journal le dit)."""
    if os.geteuid() == 0:
        return []
    if shutil.which("pkexec") and os.environ.get("DISPLAY"):
        return ["pkexec", "env", "DISPLAY=" + os.environ.get("DISPLAY", ""),
                "XAUTHORITY=" + os.environ.get("XAUTHORITY", ""), "NO_COLOR=1", "TERM=dumb"]
    return ["sudo", "-n", "-E"]


LAUNCHER = None     # la fenetre, pour que les Action puissent lui confier un journal


class Action:
    """Une appli : comment la lancer - fenetre, terminal, ou run en fond avec journal."""
    def __init__(self, argv=None, term_cmd=None, fs_argv=None, log_cmd=None, root=False, slug=None):
        self.argv = argv          # fenetre graphique normale
        self.term_cmd = term_cmd  # commande a ouvrir dans un terminal
        self.fs_argv = fs_argv    # variante plein ecran si l appli la gere
        self.log_cmd = log_cmd    # commande shell en fond, sortie dans un journal (pas de terminal)
        self.root = root          # log_cmd : en root (pkexec)
        self.slug = slug          # nom du journal
        self.nom = slug or ""     # le libelle du bouton (titre de la fenetre tty)

    def launch(self, mode):
        """mode: 'frame' (cale sur l encart) | 'full' (plein ecran) | 'top'."""
        if self.log_cmd is not None:
            if LAUNCHER is not None:
                LAUNCHER.run_logged(self)
            return
        before = _wmctrl_ids()
        if self.term_cmd is not None:
            spawn(_terminal_argv(self.term_cmd, self.nom or "Banc GSM"))
        elif mode == "full" and self.fs_argv:
            spawn(self.fs_argv)
        else:
            spawn(self.argv)
        GLib.timeout_add(400, self._settle, before, mode, 0)

    def _settle(self, before, mode, tries):
        new = _wmctrl_ids() - before
        # on ignore nos propres fenetres bureau
        new = {w for w in new}
        if not new:
            if tries < 25:  # ~10 s
                GLib.timeout_add(400, self._settle, before, mode, tries + 1)
            return False
        winid = sorted(new)[-1]
        if mode == "full":
            _fullscreen(winid)
        elif mode == "top":
            try:
                with open(TOPZONE_CMD + ".tmp", "w") as f:
                    f.write("HOST %s\n" % winid)
                os.replace(TOPZONE_CMD + ".tmp", TOPZONE_CMD)
            except OSError:
                _place(winid, box_to_screen(TOP_BOX))
        else:  # frame
            _place(winid, box_to_screen(FFT_BOX))
        return False


# ── LE CATALOGUE ──────────────────────────────────────────────────────
def _sudo():
```

```
    return "" if os.geteuid() == 0 else "sudo -E "


def catalogue():
    tmux = os.environ.get("TMUX_SESSION", "calypso")
    vty = os.environ.get("MS_VTY_PORT", "")
    if not vty:
        try:                        # 4347 en --dsp : start-native.sh publie le port
            vty = open("/run/osmo-banc/ms1-vty").read().strip() or "4247"
        except OSError:
            vty = "4247"
    dash = os.environ.get("DASH_PORT", "8080")
    browser = "firefox"
    for b in ("firefox", "chromium", "xdg-open"):
        if shutil.which(b):
            browser = b
            break
    # Les trois « run » : en fond, journal dans la barre (voir l en-tete).
    #   standalone : launch.sh --service demarre osmo-banc.service et le DIT
    #                (notification) ; on deroule ensuite le journal de l unite.
    #   multi      : launch.sh --multi (osmo-multi.service, plusieurs minutes)
    #                EN PARALLELE du journal de l unite, qui montre les
    #                conteneurs monter - lance en sequence, le journal n aurait
    #                commence qu a la fin du demarrage.
    #   deka       : deka-start.sh (pose par addition.sh --opencl), qui tee deja
    #                dans /var/log/deka.log.
    run_standalone = (f"{REPO}/launch.sh --service; echo; "
                      "exec journalctl -u osmo-banc -f -n 40 --no-pager -o cat")
    run_multi = (f"{REPO}/launch.sh --multi & "
                 "journalctl -u osmo-multi -f -n 20 --no-pager -o cat & wait")
    run_deka = ("if [ -x /root/deka/deka-start.sh ]; then exec /root/deka/deka-start.sh; "
                "else echo 'deka non installe : icone « Supplements » (addition.sh --opencl)'; fi")
    return [
        ("Banc", [
            ("run standalone", Action(log_cmd=run_standalone, root=True, slug="standalone")),
            ("run multi",      Action(log_cmd=run_multi, root=True, slug="multi")),
            ("run deka",       Action(log_cmd=run_deka, root=True, slug="deka")),
            ("Dashboard",      Action(argv=[browser, f"http://127.0.0.1:{dash}"])),
            # tmux, VTY, oFono, pmOS : interactifs, donc un terminal - VTE, dans
            # une fenetre GTK du banc, qui reste ouverte a la fin (plus de read).
            ("tmux",           Action(term_cmd=f"tmux attach -t {tmux} || tmux -S /tmp/osmocom_tmux attach -t
↪ osmocom || echo 'pas de session tmux'")),
           (f"VTY {vty}",     Action(term_cmd=f"telnet 127.0.0.1 {vty} || {{ echo; echo 'VTY injoignable'; }}")),
        ]),
        # Doom et Quake ne sont poses que par addition.sh --extras (pas dans
        # l ISO) : une entree n apparait que si son binaire est la.
        ("Jeux", [j for j in [
            ("Doom",   Action(argv=["/usr/games/chocolate-doom", "-iwad", "/usr/share/games/doom/freedoom2.wad",
↪ "-window"],
                              fs_argv=["/usr/games/chocolate-doom", "-iwad", "/usr/share/games/doom/freedoom2.wad",
↪ "-fullscreen"])),
            ("Quake",  Action(argv=["/usr/games/openarena", "+set", "r_fullscreen", "0"],
                              fs_argv=["/usr/games/openarena", "+set", "r_fullscreen", "1"])),
            ("Dino",   Action(argv=["/usr/local/bin/osmo-dino-play"],
                              fs_argv=["/usr/local/bin/osmo-dino-play", "--fullscreen"])),
        ] if not j[1].argv[0].startswith("/usr/games/") or os.path.exists(j[1].argv[0])]),
        ("Media", [
            ("Kodi",    Action(argv=["kodi"], fs_argv=["kodi", "--fullscreen"])),
            ("YouTube", Action(argv=["/usr/local/bin/osmo-youtube"],
                               fs_argv=["/usr/local/bin/osmo-youtube", "--fullscreen"])),
        ]),
        ("Telephone", [
            ("Linphone", Action(argv=["sh", "-c", "linphone || linphone-desktop"])),
            ("oFono",    Action(term_cmd="/usr/local/bin/osmo-ofono")),
            # Le telephone du banc : une VM postmarketOS/Phosh, avec son
            # ModemManager branche sur le modem du banc (tools/osmo-pmos.sh).
```

```
            ("pmOS",     Action(term_cmd="/usr/local/bin/osmo-pmos up")),
        ]),
        ("Outils", [
            ("Wireshark", Action(argv=["/usr/local/bin/osmo-wireshark-root"])),
        ]),
    ]


class Launcher(Gtk.Window):
    def __init__(self):
        super().__init__(title="osmo-launcher")
        (gx, gy, sw, sh), _ = screen_geom()
        self.geo = (gx, gy, sw, sh)
        self.set_type_hint(TYPE_HINT)
        self.set_decorated(False)
        self.set_resizable(False)
        self.set_keep_below(KEEP_BELOW)
        self.set_skip_taskbar_hint(True)
        self.set_skip_pager_hint(True)
        self.stick()
        screen = self.get_screen()
        visual = screen.get_rgba_visual()
        if visual and screen.is_composited():
            self.set_visual(visual)
        self.set_app_paintable(True)
        self.connect("draw", self._draw)
        self.connect("destroy", self._quit)

        prov = Gtk.CssProvider()
        prov.load_from_data(CSS)
        Gtk.StyleContext.add_provider_for_screen(screen, prov, Gtk.STYLE_PROVIDER_PRIORITY_APPLICATION)

        # -- les journaux des run ------------------------------------------
        self.runs = {}          # slug -> {"proc": Popen|None, "titre": str, "log": chemin, "bouton": Gtk.Button}
        self.cur = None         # slug affiche
        self._shown = ""        # dernier texte pousse dans la vue
        self.deplie = False
        os.makedirs(LOG_DIR, exist_ok=True)

        col = Gtk.Box(orientation=Gtk.Orientation.VERTICAL, spacing=0)
        col.get_style_context().add_class("osmo-launch")
        col.set_valign(Gtk.Align.END)
        self.revealer = Gtk.Revealer()
        self.revealer.set_transition_type(Gtk.RevealerTransitionType.SLIDE_UP)
        self.revealer.set_transition_duration(180)
        self.revealer.add(self._log_box(sw))
        col.pack_start(self.revealer, False, False, 0)

        row = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=2)
        row.set_halign(Gtk.Align.CENTER)
        row.set_valign(Gtk.Align.END)
        for cat, apps in catalogue():
            row.pack_start(self._cat_box(cat, apps), False, False, 0)
        # Le grand « ? » rouge : la page Info (chaque module, icone / barre / CLI).
        aide = Gtk.Button(label="?")
        aide.get_style_context().add_class("osmo-help")
        aide.set_valign(Gtk.Align.CENTER)
        aide.set_tooltip_text("Info : comment lancer chaque module (icone, barre, CLI)")
        aide.connect("clicked", lambda *_a: spawn(["xdg-open", INFO_PAGE]))
        row.pack_end(aide, False, False, 0)
        col.pack_start(row, False, False, 0)
        self.add(col)

        # bande basse, pleine largeur, ~72 px ; depliee : + le journal
        self.h = 74
        self._taille(self.h)
```

```
    self.show_all()
    self._taille(self.h)
    GLib.timeout_add(700, self._refresh_log)

def _taille(self, h):
    gx, gy, sw, sh = self.geo
    self.set_size_request(sw, h)
    self.resize(sw, h)
    self.move(gx, gy + sh - h)

def _draw(self, _w, cr):
    import cairo
    cr.set_operator(cairo.OPERATOR_SOURCE)
    cr.set_source_rgba(0, 0, 0, 0)
    cr.paint()
    return False

def _quit(self, *_a):
    self.stop_runs()
    Gtk.main_quit()

# ── le journal : la boite au-dessus des boutons ──────────────────────
def _log_box(self, sw):
    box = Gtk.Box(orientation=Gtk.Orientation.VERTICAL, spacing=4)
    box.get_style_context().add_class("osmo-log")
    box.set_halign(Gtk.Align.CENTER)
    box.set_size_request(min(sw - 60, LOG_W_MAX), LOG_H)
    head = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=6)
    self.lbl_titre = Gtk.Label(label="journal")
    self.lbl_titre.get_style_context().add_class("title")
    self.lbl_etat = Gtk.Label(label="")
    self.lbl_etat.get_style_context().add_class("state")
    head.pack_start(self.lbl_titre, False, False, 0)
    head.pack_start(self.lbl_etat, False, False, 0)
    self.onglets = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=4)
    head.pack_end(self._bouton("▾ replier", self.replier), False, False, 0)
    b_stop = self._bouton("■ stop", self._stop_courant)
    b_stop.set_tooltip_text("arrete la commande en cours (pas le banc : launch.sh --stop)")
    head.pack_end(b_stop, False, False, 0)
    head.pack_end(self.onglets, False, False, 0)
    box.pack_start(head, False, False, 0)
    sc = Gtk.ScrolledWindow()
    sc.set_policy(Gtk.PolicyType.AUTOMATIC, Gtk.PolicyType.AUTOMATIC)
    self.tv = Gtk.TextView()
    self.tv.set_editable(False)
    self.tv.set_cursor_visible(False)
    self.tv.set_monospace(True)
    self.tv.set_wrap_mode(Gtk.WrapMode.WORD_CHAR)
    self.tv.set_left_margin(6)
    self.tv.set_right_margin(6)
    sc.add(self.tv)
    box.pack_start(sc, True, True, 0)
    return box

@staticmethod
def _bouton(label, cb):
    b = Gtk.Button(label=label)
    b.connect("clicked", lambda *_a: cb())
    return b

def deplier(self, slug=None):
    if slug:
        self.cur = slug
        for s, r in self.runs.items():
            ctx = r["bouton"].get_style_context()
            (ctx.add_class if s == slug else ctx.remove_class)("actif")
```

```
        self.lbl_titre.set_text("journal : " + self.runs[slug]["titre"])
        self._shown = ""
    if not self.deplie:
        self.deplie = True
        self._taille(self.h + LOG_H + 10)
        self.revealer.set_reveal_child(True)
    self._refresh_log()

def replier(self):
    if not self.deplie:
        return
    self.deplie = False
    self.revealer.set_reveal_child(False)
    GLib.timeout_add(200, lambda: (self._taille(self.h), False)[1])

def basculer(self):
    if self.deplie:
        self.replier()
    elif self.runs:
        self.deplier(self.cur or next(iter(self.runs)))
    else:
        self.lbl_titre.set_text("journal : aucun run lance")
        self.deplier()

# ── lancer un run en fond ──────────────────────────────────────
def run_logged(self, action):
    slug = action.slug or re.sub(r"\W+", "-", action.log_cmd)[:24]
    log = os.path.join(LOG_DIR, slug + ".log")
    r = self.runs.get(slug)
    if r is None:
        b = self._bouton(slug, lambda s=slug: self.deplier(s))
        self.onglets.pack_start(b, False, False, 0)
        b.show()
        r = self.runs[slug] = {"proc": None, "titre": slug, "log": log, "bouton": b}
    self._stop(r)
    # [2026-10-12] 2G (standalone) ou 4G (multi) : l antenne se dresse dans
    # le fond d ecran des le clic (tools/osmo-fond.py, fondu) - osmo-panel.py
    # fera de meme quand il verra le banc debout, sans effet s il l est deja.
    if slug in ("standalone", "multi"):
        fond = os.path.join(REPO, "tools", "osmo-fond.py")
        if os.path.exists(fond):
            spawn([sys.executable, fond, "antenne"])
    argv = (_root_prefix() if action.root else []) + ["bash", "-lc", action.log_cmd]
    env = dict(os.environ, NO_COLOR="1", TERM="dumb")
    try:
        fh = open(log, "w")
        fh.write("$ %s\n" % action.log_cmd)
        fh.flush()
        r["proc"] = subprocess.Popen(argv, stdin=subprocess.DEVNULL, stdout=fh, stderr=subprocess.STDOUT,
                                     start_new_session=True, env=env)
        fh.close()
    except OSError as e:
        with open(log, "a") as fh:
            fh.write("lancement impossible : %s\n" % e)
    self.deplier(slug)

@staticmethod
def _stop(r):
    p = r.get("proc")
    if p is None or p.poll() is not None:
        r["proc"] = None
        return
    try:
        os.killpg(p.pid, signal.SIGTERM)
    except OSError:
        pass
```

```
    r["proc"] = None

def _stop_courant(self):
    if self.cur and self.cur in self.runs:
        self._stop(self.runs[self.cur])
        with open(self.runs[self.cur]["log"], "a") as fh:
            fh.write("\n[stop]\n")

def stop_runs(self):
    for r in self.runs.values():
        self._stop(r)

def _refresh_log(self):
    if not self.deplie or not self.cur or self.cur not in self.runs:
        return True
    r = self.runs[self.cur]
    p = r.get("proc")
    if p is not None and p.poll() is not None:
        self.lbl_etat.set_text("termine (statut %d)" % p.returncode)
    elif p is not None:
        self.lbl_etat.set_text("en cours ...")
    else:
        self.lbl_etat.set_text("")
    try:
        with open(r["log"], "rb") as fh:
            fh.seek(0, os.SEEK_END)
            taille = fh.tell()
            fh.seek(max(0, taille - 48 * 1024))
            data = fh.read().decode("utf-8", "replace")
    except OSError:
        data = ""
    data = ANSI.sub("", data)
    if data != self._shown:
        self._shown = data
        buf = self.tv.get_buffer()
        buf.set_text(data)
        fin = buf.get_end_iter()
        self.tv.scroll_to_iter(fin, 0.0, False, 0.0, 1.0)
    return True

# ── les boutons ──────────────────────────────────────────────────
def _cat_box(self, cat, apps):
    box = Gtk.Box(orientation=Gtk.Orientation.VERTICAL, spacing=2)
    box.get_style_context().add_class("osmo-cat")
    lbl = Gtk.Label(label=cat)
    lbl.get_style_context().add_class("title")
    lbl.set_halign(Gtk.Align.START)
    box.pack_start(lbl, False, False, 0)
    approw = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=3)
    for name, action in apps:
        approw.pack_start(self._app_widget(name, action), False, False, 0)
    if cat == "Banc":
        wrap = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=0)
        wrap.get_style_context().add_class("osmo-app")
        bj = Gtk.Button(label="▴ journal")
        bj.get_style_context().add_class("journal")
        bj.set_tooltip_text("deplie / replie le journal des run")
        bj.connect("clicked", lambda *_a: self.basculer())
        wrap.pack_start(bj, False, False, 0)
        approw.pack_start(wrap, False, False, 0)
    box.pack_start(approw, False, False, 0)
    return box

def _app_widget(self, name, action):
    wrap = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=0)
    wrap.get_style_context().add_class("osmo-app")
```

```
        b = Gtk.Button(label=name)
        action.nom = name
        b.connect("clicked", lambda *_a: action.launch("frame"))
        wrap.pack_start(b, False, False, 0)
        if action.log_cmd is not None:
            # un run n a pas de fenetre a caler ni a envoyer en haut : un seul bouton
            b.set_tooltip_text(f"{name} - en fond, journal dans la barre")
            return wrap
        b.set_tooltip_text(f"{name} - clic: cale sur le cadre")
        # □ plein ecran
        bf = Gtk.Button(label="□")
        bf.get_style_context().add_class("mini")
        bf.set_tooltip_text("plein ecran")
        bf.connect("clicked", lambda *_a: action.launch("full"))
        wrap.pack_start(bf, False, False, 0)
        # ⇧ envoyer en haut (zone LAB GSM)
        bt = Gtk.Button(label="⇧")
        bt.get_style_context().add_class("mini")
        bt.set_tooltip_text("envoyer en haut (zone LAB GSM)")
        bt.connect("clicked", lambda *_a: action.launch("top"))
        wrap.pack_start(bt, False, False, 0)
        return wrap


def single_instance():
    try:
        old = int(open(PID_FILE).read().strip())
        if old != os.getpid():
            os.kill(old, signal.SIGTERM)
            time.sleep(0.3)
    except (OSError, ValueError):
        pass
    try:
        with open(PID_FILE, "w") as f:
            f.write(str(os.getpid()))
    except OSError as e:
        print("osmo-launcher : verrou %s non ecrit (%s)" % (PID_FILE, e), file=sys.stderr)


if __name__ == "__main__":
    single_instance()
    LAUNCHER = Launcher()
    # SIGTERM (relance par le gardien du bureau, single_instance) : les
    # journalctl -f lances en fond ne doivent pas survivre a la barre.
    GLib.unix_signal_add(GLib.PRIORITY_DEFAULT, signal.SIGTERM, lambda *_a: (LAUNCHER._quit(), False)[1])
    Gtk.main()
```

3.418 /opt/GSM/osmo-operator/tools/osmo-lte-install.sh

34703 octets, 543 lignes → 543 lignes

```
#!/bin/bash
# ============================================================================
# osmo-lte-install.sh - LA 4G DU BANC, POSEE UNE FOIS POUR LES TROIS CHEMINS.
#
# [2026-09-08] srsRAN (ZeroMQ) et Open5GS vivaient HORS du depot : /opt/LTE
# compile a la main, Open5GS dans /root/open5gs, les configs dans
# ~/.config/srsran de root, open5gs-epc.sh dans /usr/local/bin. Rien de tout
# cela n arrivait dans l ISO, ni sur une machine passee par addition.sh. Ce
# script est l endroit UNIQUE ou la 4G est posee ; les trois chemins l appellent :
#   - addition.sh            (une machine deja installee)      : --all
#   - update.sh              (le rattrapage)                    : --configs --launchers
#   - iso_modules/72-lte-pmos.sh (le build ISO, dans le chroot) : --deps --configs --launchers
#     (les binaires y viennent des .deb du build docker : osmo-build-srsran,
```

```
#      osmo-build-open5gs, poses par 50-injection-image.sh)
#
# OU VIVENT LES CHOSES :
#   /opt/LTE/srsGUI             srsGUI (Qt5 + Qwt) : les traces temps reel de
#                               srsenb / srsue. COMPILE AVANT srsRAN - son cmake
#                               le cherche a sa configuration - et installe en
#                               /usr/local/lib/libsrsgui
#   /opt/LTE/srsRAN_4G          les sources et le build srsRAN (ZeroMQ, GUI),
#                               binaires dans /usr/local/bin (srsenb, srsue, srsepc)
#   /opt/LTE/open5gs            les sources Open5GS ; prefixe d installation
#                               /opt/LTE/open5gs/install (bin, etc/open5gs, var/log)
#   /root/.config/srsran        enb.conf ue.conf sib.conf rr.conf rb.conf user_db.csv
#                               = configs/srsran du depot (srs* lisent le HOME de root)
#   /usr/local/bin/osmo-lte     le lanceur (tools/osmo-lte.sh)
#   /usr/local/bin/osmo-epc     le coeur (tools/osmo-epc.sh), alias open5gs-epc.sh
#
#   osmo-lte-install                = --deps --debs --configs --launchers
#   osmo-lte-install --all          idem, plus --build si les binaires manquent
#   osmo-lte-install --build        compile srsRAN et Open5GS dans /opt/LTE
#   osmo-lte-install --deps         les paquets apt (runtime + build), mongodb-org
#   osmo-lte-install --debs         pose les .deb du cache (/var/cache/osmo-debs)
#   osmo-lte-install --configs      pose les configs du depot
#   osmo-lte-install --launchers    osmo-lte, osmo-epc dans /usr/local/bin
#                                   (+ l unite open5gs-webui, activee)
#   osmo-lte-install --webui        construit le WebUI Open5GS (console :9999)
#   osmo-lte-install --force        (avec --configs) ecrase les configs existantes
#
# Idempotent, non fatal paquet par paquet. Utilisable en session ET dans le
# chroot de l ISO (pas de systemd : les systemctl y echouent sans arreter).
# ============================================================================
set -u

REPO="${OSMO_REPO:-${DIR:-/opt/GSM/osmo-operator}}"
LTE="${OSMO_LTE_ROOT:-/opt/LTE}"
O5GS="$LTE/open5gs"
O5GS_PREFIX="${OPEN5GS_PREFIX:-$O5GS/install}"
SRS="$LTE/srsRAN_4G"
SRSGUI="$LTE/srsGUI"
SRS_DIR="${OSMO_SRSRAN_DIR:-/root/.config/srsran}"
DEB_CACHE="${OSMO_DEB_CACHE:-/var/cache/osmo-debs}"
SRS_REF="${OSMO_SRSRAN_REF:-release_25_10}"
# srsGUI n a ni tag ni release : c est master. La version du .deb est donc
# nommee ici, une fois, et les trois chemins la reprennent (Dockerfile,
# packaging/snapshot-lte-debs.sh, lte_debs).
SRSGUI_REF="${OSMO_SRSGUI_REF:-master}"
SRSGUI_VER="${OSMO_SRSGUI_VER:-0.1+git}"
O5GS_REF="${OSMO_OPEN5GS_REF:-v2.8.0}"
JOBS="${JOBS:-$(nproc)}"
# ── L ISA DE srsRAN : PORTABLE, PAS « native » ──────────────────────────
# [2026-09-14] LA PANNE : sur une machine autre que celle du build, le lanceur
# mourait aussitot, deux fois de suite, sans un mot dans ses logs :
#     /usr/local/bin/osmo-lte: line 204: 14521 Illegal instruction (core dumped) ... srsenb
#     /usr/local/bin/osmo-lte: line 244: 14661 Illegal instruction (core dumped) ... srsue
# Ce n est ni une config ni une dependance : c est le CPU. Le CMakeLists de
# srsRAN_4G met GCC_ARCH a « native » par defaut (l.110-116), et son
# cmake/modules/FindSSE.cmake decide de -mavx/-mavx2/-mfma/-mavx512 par des
# check_c_source_runs - c est-a-dire en EXECUTANT le test sur la machine qui
# compile. Les binaires sortent donc tailles pour ce processeur-la : le srsenb
# de l ISO portait 19 353 instructions %ymm (AVX/AVX2) et 417 FMA. Sur un CPU
# sans AVX2 - une VM qui n expose pas le jeu complet, un portable plus ancien,
# un runner de CI different - la premiere de ces instructions fait SIGILL.
# Une ISO se promene par definition d une machine a l autre : elle ne peut pas
# etre compilee « pour la machine du build ».
# On fige donc une base commune (SSE4.1/SSE4.2, tout x86_64 depuis 2009 - et
# srsRAN exige de toute facon SSE4.1, cf. son « no SIMD instructions found »)
# et on coupe les detections AVX/AVX2/FMA/AVX512, qui ne mesurent que l hote.
```

```
# OSMO_SRSRAN_ARCH=native pour retrouver l ancien comportement (binaires plus
# rapides, valables sur CETTE machine seulement) ; toute autre valeur est
# passee telle quelle a -march= (x86-64, nehalem, haswell...). Un -march que
# le gcc local ne connait pas est ignore par ADD_C_COMPILER_FLAG_IF_AVAILABLE :
# on retombe alors sur la base x86-64 du compilateur, portable elle aussi.
SRS_ARCH="${OSMO_SRSRAN_ARCH:-x86-64-v2}"

# ── LA VERSION DU .deb srsRAN PORTE SON ISA ──────────────────────────────
# [2026-09-14] LE .deb EMPOISONNE QUI REVENAIT A CHAQUE BUILD. La version
# etait « 25.10+zmq » : rien n y disait POUR QUEL PROCESSEUR le binaire avait
# ete compile. Un osmo-build-srsran_25.10+zmq~noble_amd64.deb fabrique avant
# le 14/09 (donc en -march=native, sur un runner de CI ou sur la machine de
# reference) satisfaisait donc « osmo-deb install srsran 25.10+zmq », et
# l image repartait avec un srsenb plein d AVX. L ISO ne s en apercevait qu au
# controle objdump de iso_modules/88-lte-pmos.sh, qui l ecartait et relancait
# une compilation d une heure dans le chroot - A CHAQUE BUILD, parce que le
# .deb fautif revenait du cache (sur GitHub : restore-keys « osmo-debs-noble- »
# de .github/workflows/build-iso.yml, qui repeche la derniere entree meme
# quand la cle exacte a change).
# L ISA fait maintenant partie de la version : un binaire natif ne peut plus
# etre confondu avec un binaire portable, le cache garde les deux sans qu ils
# se marchent dessus, et rien ne se recompile une fois le bon .deb la.
srs_deb_version() {
    local v a
    v="${SRS_REF#release_}"; v="${v//_/.}"
    a="$(printf '%s' "$SRS_ARCH" | tr -c 'A-Za-z0-9.' '-')"   # osmo-deb : ^[0-9][A-Za-z0-9.+~-]*$
    printf '%s+zmq+%s\n' "$v" "$a"
}

: "${GREEN:=}"; : "${YELLOW:=}"; : "${CYAN:=}"; : "${RED:=}"; : "${BOLD:=}"; : "${NC:=}"
_l_say()  { echo -e "  ${CYAN}→${NC} $*"; }
_l_ok()   { echo -e "      ${GREEN}✓${NC} $*"; }
_l_warn() { echo -e "      ${YELLOW}!${NC} $*"; }
_l_err()  { echo -e "      ${RED}✗${NC} $*" >&2; }

# ── CE srsenb-LA TOURNERA-T-IL AILLEURS QU ICI ? ─────────────────────────
# [2026-09-14] Un srsenb deja pose - compile avant aujourd hui, ou venu d un
# .deb du cache fabrique sur une AUTRE machine - reste taille pour le CPU qui
# l a compile, et rien ne le trahit : il existe, ses bibliotheques sont
# resolues, --help repond. On ne l apprend qu au premier vrai lancement, par
# un « Illegal instruction (core dumped) » sans une ligne de log (voir
# SRS_ARCH en tete). On regarde donc ce que le binaire CONTIENT : une seule
# instruction %ymm (AVX/AVX2) ou %zmm (AVX512) le condamne hors de sa machine
# d origine. Sans objdump (binutils absent) on ne juge pas : on garde ce qui
# est la, et on ne declenche pas une recompilation d une heure sur un doute.
_srs_isa_portable() {
    local b; b="$(command -v srsenb 2>/dev/null)" || return 0
    [ -n "$b" ] || return 0
    command -v objdump >/dev/null 2>&1 || return 0
    ! objdump -d "$b" 2>/dev/null | grep -qE '%[yz]mm[0-9]'
}

# ── LES PAQUETS ──────────────────────────────────────────────────────────
# Runtime (ce que ldd de srsenb et d open5gs-mmed reclame) ET build : l ISO
# est un atelier, on doit pouvoir refaire « make » dans /opt/LTE. mongodb-org
# vient du depot de MongoDB (noble/8.0) : la cle et la source sont posees si
# elles manquent. Non fatal paquet par paquet.
_lte_apt_mongo_source() {
    local k=/usr/share/keyrings/mongodb-server-8.0.gpg
    local s=/etc/apt/sources.list.d/mongodb-org-8.0.list
    [ -s "$s" ] && [ -s "$k" ] && return 0
    command -v curl >/dev/null 2>&1 && command -v gpg >/dev/null 2>&1 || { _l_warn "curl/gpg absents : pas de
    ↪  depot MongoDB"; return 1; }
    if curl -fsSL --retry 3 https://www.mongodb.org/static/pgp/server-8.0.asc | gpg --dearmor --yes -o "$k"
    ↪  2>/dev/null; then
```

```
        echo "deb [ arch=amd64,arm64 signed-by=$k ] https://repo.mongodb.org/apt/ubuntu noble/mongodb-org/8.0
         ↪  multiverse" > "$s"
        _l_ok "depot MongoDB 8.0 pose ($s)"
    else
        _l_warn "cle MongoDB injoignable : mongodb-org ne s installera pas (HSS/PCRF sans base)"
        return 1
    fi
}

lte_deps() {
    export DEBIAN_FRONTEND=noninteractive
    _l_say "paquets de la 4G (srsRAN ZeroMQ + Open5GS + MongoDB)"
    _lte_apt_mongo_source || true
    apt-get update -y >/dev/null 2>&1 || true
    local p
    # qtbase5/qwt/boost : srsGUI (les traces temps reel de srsenb et srsue).
    # Sans eux le cmake de srsGUI echoue et srsRAN se compile sans traces - ca
    # n arrete rien, mais on ne verrait jamais la constellation.
    # [2026-09-10] libpcsclite-dev : srsUE lit une VRAIE carte SIM par PC/SC
    # (srsue/hdr/stack/upper/pcsc_usim.h -> winscard.h). Il n etait dans aucune
    # liste : la machine de reference l avait, l arbre srsRAN livre par
    # osmo-build-srsran porte donc un CMakeCache ou PCSC est ACTIF, et toute
    # recompilation ailleurs tombait net -
    #     pcsc_usim.h:29:10: fatal error: winscard.h: No such file or directory
    # C est exactement ce qui est arrive en recompilant pour srsGUI.
    for p in libzmq5 libzmq3-dev libboost-program-options-dev libmbedtls-dev libconfig++-dev libfftw3-dev
     ↪  libsctp-dev lksctp-tools cmake \
            qtbase5-dev libqt5opengl5-dev libqwt-qt5-dev libboost-system-dev libboost-thread-dev
 ↪  libboost-test-dev \
            libpcsclite-dev \
            meson ninja-build flex bison libgnutls28-dev libgcrypt20-dev libssl-dev libidn-dev libmongoc-dev
 ↪  libbson-dev \
            libyaml-dev libnghttp2-dev libmicrohttpd-dev libcurl4-gnutls-dev libtins-dev libtalloc-dev
 ↪  libc-ares-dev \
            mongodb-org mongodb-mongosh mongodb-database-tools; do
        dpkg -s "$p" >/dev/null 2>&1 && continue
        apt-get install -y --no-install-recommends "$p" >/dev/null 2>&1 \
            && _l_ok "$p" || _l_warn "$p indisponible (apt) - ignore"
    done
    # mongod : ACTIVE au boot (osmo-epc le lance de toute facon s il dort).
    # Dans un chroot sans systemd, enable pose le lien tout de meme.
    systemctl enable mongod >/dev/null 2>&1 && _l_ok "mongod active au demarrage" || true
}

# ── LES .deb DU BUILD DOCKER ──────────────────────────────────────────
# osmo-build-libzmq, osmo-build-srsran, osmo-build-open5gs : produits par le
# Dockerfile (osmo-deb pack) ou par un snapshot de la machine de reference
# (packaging/snapshot-lte-debs.sh). Poses seulement si les binaires manquent.
lte_debs() {
    local n deb suite
    suite="$(. /etc/os-release 2>/dev/null; echo "${VERSION_CODENAME:-noble}")"
    # srsgui AVANT srsran : c est la bibliotheque des traces, et srsRAN la
    # cherche a SA configuration (voir lte_build).
    for n in libzmq srsgui srsran open5gs; do
        case "$n" in
            libzmq)  [ -e /usr/local/lib/libzmq.so.5 ] || [ -e /usr/lib/x86_64-linux-gnu/libzmq.so.5 ] && continue
             ↪  ;;
            srsgui)  [ -e /usr/local/lib/libsrsgui.so ] || [ -e /usr/local/lib/libsrsgui.a ] && continue ;;
            srsran)  command -v srsenb >/dev/null 2>&1 && continue ;;
            open5gs) [ -x "$O5GS_PREFIX/bin/open5gs-mmed" ] && continue ;;
        esac
        deb="$(ls -1 "$DEB_CACHE"/osmo-build-${n}_*"~${suite}_$(dpkg --print-architecture).deb" 2>/dev/null | tail
         ↪  -1)"
        [ -n "$deb" ] || { _l_warn "$n : pas de binaire ni de .deb dans $DEB_CACHE (osmo-lte-install --build)";
         ↪  continue; }
```

```
        dpkg -i --force-overwrite "$deb" >/dev/null 2>&1 && _l_ok "$n pose depuis $(basename "$deb")" || _l_warn
        ↪  "dpkg -i $(basename "$deb") a echoue"
    done
    ldconfig 2>/dev/null || true
}

# ── LA COMPILATION, DANS /opt/LTE ──────────────────────────────────────
# srsRAN_4G avec ZeroMQ (libzmq3-dev d Ubuntu suffit : plus besoin de compiler
# libzmq et czmq a la main), Open5GS par meson avec le prefixe /opt/LTE/open5gs/
# install. Sous OSMO_DEB=1 (le Dockerfile), chaque etape sort en .deb via
# osmo-deb pack ; sinon make/ninja install direct.
lte_build() {
    mkdir -p "$LTE"
    local pack=""
    [ "${OSMO_DEB:-0}" = "1" ] && command -v osmo-deb >/dev/null 2>&1 && pack=1
    # ── LES VERSIONS DES .deb : UNE SEULE ECRITURE, POUR TOUT LE MONDE ──────
    # [2026-09-09] Elles etaient calculees a l arrache dans les deux appels a
    # `osmo-deb pack` ci-dessous, et les deux etaient FAUSSES :
    #   srsran  : "${SRS_REF#release_}+zmq" donnait 25_10+zmq - un souligne,
    #             que osmo-deb refuse (^[0-9][A-Za-z0-9.+~-]*$) : « version
    #             invalide : '25_10+zmq' ». Et comme `pack` meurt AVANT de
    #             lancer la commande qu on lui confie, le `make install`
    #             n avait jamais lieu : srsRAN compilait 3 minutes pour rien.
    #   open5gs : "${O5GS_REF#v}" donnait 2.8.0, sans le +git - le .deb sortait
    #             sous un nom que le `osmo-deb install open5gs 2.8.0+git` du
    #             Dockerfile ne retrouvait jamais : recompilation a chaque build.
    # Les noms qui font foi sont ceux du Dockerfile et de packaging/
    # snapshot-lte-debs.sh : osmo-build-srsran_25.10+zmq, osmo-build-open5gs_
    # 2.8.0+git. On les derive ici des memes refs git, en points.
    local srs_ver o5gs_ver
    srs_ver="$(srs_deb_version)"                                  # release_25_10 -> 25.10+zmq+x86-64-v2
    o5gs_ver="${O5GS_REF#v}+git"                                  # v2.8.0       -> 2.8.0+git
    # ── srsGUI D ABORD : C EST srsRAN QUI LE CHERCHE ────────────────────────
    # [2026-09-10] srsRAN_4G etait configure avec -DENABLE_GUI=OFF : ni srsenb
    # ni srsue n avaient leurs traces temps reel (constellation, spectre,
    # PDSCH), celles que montre la doc srsRAN. Elles viennent de srsGUI
    # (github.com/srsran/srsGUI, Qt5 + Qwt), une bibliotheque A PART - et le
    # cmake de srsRAN la cherche AU MOMENT DE SA PROPRE CONFIGURATION. Posee
    # apres, elle n aurait servi a rien : elle se construit donc ICI, juste
    # avant, et srsRAN passe a ENABLE_GUI=ON dessous.
    #
    # NON FATAL, ET C EST VOLONTAIRE : sans Qt, sans reseau, ou si le cmake de
    # srsGUI tombe, on previent et on continue. ENABLE_GUI=ON ne casse alors
    # rien - le cmake de srsRAN constate l absence et n arme pas les traces.
    # Un banc sans fenetre de constellation reste un banc ; un banc qui ne
    # compile plus, non.
    if [ ! -e /usr/local/lib/libsrsgui.so ] && [ ! -e /usr/local/lib/libsrsgui.a ]; then
        _l_say "srsGUI $SRSGUI_REF -> $SRSGUI (les traces de srsenb / srsue)"
        if [ ! -d "$SRSGUI/.git" ]; then
            git clone --depth 1 -b "$SRSGUI_REF" https://github.com/srsran/srsGUI "$SRSGUI" >/dev/null 2>&1 \
              || git clone --depth 1 https://github.com/srsran/srsGUI "$SRSGUI" >/dev/null 2>&1 \
              || _l_warn "srsGUI : clone impossible - srsRAN se compilera sans traces"
        fi
        if [ -d "$SRSGUI" ]; then
            # ── -DBOOST_TIMER_ENABLE_DEPRECATED : SANS LUI, RIEN NE SORT ────
            # [2026-09-10] srsGUI ne compile pas sur Boost 1.83 (noble) :
            #     /usr/include/boost/progress.hpp:23:3: error: #error This
            #     header is deprecated and will be removed.
            # Ce sont ses TESTS (test/cxx/*_test.cpp) qui incluent
            # boost/progress.hpp et boost/timer.hpp, pas la bibliotheque - mais
            # `make install` passe par la cible `all`, tests compris : un test
            # qui ne compile pas, et c est l installation entiere qui n a pas
            # lieu. Boost dit lui-meme quoi faire (« You can define
            # BOOST_TIMER_ENABLE_DEPRECATED to suppress this error ») : on le
            # definit, plutot que de retirer des cibles du projet amont.
            ( cd "$SRSGUI" && mkdir -p build && cd build \
```

```
          && cmake -DCMAKE_BUILD_TYPE=Release \
                   -DCMAKE_CXX_FLAGS=-DBOOST_TIMER_ENABLE_DEPRECATED .. >/dev/null 2>&1 \
          && make -j"$JOBS" >/dev/null 2>&1 \
          && if [ -n "$pack" ]; then OSMO_DEB_SRC_ROOT="$LTE" OSMO_DEB_SRC="$SRSGUI" osmo-deb pack srsgui
           ↪  "$SRSGUI_VER" make install
             else make install >/dev/null 2>&1; fi ) \
          && { ldconfig; _l_ok "srsGUI : /usr/local/lib/libsrsgui"; } \
          || _l_warn "srsGUI : compilation echouee - srsRAN se compilera sans traces"
    fi
else
    _l_ok "srsGUI deja la (/usr/local/lib/libsrsgui)"
fi
# ── UN srsenb SANS TRACES NE SE CORRIGE PAS TOUT SEUL ────────────────────
# [2026-09-10] La condition etait « srsenb existe-t-il ? ». Une machine qui
# avait deja srsRAN - compile avant aujourd hui, ou pose par osmo-build-
# srsran - gardait donc pour toujours des binaires sans traces, meme avec
# libsrsgui fraichement installee dix lignes plus haut : on n en aurait rien
# vu, et la seule facon de s en sortir aurait ete d effacer srsenb a la
# main. On regarde ce a quoi le binaire est LIE, pas seulement s il existe.
_srs_lie_gui() {
    local b; b="$(command -v srsenb 2>/dev/null)" || return 1
    [ -n "$b" ] && ldd "$b" 2>/dev/null | grep -q libsrsgui
}
if ! command -v srsenb >/dev/null 2>&1 \
   || { [ -e /usr/local/lib/libsrsgui.so ] && ! _srs_lie_gui; } \
   || { [ "$SRS_ARCH" != native ] && ! _srs_isa_portable; }; then
    if command -v srsenb >/dev/null 2>&1; then
        if [ "$SRS_ARCH" != native ] && ! _srs_isa_portable; then
            _l_say "srsRAN_4G : recompilation, srsenb porte de l AVX (taille pour le CPU du build, SIGILL
             ↪  ailleurs)"
        else
            _l_say "srsRAN_4G : recompilation, srsenb n est pas lie a srsGUI"
        fi
    else
        _l_say "srsRAN_4G $SRS_REF (ZeroMQ, -march=$SRS_ARCH) -> $SRS"
    fi
    if [ ! -d "$SRS/.git" ]; then
        git clone --depth 1 -b "$SRS_REF" https://github.com/srsran/srsRAN_4G "$SRS" || { _l_err "clone
         ↪  srsRAN_4G"; return 1; }
    fi
    # ── LE CACHE CMAKE MENT SUR L ISA ──────────────────────────────────────
    # HAVE_AVX2, HAVE_FMA, HAVE_AVX512 sont des variables de CACHE posees
    # par les check_c_source_runs de FindSSE.cmake. -DENABLE_AVX2=OFF
    # empeche le TEST de tourner, il n efface pas sa reponse d hier : sur
    # un arbre deja configure, le `if (HAVE_AVX2)` du CMakeLists relirait
    # le cache et remettrait -mavx2. On repart donc d un build vide - de
    # toute facon les drapeaux changent, tous les objets sont a refaire.
    rm -rf "$SRS/build"
    # -DGCC_ARCH : la base commune (voir SRS_ARCH en tete). Les quatre
    # ENABLE_* coupent les detections qui ne mesurent que l hote ; SSE4.1
    # reste actif (ENABLE_SSE=ON par defaut), sans quoi le cmake s arrete
    # sur « no SIMD instructions found » et srsRAN perd son -Ofast.
    _srs_isa=(-DGCC_ARCH="$SRS_ARCH" -DENABLE_AVX=OFF -DENABLE_AVX2=OFF -DENABLE_FMA=OFF -DENABLE_AVX512=OFF)
    [ "$SRS_ARCH" = native ] && _srs_isa=(-DGCC_ARCH=native)
    ( cd "$SRS" && mkdir -p build && cd build \
     && cmake -DENABLE_ZEROMQ=ON -DENABLE_GUI=ON -DENABLE_UHD=OFF -DENABLE_BLADERF=OFF -DENABLE_SOAPYSDR=OFF \
              "${_srs_isa[@]}" \
              -DCMAKE_BUILD_TYPE=Release .. >/dev/null \
      && make -j"$JOBS" >/dev/null \
      && if [ -n "$pack" ]; then OSMO_DEB_SRC_ROOT="$LTE" OSMO_DEB_SRC="$SRS" osmo-deb pack srsran "$srs_ver"
       ↪  make install
         else make install >/dev/null; fi ) \
      && { ldconfig; _l_ok "srsRAN : $(command -v srsenb)"; } || { _l_err "srsRAN : compilation echouee";
       ↪  return 1; }
else
    _l_ok "srsRAN deja la ($(command -v srsenb))$(_srs_lie_gui && echo ' + traces srsGUI')"
```

```
    fi
    if [ ! -x "$O5GS_PREFIX/bin/open5gs-mmed" ]; then
        _l_say "Open5GS $O5GS_REF -> $O5GS (prefixe $O5GS_PREFIX)"
        if [ ! -d "$O5GS/.git" ]; then
            git clone --depth 1 -b "$O5GS_REF" https://github.com/open5gs/open5gs "$O5GS" || { _l_err "clone
             ↪  open5gs"; return 1; }
        fi
        ( cd "$O5GS" \
          && { [ -f build/build.ninja ] || meson setup build --prefix="$O5GS_PREFIX" >/dev/null; } \
          && ninja -C build -j"$JOBS" >/dev/null \
          && if [ -n "$pack" ]; then OSMO_DEB_SRC_ROOT="$LTE" OSMO_DEB_SRC="$O5GS" osmo-deb pack open5gs
           ↪  "$o5gs_ver" ninja -C build install
             else ninja -C build install >/dev/null; fi ) \
          && { ldconfig; _l_ok "Open5GS : $O5GS_PREFIX/bin"; } || { _l_err "Open5GS : compilation echouee"; return
           ↪  1; }
    else
        _l_ok "Open5GS deja la ($O5GS_PREFIX/bin/open5gs-mmed)"
    fi
}

# ── LES CERTIFICATS freeDiameter ──────────────────────────────────────
# [2026-09-09] Les pairs Diameter du banc sont en No_TLS, mais freeDiameter
# charge TLS_Cred / TLS_CA a l init et refuse de demarrer sans. Open5GS livre
# les siens (configs/open5gs/tls de son depot) par le meme install_conf qui
# ne fait rien sous DESTDIR (voir lte_configs) : on fabrique donc les notres,
# une fois (CA auto-signee + un certificat par demon), sans toucher a ceux
# qui existent.
lte_tls() {
    local t="$O5GS_PREFIX/etc/open5gs/tls" n
    command -v openssl >/dev/null 2>&1 || { _l_warn "openssl absent : pas de certificats freeDiameter (apt install
     ↪  openssl)"; return 0; }
    mkdir -p "$t"
    if [ ! -s "$t/ca.crt" ] || [ ! -s "$t/ca.key" ]; then
        openssl req -x509 -newkey rsa:2048 -nodes -days 3650 -subj "/CN=ca.localdomain" \
            -keyout "$t/ca.key" -out "$t/ca.crt" >/dev/null 2>&1 || { _l_warn "CA freeDiameter : openssl a
             ↪  echoue"; return 0; }
    fi
    for n in mme hss pcrf smf; do
        [ -s "$t/$n.crt" ] && [ -s "$t/$n.key" ] && continue
        openssl req -newkey rsa:2048 -nodes -subj "/CN=$n.localdomain" -keyout "$t/$n.key" -out "$t/$n.csr"
         ↪  >/dev/null 2>&1 \
        && openssl x509 -req -in "$t/$n.csr" -CA "$t/ca.crt" -CAkey "$t/ca.key" -CAcreateserial -days 3650 -out
         ↪  "$t/$n.crt" >/dev/null 2>&1 \
        || _l_warn "certificat freeDiameter $n : openssl a echoue"
        rm -f "$t/$n.csr"
    done
    chmod 600 "$t"/*.key 2>/dev/null || true
}

# ── LES CONFIGS ──────────────────────────────────────────────────────
# Celles du depot, telles qu elles tournent sur le banc de reference :
#   configs/srsran/*   -> /root/.config/srsran   (eNB 0x19B, TAC 7, EARFCN 3350,
#                         MME 127.0.0.2, ZeroMQ 2000/2001 a 11.52e6, SIB3+SIB7,
#                         UE 001010001000001, APN srsapn, netns ue1)
#   configs/open5gs/*.yaml -> $PREFIX/etc/open5gs (MME 127.0.0.2, SGs vers
#                         OsmoMSC 127.0.0.1, TAC 7 / LAC 1, UE 10.45.0.0/16)
# Une config existante n est pas ecrasee (les reglages vivants du banc), sauf
# --force : on la garde en .bak-osmo a cote.
lte_configs() {
    local force="${1:-0}" f dst
    mkdir -p "$SRS_DIR"
    for f in "$REPO"/configs/srsran/*; do
        [ -f "$f" ] || continue
        dst="$SRS_DIR/$(basename "$f")"
        if [ -f "$dst" ] && [ "$force" != "1" ]; then continue; fi
        [ -f "$dst" ] && ! cmp -s "$f" "$dst" && cp -a "$dst" "$dst.bak-osmo"
```

```
        install -m644 "$f" "$dst"
    done
    _l_ok "srsRAN : configs dans $SRS_DIR ($(ls "$SRS_DIR" | grep -c '\.conf$') .conf, user_db.csv)"
    # [2026-09-09] LA CONDITION TESTAIT LE MAUVAIS OBJET. Elle demandait que
    # $PREFIX/etc/open5gs EXISTE DEJA - c est a dire que le `ninja install`
    # d Open5GS ait pose ses yaml d exemple. Un Open5GS venu du .deb du cache,
    # ou une installation dont le etc/ a ete nettoye (l ISO lite fait le
    # menage dans /opt/LTE), avait donc ses binaires mais jamais nos configs :
    #     ! Open5GS pas installe (...) : ses configs seront posees apres --build
    # alors qu open5gs-mmed etait la, juste a cote. On teste desormais LE
    # BINAIRE - la seule chose qui dise si Open5GS est installe - et on cree
    # le etc/ nous-memes (le mkdir juste en dessous le faisait deja).
    if [ -x "$O5GS_PREFIX/bin/open5gs-mmed" ] || [ -d "$O5GS_PREFIX/etc/open5gs" ] || [ "$force" = "1" ]; then
        mkdir -p "$O5GS_PREFIX/etc/open5gs" "$O5GS_PREFIX/var/log/open5gs"
        for f in "$REPO"/configs/open5gs/*.yaml; do
            [ -f "$f" ] || continue
            dst="$O5GS_PREFIX/etc/open5gs/$(basename "$f")"
            # Les yaml portent le prefixe /opt/LTE/open5gs/install : un autre
            # prefixe (OPEN5GS_PREFIX) est substitue a la volee.
           if [ -f "$dst" ] && [ "$force" != "1" ] && grep -q 'osmo-operator' "$dst" 2>/dev/null; then continue; fi
            [ -f "$dst" ] && ! cmp -s "$f" "$dst" && cp -a "$dst" "$dst.bak-osmo"
            sed "s#/opt/LTE/open5gs/install#$O5GS_PREFIX#g" "$f" > "$dst"
        done
        _l_ok "Open5GS : configs dans $O5GS_PREFIX/etc/open5gs"
        # [2026-09-09] freeDiameter ET les certificats TLS : ce que « ninja
        # install » ne pose PAS sous DESTDIR (install_conf de meson.build :
        # rien si DESTDIR est defini), donc jamais dans le .deb osmo-build-
        # open5gs - et le MME, le HSS, le PCRF et le SMF mouraient a l init
        # sur l ISO (« Unable to open configuration file … freeDiameter/
        # mme.conf ») : pas de S1AP, ConnectionReject, pas d attach. Voir
        # configs/open5gs/freeDiameter/mme.conf. Le dossier des .fdx depend
        # du triplet (lib/x86_64-linux-gnu sur amd64) : substitue aussi.
        local fdd fdlib
        fdd="$O5GS_PREFIX/etc/freeDiameter"; mkdir -p "$fdd"
        fdlib="$(ls -d "$O5GS_PREFIX"/lib/*/freeDiameter "$O5GS_PREFIX"/lib/freeDiameter 2>/dev/null | head -1)"
        for f in "$REPO"/configs/open5gs/freeDiameter/*.conf; do
            [ -f "$f" ] || continue
            dst="$fdd/$(basename "$f")"
           if [ -f "$dst" ] && [ "$force" != "1" ] && grep -q 'osmo-operator' "$dst" 2>/dev/null; then continue; fi
            [ -f "$dst" ] && ! cmp -s "$f" "$dst" && cp -a "$dst" "$dst.bak-osmo"
            sed "s#/opt/LTE/open5gs/install/lib/x86_64-linux-gnu/freeDiameter#${fdlib:-$O5GS_PREFIX/lib/x86_64-
             ↪  linux-gnu/freeDiameter}#g; s#/opt/LTE/open5gs/install#$O5GS_PREFIX#g" "$f" >
             ↪  "$dst"
        done
        lte_tls
        _l_ok "freeDiameter : $(ls "$fdd" 2>/dev/null | grep -c '\.conf$') .conf dans $fdd, TLS dans
         ↪  $O5GS_PREFIX/etc/open5gs/tls"
    else
        _l_warn "Open5GS pas installe (pas de $O5GS_PREFIX/bin/open5gs-mmed) : ses configs seront posees apres
         ↪  --build / --debs"
    fi
}

# ── LES LANCEURS ──────────────────────────────────────────────────────
lte_launchers() {
    local t
    for t in osmo-lte osmo-epc; do
        [ -f "$REPO/tools/$t.sh" ] || { _l_warn "$REPO/tools/$t.sh absent"; continue; }
        chmod 755 "$REPO/tools/$t.sh"
        ln -sfn "$REPO/tools/$t.sh" "/usr/local/bin/$t"
    done
    # L ancien nom du coeur, garde : les habitudes et les notes du banc.
    ln -sfn "$REPO/tools/osmo-epc.sh" /usr/local/bin/open5gs-epc.sh
    _l_ok "lanceurs : /usr/local/bin/osmo-lte, osmo-epc (alias open5gs-epc.sh)"
    # ── LA CONSOLE DES ABONNES 4G : POSEE ET ACTIVEE PAR DEFAUT ──────────
    # [2026-09-10] Comme le tableau de bord 2G (osmo-egprs-web). L unite ne
```

```
    # demarre pas pour autant tant que le WebUI n est pas construit : son
    # ConditionPathExists la saute, sans echec. « --webui » le construit.
    # Une unite MASQUEE est laissee tranquille : c est le moyen de dire non.
    if [ -f "$REPO/services/open5gs-webui.service" ]; then
        install -m644 "$REPO/services/open5gs-webui.service" \
                /etc/systemd/system/open5gs-webui.service 2>/dev/null || true
        systemctl daemon-reload 2>/dev/null || true
        if [ "$(systemctl is-enabled open5gs-webui 2>/dev/null)" != masked ]; then
            systemctl enable open5gs-webui >/dev/null 2>&1 || true
        fi
        _l_ok "console 4G : open5gs-webui.service posee et activee (:9999)"
    fi
}

# ── LE WEBUI D OPEN5GS : LA CONSOLE DES ABONNES 4G ──────────────────────
# [2026-09-10] Provisionner un abonne LTE se faisait a la main dans MongoDB (ou
# par « osmo-epc subscribers », qui rejoue le JSON du depot). Le WebUI est fait
# pour ca et vit dans les SOURCES d Open5GS (webui/), pas dans le prefixe
# d installation : une machine posee depuis les .deb du banc
# (osmo-build-open5gs) ne l a donc PAS. On le recupere en sparse checkout - le
# depot open5gs entier pour un repertoire de Next.js serait absurde - et on le
# construit sur place. C est du npm : ca demande le reseau, et ca prend
# quelques minutes. D ou une sous-commande a part, hors du chemin par defaut.
lte_webui() {
    local dir="$O5GS/webui" tmp="$LTE/.open5gs-webui-src"
    command -v node >/dev/null 2>&1 && command -v npm >/dev/null 2>&1 \
        || { _l_warn "WebUI Open5GS : node/npm absents - rien a construire"; return 1; }
    if [ ! -f "$dir/package.json" ]; then
        _l_say "WebUI Open5GS $O5GS_REF -> $dir"
        rm -rf "$tmp"
        # sparse : on ne tire que webui/ (et les blobs a la demande).
        if git clone --depth 1 --filter=blob:none --sparse -b "$O5GS_REF" \
               https://github.com/open5gs/open5gs "$tmp" >/dev/null 2>&1 \
           && ( cd "$tmp" && git sparse-checkout set webui >/dev/null 2>&1 ) \
           && [ -d "$tmp/webui" ]; then
            mkdir -p "$O5GS"
            rm -rf "$dir"; mv "$tmp/webui" "$dir"; rm -rf "$tmp"
        else
            rm -rf "$tmp"; _l_err "WebUI Open5GS : recuperation des sources echouee"; return 1
        fi
    fi
    # `npm install` et pas `npm ci --omit=dev` : Next.js compile AVEC ses
    # dependances de developpement, et le depot ne livre pas toujours un
    # package-lock a jour.
    ( cd "$dir" && npm install --no-audit --no-fund >/dev/null 2>&1 \
      && npm run build >/dev/null 2>&1 ) \
        || { _l_err "WebUI Open5GS : npm install / build echoue (voir $dir)"; return 1; }
    [ -f "$dir/server/index.js" ] || { _l_err "WebUI Open5GS : pas de server/index.js apres build"; return 1; }
    _l_ok "WebUI Open5GS construit ($dir) - console des abonnes sur :9999"
    systemctl restart open5gs-webui >/dev/null 2>&1 || true
}

osmo_lte_install() {
    local deps=0 debs=0 configs=0 launchers=0 build=0 all=0 force=0 webui=0 a rc=0
    for a in "$@"; do case "$a" in
        --deps) deps=1 ;; --debs) debs=1 ;; --configs) configs=1 ;; --launchers) launchers=1 ;;
        --build) build=1 ;; --all) all=1 ;; --force) force=1 ;; --webui) webui=1 ;;
        # Le Dockerfile demande le nom au lieu de le recopier : une seule
        # ecriture, et changer OSMO_SRSRAN_ARCH suffit a changer les deux.
        --srs-deb-version) srs_deb_version; return 0 ;;
        *) echo "osmo-lte-install : option inconnue $a" >&2; return 2 ;;
    esac; done
    if [ $((deps + debs + configs + launchers + build + all + webui)) -eq 0 ]; then deps=1; debs=1; configs=1;
     ↪  launchers=1; fi
    if [ "$all" = 1 ]; then deps=1; debs=1; configs=1; launchers=1; fi
    [ "$deps" = 1 ] && lte_deps
```

```
    [ "$debs" = 1 ] && lte_debs
    # [2026-09-14] « ! command -v srsenb » ne suffit pas a decider de compiler :
    # un srsenb PRESENT mais taille pour le CPU d une autre machine passait au
    # travers de --all (donc d addition.sh et d update.sh), et la 4G mourait en
    # « Illegal instruction » chez l operateur. La meme question qu ailleurs :
    # ce binaire tournera-t-il ICI ?
    if [ "$build" = 1 ] || { [ "$all" = 1 ] && { ! command -v srsenb >/dev/null 2>&1 \
            || { [ "$SRS_ARCH" != native ] && ! _srs_isa_portable; } \
            || [ ! -x "$O5GS_PREFIX/bin/open5gs-mmed" ]; }; }; then
        # [2026-09-09] Le `|| true` d ici MENTAIT au Dockerfile : la compilation
        # tombait (MbedTLS absent), osmo-lte-install sortait 0, le « ECHEC build
        # srsRAN » n etait jamais imprime et l image n echouait que 200 lignes
        # plus bas sur le `test -x`. On garde la suite (configs, lanceurs : le
        # banc doit rester utilisable sans la 4G) mais le code de retour dit
        # desormais la verite. Les appelants encadrent deja (`|| true`,
        # `|| inst_hint`) : addition.sh, update.sh, install_modules/50-build.sh.
        lte_build || rc=1
    fi
    [ "$configs" = 1 ] && lte_configs "$force"
    [ "$launchers" = 1 ] && lte_launchers
    # Le WebUI passe par npm et par le reseau : jamais dans le chemin par
    # defaut, seulement quand on le demande (ou avec --all, qui est deja le
    # chemin long).
    if [ "$webui" = 1 ] || [ "$all" = 1 ]; then lte_webui || rc=1; fi
    return $rc
}

# Source (addition.sh, update.sh) ou execute (ISO, ligne de commande).
if [ "${BASH_SOURCE[0]}" = "$0" ]; then
    [ "$(id -u)" -eq 0 ] || { echo "osmo-lte-install : root requis" >&2; exit 1; }
    osmo_lte_install "$@"
fi
```

3.419 /opt/GSM/osmo-operator/tools/osmo-lte.sh

20541 octets, 401 lignes → 401 lignes

```
#!/bin/bash
# ============================================================================
# osmo-lte.sh - LA 4G DU BANC : Open5GS (le coeur), srsENB et srsUE sur la
# radio ZeroMQ, lances DANS L ORDRE ET DANS LES TEMPS.
#
# [2026-09-07] POURQUOI UN LANCEUR, alors que trois lignes de commande
# suffisent. Parce que les trois lignes ont deux pieges, et qu on est tombe
# dans les deux le meme jour :
#
#   1. « fail_on_disconnect=true » sur l eNB. C est l argument de la doc, et il
#      fait ce qu il dit : quand l eNB ne recoit AUCUN echantillon de l UE
#      pendant 2000 ms (ZMQ_TIMEOUT_MS, rf_zmq_imp_trx.h), sa radio S ARRETE.
#      Le processus reste, le S1 vers le coeur reste, mais plus rien ne sort
#      sur ZeroMQ - on l a mesure : une centaine d octets par sens, la poignee
#      de main ZMTP et rien d autre. Un srsUE lance UNE MINUTE apres l eNB
#      trouve donc une radio morte : « Attaching UE... » et jamais « Found
#      Cell ». Ici l UE part une demi-seconde apres l eNB.
#
#   2. Le SIB3. sib.conf programme les SIB emis par « si_mapping_info » ; en
#      y mettant [ 7 ] (le SIB7, la liste GERAN du CSFB) A LA PLACE de [ 3 ],
#      on a retire le SIB3. Or srsUE ne juge une cellule « convenable » QUE
#      s il a le SIB3 (srsue rrc.cc, cell_selection_criteria : has_sib3()) :
#      « Failed to configure serving cell », T3410 expire, cinq essais, rien.
#      Il faut [ 3, 7 ] - on le verifie avant de lancer.
#
# [2026-09-08] LE COEUR EST OPEN5GS, plus srsepc (tools/osmo-epc.sh : mongod,
# ogstun, les huit demons, le SGs vers OsmoMSC pour le CSFB). Le MME ecoute
```

```
# le S1 sur 127.0.0.2 - c est ce que dit enb.conf (mme_addr). Et le netns
# « ue1 » de l UE vit dans /run : il disparait au reboot, on le recree ici.
#
# LES CONFIGS SONT CELLES DU DEPOT (configs/srsran/*.conf, user_db.csv),
# posees dans ~/.config/srsran de root par tools/osmo-lte-install.sh. Elles
# portent la radio ZeroMQ (device_name / device_args : ports 2000/2001,
# base_srate 11.52e6) : on ne les redit PAS sur la ligne de commande, sauf
# OSMO_LTE_SRATE pour forcer une autre cadence. Les srs* sont lances DEPUIS
# ce repertoire : sib.conf / rr.conf / rb.conf y sont resolus, et le .ctxt
# (le contexte NAS que srsue sauve dans son cwd) y vit aussi.
#
#   osmo-lte start      coeur (osmo-epc), puis eNB, puis UE (refuse si l un tourne deja)
#   osmo-lte start-gui  LE MEME LANCEMENT, AVEC LES TRACES srsGUI a l ecran
#                       (constellation, spectre, PDSCH) : --gui.enable sur l eNB
#                       et l UE, et un DISPLAY. Jamais par systemd - une unite
#                       systeme n a pas d ecran ; c est un lancement direct.
#   osmo-lte traces     LES TRACES SEULES : relance eNB et UE avec srsGUI, SANS
#                       toucher au coeur Open5GS (qui reste debout, abonnes
#                       compris). srsGUI n est pas une appli a part : ce sont
#                       des fenetres Qt ouvertes par srsenb/srsue eux-memes
#                       (--gui.enable), on ne peut pas les accrocher a un eNB
#                       deja lance - d ou la relance de la radio, et d elle seule.
#                       Rien ne tourne ? C est alors un start-gui complet.
#   osmo-lte stop       arrete eNB et UE (le coeur reste : « osmo-epc stop »)
#   osmo-lte restart    stop puis start
#                       [2026-10-12] start / stop / restart hors root : le
#                       script SE RELANCE par pkexec (repli sudo) au lieu de
#                       refuser - « osmo-lte stop » tape dans un terminal de
#                       session arrete bien la 4G.
#   osmo-lte status     processus, S1, debit ZeroMQ, adresse de l UE
#   osmo-lte log        les journaux, en direct
#   osmo-lte toggle     L ICONE : le service osmo-lte (systemd) - le demarre s il
#                       est arrete, l arrete s il tourne (pkexec hors root).
#                       [2026-10-12] « tourne » = le service actif OU les demons
#                       (srsenb, srsue, open5gs-mmed) debout : une 4G partie par
#                       « sudo osmo-lte start » ou start-gui, que l unite ne
#                       connait pas, s arrete aussi par l icone - avant, le
#                       clic faisait « systemctl stop » d une unite inactive,
#                       rendait 0 et annoncait « 4G arretee » devant une 4G
#                       bien vivante. L arret eteint TOUT (eNB, UE, coeur).
#   osmo-lte up|down    le service, explicitement (Actions du clic droit)
#   osmo-lte journal    journalctl -u osmo-lte -f
#   osmo-lte install    pose les configs et les lanceurs (tools/osmo-lte-install.sh)
# ============================================================================
set -u

REPO="${OSMO_REPO:-/opt/GSM/osmo-operator}"
SRS_DIR="${OSMO_SRSRAN_DIR:-/root/.config/srsran}"
NETNS="${OSMO_LTE_NETNS:-ue1}"
ZMQ_ENB="${OSMO_ZMQ_PORT:-2000}"
ZMQ_UE="${OSMO_ZMQ_PORT_UE:-2001}"
SRATE="${OSMO_LTE_SRATE:-}"           # vide = ce que disent enb.conf / ue.conf
LOGDIR="${OSMO_LTE_LOGDIR:-/tmp}"
BUILD="${OSMO_SRSRAN_BUILD:-/opt/LTE/srsRAN_4G/build}"
EPC="${OSMO_EPC:-/usr/local/bin/osmo-epc}"
[ -x "$EPC" ] || EPC="$REPO/tools/osmo-epc.sh"
MME_S1="${OSMO_LTE_MME:-127.0.0.2}"

RED='\033[0;31m'; GREEN='\033[0;32m'; YELLOW='\033[1;33m'; CYAN='\033[0;36m'; NC='\033[0m'
_say()  { echo -e "  ${CYAN}→${NC} $*"; }
_ok()   { echo -e "  ${GREEN}✓${NC} $*"; }
_warn() { echo -e "  ${YELLOW}!${NC} $*"; }
_err()  { echo -e "  ${RED}✗${NC} $*" >&2; }

_root() { [ "$(id -u)" -eq 0 ] && return 0; _err "il faut root (netns, tun, SCTP) : sudo $0 $*"; return 1; }
```

```
# [2026-09-08] LE GESTE DE L ICONE. La 4G vit dans services/osmo-lte.service
# (Open5GS + eNB + UE) ; l icone du bureau (data/desktop/osmo-lte.desktop)
# appelle « osmo-lte toggle » : un clic demarre, le suivant arrete. Depuis
# une session sans root, pkexec ouvre la fenetre de mot de passe (repli
# sudo) ; le resultat part en notification, il n y a pas de terminal.
SVC="${OSMO_LTE_SERVICE:-osmo-lte.service}"
ICONE=/usr/share/osmo-operator/icons/osmo-lte.svg
_notif() { command -v notify-send >/dev/null 2>&1 && notify-send -i "$ICONE" "4G du banc" "$1" 2>/dev/null ||
 ↪  true; _say "$1"; }
_lte_debout() {
    case "$(systemctl is-active "$SVC" 2>/dev/null)" in active|activating) return 0 ;; esac
    local d
    for d in srsenb srsue open5gs-mmed; do pgrep -x "$d" >/dev/null 2>&1 && return 0; done
    return 1
}
# Ce que _svc fait EN ROOT, en un seul passage (une seule fenetre de mot de
# passe) : l unite, puis ce qu elle ne portait pas.
_svc_root() {
    local act="$1" rc=0
    case "$act" in
        start)
            # le journal de l unite defile pendant le demarrage : c est lui
            # que la fenetre de lancement lit (reperes [n/4] de lte_start)
            journalctl -u "$SVC" -f -n 0 -o cat 2>/dev/null & local _j=$!
            systemctl start "$SVC"; rc=$?
            sleep 1; kill "$_j" 2>/dev/null; wait "$_j" 2>/dev/null ;;
        stop)
            case "$(systemctl is-active "$SVC" 2>/dev/null)" in
                active|activating) systemctl stop "$SVC"; rc=$? ;;
            esac
            # une 4G lancee hors unite (sudo osmo-lte start, start-gui) : on
            # l eteint de meme - eNB, UE, puis le coeur
            if _lte_debout; then
                lte_stop || rc=1
                "$EPC" stop || rc=1
            fi ;;
    esac
    return "$rc"
}
_svc() {
    local act="$1" rc
    if [ "$act" = toggle ]; then
        if _lte_debout; then act=stop; else act=start; fi
    fi
    [ "$act" = start ] && _notif "demarrage de la 4G (Open5GS, eNB, UE)... une trentaine de secondes"
    if [ "$(id -u)" -eq 0 ]; then _svc_root "$act"; rc=$?
    elif command -v pkexec >/dev/null 2>&1; then pkexec /bin/bash "$0" _svc-root "$act"; rc=$?
    else sudo /bin/bash "$0" _svc-root "$act"; rc=$?; fi
    if [ "$rc" -eq 0 ]; then
        [ "$act" = start ] && _notif "4G en marche : UE attache dans $NETNS (osmo-lte status)" || _notif "4G
         ↪  arretee (eNB, UE et coeur)"
    else
        _notif "echec ($act, code $rc) - journalctl -u $SVC"
    fi
    return "$rc"
}

# [2026-10-12] start / stop / restart SANS root : se relancer eleve, DISPLAY
# conserve (pkexec le nettoie, et l icone/ le terminal de session en ont un).
_eleve() {
    [ "$(id -u)" -eq 0 ] && return 0
    _say "elevation (pkexec) pour « $* »"
    if command -v pkexec >/dev/null 2>&1; then
        exec pkexec /usr/bin/env DISPLAY="${DISPLAY:-}" XAUTHORITY="${XAUTHORITY:-}" /bin/bash "$0" "$@"
    fi
    exec sudo /bin/bash "$0" "$@"
```

```
}

# ── LE MODE « AVEC TRACES » (srsGUI) ──────────────────────────────────────
# [2026-09-10] srsRAN est desormais compile avec ENABLE_GUI=ON (srsGUI, pose
# avant lui par tools/osmo-lte-install.sh) : srsenb et srsue SAVENT tracer leur
# constellation, leur spectre et leur PDSCH en temps reel. Ils ne le FONT que
# si on le demande - « [gui] enable » de enb.conf/ue.conf vaut false, et on ne
# le change pas : la 4G du banc demarre normalement par un service systemd, qui
# n a pas d ecran, et une fenetre Qt qui ne peut pas s ouvrir ferait echouer un
# demarrage qui marchait.
#
# D ou DEUX modes, et le mode normal reste EXACTEMENT ce qu il etait :
#     « start »      ce que fait le service, sans rien a l ecran
#     « start-gui »  le meme lancement + --gui.enable=1 sur les deux
#   (ces deux libelles ne commencent pas par « osmo-lte start » a dessein :
#    l aide du bas de fichier est un `sed` de plage sur ce motif, et une
#    seconde occurrence lui faisait recracher le script entier.)
# La seule difference dans lte_start est un tableau vide quand GUI=0.
GUI=0

# Qt a besoin d un serveur X et d un cookie. Le mode traces est lance depuis le
# bureau (icone, terminal de session) mais TOURNE EN ROOT : DISPLAY et
# XAUTHORITY ne suivent pas toujours (pkexec nettoie l environnement, sudo
# aussi selon env_reset). On les retrouve : le socket X du serveur en marche, et
# le cookie de la session graphique ACTIVE.
_display_ou_rien() {
    if [ -z "${DISPLAY:-}" ]; then
        local s n=""
        for s in /tmp/.X11-unix/X*; do
            [ -S "$s" ] || continue
            n="${s##*/X}"; break
        done
        [ -n "$n" ] || { _err "aucun serveur X (/tmp/.X11-unix vide) : le mode traces demande un ecran - « $0
        ↪  start » sans traces"; return 1; }
        export DISPLAY=":$n"
        _say "DISPLAY absent : on prend $DISPLAY"
    fi
    if [ -z "${XAUTHORITY:-}" ] || [ ! -r "${XAUTHORITY:-}" ]; then
        local uid f
        uid="$(loginctl list-sessions --no-legend 2>/dev/null | awk '$0 ~ / active / {print $2; exit}')"
        [ -n "$uid" ] || uid="$(loginctl list-sessions --no-legend 2>/dev/null | awk 'NR==1{print $2}')"
        for f in "/run/user/$uid/gdm/Xauthority" "/run/user/$uid/.Xauthority" \
                 "$(getent passwd "${uid:-0}" | cut -d: -f6)/.Xauthority"; do
            [ -n "$uid" ] || break
            [ -r "$f" ] && { export XAUTHORITY="$f"; break; }
        done
    fi
    # Qt suivrait WAYLAND_DISPLAY s il traine dans l environnement, et echouerait
    # en root sur le socket wayland de l utilisateur : on impose xcb.
    export QT_QPA_PLATFORM=xcb
    # Un srsenb qui n est pas lie a srsGUI accepte --gui.enable sans rien
    # afficher : on le dit plutot que de laisser chercher.
    local b
    b="$(_bin srsenb 2>/dev/null)"
    if [ -n "$b" ] && ! ldd "$b" 2>/dev/null | grep -q libsrsgui; then
        _warn "srsenb n est pas lie a srsGUI : aucune trace ne s affichera - « osmo-lte-install --build »"
    fi
    return 0
}

# Le mode traces se lance depuis la session, pas depuis systemd : c est donc
# LUI qui va chercher root, en emportant l ecran (meme geste que l icone du
# banc, launch/osmo-launch.sh : pkexec env DISPLAY=... XAUTHORITY=...).
_root_avec_display() {
    local act="${1:-start-gui}"
    [ "$(id -u)" -eq 0 ] && return 0
```

```
    _display_ou_rien || return 1
    _say "elevation (pkexec) en gardant $DISPLAY"
    if command -v pkexec >/dev/null 2>&1; then
        exec pkexec /usr/bin/env DISPLAY="$DISPLAY" XAUTHORITY="${XAUTHORITY:-}" \
             QT_QPA_PLATFORM=xcb "$0" "$act"
    fi
    exec sudo DISPLAY="$DISPLAY" XAUTHORITY="${XAUTHORITY:-}" QT_QPA_PLATFORM=xcb "$0" "$act"
}

# Le binaire installe d abord, la compilation locale sinon.
_bin() {
    local n="$1"
    command -v "$n" 2>/dev/null && return 0
    for f in "$BUILD/srs${n#srs}/src/$n" "$BUILD/$n/src/$n"; do [ -x "$f" ] && { echo "$f"; return 0; }; done
    return 1
}

# Le SIB3 doit etre programme (piege n° 2 de l entete).
_verifier_sib() {
    local sib="$SRS_DIR/sib.conf"
    [ -f "$sib" ] || { _warn "pas de $sib : srsENB prendra ses valeurs par defaut"; return 0; }
    local map
    map="$(grep -oE 'si_mapping_info\s*=\s*\[[^]]*\]' "$sib" | head -1)"
    if [ -n "$map" ] && ! printf '%s' "$map" | grep -qE '\[\s*(3|[0-9]+\s*,\s*)*3(\s*,|\s*\])'; then
        _err "sib.conf : « $map » ne programme pas le SIB3 - srsUE ne s attachera jamais (il faut [ 3, 7 ])"
        return 1
    fi
    _ok "sib.conf : $map"
}

# Les configs : celles de ~/.config/srsran, sinon celles du depot, posees.
_configs() {
    [ -f "$SRS_DIR/enb.conf" ] && [ -f "$SRS_DIR/ue.conf" ] && return 0
    [ -d "$REPO/configs/srsran" ] || { _err "pas de configs srsRAN ($SRS_DIR, ni $REPO/configs/srsran)"; return 1;
     ↪  }
    mkdir -p "$SRS_DIR"
    cp -n "$REPO/configs/srsran/"* "$SRS_DIR/"
    _ok "configs srsRAN posees depuis le depot dans $SRS_DIR"
}

lte_tourne() { pgrep -x "$1" >/dev/null 2>&1; }

lte_start() {
    _root start || return 1
    local enb ue
    enb="$(_bin srsenb)" || { _err "srsenb introuvable (dpkg -i osmo-build-srsran, ou osmo-lte-install --build)";
     ↪  return 1; }
    ue="$(_bin srsue)"   || { _err "srsue introuvable"; return 1; }
    for p in srsenb srsue; do
        lte_tourne "$p" && { _err "$p tourne deja (pid $(pgrep -x "$p" | head -1)) - « $0 restart » pour repartir
         ↪  propre"; return 1; }
    done
    _configs || return 1
    _verifier_sib || return 1
    # [2026-10-12] le bureau suit : l antenne se dresse dans le fond d ecran
    # (tools/osmo-fond.py se relance sous l utilisateur du bureau ; le retour
    # a la ville, a l arret, est le fait de osmo-panel.py quand plus rien ne tourne)
    [ -x "$REPO/tools/osmo-fond.py" ] && ("$REPO/tools/osmo-fond.py" antenne >/dev/null 2>&1 &)
    ip netns list 2>/dev/null | grep -qw "$NETNS" || { ip netns add "$NETNS" && _ok "espace reseau $NETNS cree"; }

    # [2026-10-12] Les reperes « [n/4] » : la fenetre de lancement du bureau
    # (tools/osmo-gtk-run.py --progress) en fait sa barre de progression.
    echo "[1/4] coeur Open5GS"
    # Le coeur : Open5GS, par osmo-epc (idempotent : ce qui tourne deja reste).
    "$EPC" start | sed 's/^/  /'
    for _ in $(seq 1 20); do ss -Sln 2>/dev/null | grep -q "$MME_S1:36412 " && break; sleep 0.25; done
```

```
    ss -Sln 2>/dev/null | grep -q "$MME_S1:36412 " && _ok "MME : S1 en ecoute sur $MME_S1" \
        || _warn "MME : le S1 n ecoute pas sur $MME_S1 (voir « osmo-epc status », osmo-epc log mme)"

    export HOME=/root                       # srs* lisent ~/.config/srsran : celui de root
    cd "$SRS_DIR" || return 1               # sib.conf & co. relatifs, et le .ctxt de srsue
    rm -f .ctxt                             # un vieux contexte fait echouer l attach apres un redemarrage du coeur

    # Les traces : rien du tout quand GUI=0 - le mode normal est intact.
    local -a gui_opt=()
    [ "$GUI" = 1 ] && gui_opt=(--gui.enable=1)
    local -a enb_rf=() ue_rf=()
    if [ -n "$SRATE" ]; then
        enb_rf=(--rf.device_name=zmq --rf.device_args="fail_on_disconnect=true,tx_port=tcp://*:$ZMQ_ENB
        ↪  ,rx_port=tcp://localhost:$ZMQ_UE,id=enb,base_srate=$SRATE")
        ue_rf=(--rf.device_name=zmq
        ↪  --rf.device_args="tx_port=tcp://*:$ZMQ_UE,rx_port=tcp://localhost:$ZMQ_ENB,id=ue,base_srate=$SRATE")
    fi
    echo "[2/4] station de base srsENB"
    # L eNB, et l UE tout de suite derriere : voir le piege n° 1.
    setsid "$enb" "$SRS_DIR/enb.conf" "${enb_rf[@]}" "${gui_opt[@]}" \
        --log.filename="$LOGDIR/osmo-lte-enb.log" >"$LOGDIR/osmo-lte-enb.console" 2>&1 </dev/null &
    sleep 0.5
    echo "[3/4] mobile srsUE"
    setsid "$ue" "$SRS_DIR/ue.conf" "${ue_rf[@]}" "${gui_opt[@]}" \
        --gw.netns="$NETNS" --log.filename="$LOGDIR/osmo-lte-ue.log" >"$LOGDIR/osmo-lte-ue.console" 2>&1
        ↪  </dev/null &
    _ok "srsENB puis srsUE lances (ZeroMQ $ZMQ_ENB/$ZMQ_UE, UE dans $NETNS${SRATE:+, base_srate
    ↪  $SRATE})${gui_opt:+ ${GREEN}avec les traces srsGUI sur $DISPLAY${NC}}"

    echo "[4/4] attach du mobile"
    _say "attente de l attach (30 s max)..."
    local i ip
    for i in $(seq 1 60); do
        ip="$(ip -n "$NETNS" -4 -o addr show 2>/dev/null | awk '/tun_srsue/ {print $4}')"
        [ -n "$ip" ] && break
        sleep 0.5
    done
    if [ -n "$ip" ]; then
        _ok "UE attache : $ip (tun_srsue dans $NETNS) en $((i / 2)) s"
    else
        _err "pas d attach en 30 s - voir $LOGDIR/osmo-lte-ue.console et « $0 status »"
        grep -E "Found|Attach|fail|Reject" "$LOGDIR/osmo-lte-ue.console" | tail -4 | sed 's/^/      /'
        return 1
    fi
}

lte_stop() {
    _root stop || return 1
    local p n=0
    for p in srsue srsenb; do
        if lte_tourne "$p"; then pkill -x "$p"; n=$((n + 1)); fi
    done
    sleep 2
    for p in srsue srsenb; do lte_tourne "$p" && pkill -9 -x "$p"; done
    [ "$n" -gt 0 ] && _ok "$n processus srs* arrete(s)" || _say "rien ne tournait"
    # Le S1 met un instant a se refermer cote SCTP ; l eNB suivant en a besoin.
    sleep 1
}

lte_status() {
    echo -e "\033[1msrsRAN sur ZeroMQ + Open5GS - la 4G du banc\033[0m"
    local p
    for p in open5gs-mmed srsenb srsue; do
        lte_tourne "$p" && _ok "$p : pid $(pgrep -x "$p" | head -1)" || _warn "$p : arrete"
    done
```

```
    ss -San 2>/dev/null | grep -q "ESTAB.*:36412" && _ok "S1 : eNB associe au MME" || _warn "S1 : pas d
     ↪  association eNB-MME"
    # Les echantillons eNB -> UE partent de la douille locale :$ZMQ_ENB de
    # l eNB (bytes_sent) ; ce qu elle RECOIT n est que les requetes de l UE.
    local a b
    a="$(ss -tin state established "( sport = :$ZMQ_ENB )" 2>/dev/null | grep -oE "bytes_sent:[0-9]+" | head -1 |
     ↪  cut -d: -f2)"
    if [ -n "$a" ]; then
        sleep 1
        b="$(ss -tin state established "( sport = :$ZMQ_ENB )" 2>/dev/null | grep -oE "bytes_sent:[0-9]+" | head
         ↪  -1 | cut -d: -f2)"
        local ko=$(( (${b:-0} - a) / 1024 ))
        if [ "$ko" -gt 100 ]; then _ok "ZeroMQ : $ko Ko/s d echantillons eNB -> UE"
        else _warn "ZeroMQ : $ko Ko/s - la radio de l eNB est morte (fail_on_disconnect) : « $0 restart »"; fi
    else
        _warn "ZeroMQ : aucun lien eNB <-> UE"
    fi
    local ip
    ip="$(ip -n "$NETNS" -4 -o addr show 2>/dev/null | awk '/tun_srsue/ {print $4}')"
    [ -n "$ip" ] && _ok "UE : attache, $ip dans $NETNS" || _warn "UE : pas attache (pas de tun_srsue dans $NETNS)"
}

case "${1:-status}" in
    start|stop|restart) _eleve "$@" ;;      # hors root : on repasse ici en root
esac
case "${1:-status}" in
    start)   lte_start ;;
    start-gui|gui|start-display)
        # Le mode traces : root AVEC l ecran, puis le lancement normal + GUI=1.
        _root_avec_display || exit 1
        _display_ou_rien || exit 1
        GUI=1
        lte_start ;;
    traces|gui-seul)
        # [2026-10-12] Les traces sans relancer toute la 4G : la radio seule
        # repart avec srsGUI ; lte_start laisse le coeur tel quel (osmo-epc
        # start est idempotent). Lance hors unite : « osmo-lte down » / l icone
        # savent arreter une radio partie ainsi (_svc_root).
        _root_avec_display traces || exit 1
        _display_ou_rien || exit 1
        GUI=1
        if lte_tourne srsenb || lte_tourne srsue; then
            _say "radio relancee avec les traces srsGUI - le coeur Open5GS reste debout"
            lte_stop
        fi
        lte_start ;;
    stop)    lte_stop ;;
    restart) lte_stop; lte_start ;;
    status)  lte_status ;;
    log)     exec tail -F "$LOGDIR"/osmo-lte-{enb,ue}.console ;;
    toggle|bascule) _svc toggle ;;
    _svc-root) _root _svc-root || exit 1; _svc_root "$2" ;;
    up)      _svc start ;;
    down)    _svc stop ;;
    journal) exec journalctl -u "$SVC" -f --no-pager ;;
    install) shift; exec bash "$REPO/tools/osmo-lte-install.sh" "$@" ;;
    *)       sed -n '/^#   osmo-lte start/,/^#   osmo-lte install/p' "$0" | sed 's/^# \{0,2\}//'; exit 2 ;;
esac
```

3.420 /opt/GSM/osmo-operator/tools/osmo-moon.py

7331 octets, 164 lignes → 164 lignes

```
#!/usr/bin/env python3
# osmo-moon.py - le cavalier/lune ANIME, pose sous le Conky (colonne haut-droite).
#
# Meme mecanisme de fenetre que tools/osmo-panel.py : une fenetre GTK de type
# BUREAU (Gdk.WindowTypeHint.DESKTOP) - sous toutes les fenetres, au-dessus du
# fond, non decoree, collante, transparente (argb). Elle N'A RIEN A VOIR avec la
# conf Conky (osmo-conky.conf) : c'est un element SEPARE, aligne A LA VERTICALE
# du Conky (meme colonne haut-droite, meme largeur 400), colle EN DESSOUS de lui.
#
# Le gif (820x560, 240 frames) est redimensionne PROPORTIONNELLEMENT a la largeur
# demandee (defaut 400, comme le Conky) : hauteur = largeur * 560/820. L'animation
# est jouee frame par frame (GdkPixbuf.PixbufAnimation + iterateur), chaque frame
# etant mise a l'echelle avant affichage.
#
# Reglage par variables d'environnement :
#   OSMO_MOON_GIF     chemin du gif      (defaut: configs/conky/cavalier_lune.gif)
#   OSMO_MOON_W       largeur en px      (defaut: 400, la largeur du Conky)
#   OSMO_MOON_GAP_X   marge a droite     (defaut: 24, le gap_x du Conky)
#   OSMO_MOON_GAP_Y   y du haut du gif   (defaut: 600 = sous le Conky ; a ajuster
#                                         selon la hauteur reelle du Conky)
#
# Lance par /usr/local/bin/osmo-desktop-panel (comme osmo-panel.py + conky).
import os
import sys

# [2026-09-06] LA FENETRE DE BUREAU EXIGE X11, PAS WAYLAND.
# Sous Wayland, GTK3 ignore window.move() ET Gdk.WindowTypeHint.DESKTOP : le
# compositeur place la fenetre ou il veut (au milieu, au premier plan) - d ou
# l encart et le dino qui « demarrent n importe ou ». Le protocole ne donne au
# client aucun moyen de se poser a des coordonnees absolues (il faudrait
# gtk-layer-shell, absent sur mutter). On passe donc par XWayland, ou le
# positionnement X11 est honore, exactement comme le Conky voisin (own_window
# type desktop) qui, lui, se cale bien. On ne force que si un DISPLAY existe :
# sans serveur X, mieux vaut le comportement par defaut qu un echec au demarrage.
X11_FORCE = os.environ.get("XDG_SESSION_TYPE") == "wayland" and bool(os.environ.get("DISPLAY"))
if X11_FORCE:
    os.environ["GDK_BACKEND"] = "x11"

import gi

gi.require_version("Gtk", "3.0")
gi.require_version("Gdk", "3.0")
gi.require_version("GdkPixbuf", "2.0")
from gi.repository import Gdk, GdkPixbuf, GLib, Gtk  # noqa: E402

# [2026-09-06] ET ALORS LE TYPE BUREAU NE VA PLUS. Sur XWayland, une fenetre
# _NET_WM_WINDOW_TYPE_DESKTOP est rangee par GNOME Shell dans la couche du
# bureau, SOUS la fenetre plein ecran des icones (extension DING) - qui, elle,
# avale tous les clics : l encart s affichait mais ses boutons ne repondaient
# plus. On garde donc une fenetre NORMALE, simplement maintenue en dessous des
# autres (keep_below + stick + hors barre des taches) : meme rendu, et les
# clics arrivent. Sur X11 natif (la cle live) le type BUREAU marchait : on n y
# touche pas.
TYPE_HINT = Gdk.WindowTypeHint.NORMAL if X11_FORCE else Gdk.WindowTypeHint.DESKTOP
# [2026-09-06] ... ET « EN DESSOUS » NON PLUS. Deuxieme moitie du meme piege :
# avec _NET_WM_STATE_BELOW, GNOME Shell range la fenetre XWayland sous sa
# propre couche de bureau (native Wayland), qui est plein ecran et prend tous
# les clics - la fenetre restait visible mais totalement inerte. Sous XWayland
# on renonce donc a « toujours dessous » : fenetre ordinaire, collante et hors
# barre des taches. Elle passera au premier plan quand on clique dedans, ce qui
# est le prix a payer pour qu on puisse justement cliquer dedans. Mettre
# OSMO_DESKTOP_BELOW=1 pour retrouver l ancien comportement (fenetre sous tout,
# clics perdus) ; sur X11 natif rien ne change.
KEEP_BELOW = (not X11_FORCE) or os.environ.get("OSMO_DESKTOP_BELOW") == "1"

import cairo  # noqa: E402
```

```
REPO = os.environ.get("OSMO_REPO", "/opt/GSM/osmo-operator")
GIF = os.environ.get("OSMO_MOON_GIF", os.path.join(REPO, "configs/conky/cavalier_lune.gif"))
WIN_W = int(os.environ.get("OSMO_MOON_W", "400"))     # meme largeur que le Conky
GAP_X = int(os.environ.get("OSMO_MOON_GAP_X", "24"))  # meme gap_x que le Conky
GAP_Y = int(os.environ.get("OSMO_MOON_GAP_Y", "600")) # sous le Conky (a ajuster)


def target_size(anim):
    """Largeur imposee, hauteur PROPORTIONNELLE au ratio natif du gif."""
    gw, gh = anim.get_width(), anim.get_height()
    w = WIN_W
    h = max(1, round(w * gh / gw))
    return w, h


class Moon(Gtk.Window):
    def __init__(self):
        super().__init__(title="osmo-moon")
        try:
            self.anim = GdkPixbuf.PixbufAnimation.new_from_file(GIF)
        except GLib.Error as e:
            print(f"[moon] gif introuvable/illisible: {GIF} ({e})", file=sys.stderr)
            sys.exit(1)
        self.w, self.h = target_size(self.anim)

        # colonne haut-droite du moniteur primaire, comme le Conky (top_right)
        disp = Gdk.Display.get_default()
        mon = disp.get_primary_monitor() or disp.get_monitor(0)
        g = mon.get_geometry()
        self.x = g.x + g.width - GAP_X - self.w
        self.y = g.y + GAP_Y
        print(f"[moon] {self.w}x{self.h} @ {self.x},{self.y} (gif
         ↪  {self.anim.get_width()}x{self.anim.get_height()})",
              flush=True)

        # fenetre de type BUREAU (cf. osmo-panel.py)
        self.set_type_hint(TYPE_HINT)
        self.set_decorated(False)
        self.set_resizable(False)
        self.set_keep_below(KEEP_BELOW)
        self.set_skip_taskbar_hint(True)
        self.set_skip_pager_hint(True)
        self.stick()
        self.set_default_size(self.w, self.h)
        self.set_size_request(self.w, self.h)
        self.move(self.x, self.y)

        screen = self.get_screen()
        visual = screen.get_rgba_visual()
        if visual and screen.is_composited():
            self.set_visual(visual)
        self.set_app_paintable(True)
        self.connect("draw", self.on_draw)
        self.connect("destroy", Gtk.main_quit)

        self.image = Gtk.Image()
        self.add(self.image)

        # lecture animee : iterateur de frames, mise a l'echelle proportionnelle
        self.it = self.anim.get_iter(None)
        self.show_all()
        self._pos = (self.x, self.y)
        self._pin()
        self.connect("map-event", self._pin)
        GLib.timeout_add(500, self._pin)
```

```
        self._render()

    # [2026-09-06] SE REPOSER APRES COUP. Meme sous X11, le gestionnaire de
    # fenetres peut deplacer la fenetre au moment ou il la mappe (mutter le
    # fait pour les fenetres non decorees qu il ne reconnait pas comme du
    # bureau). Un seul move() avant show_all() ne tient donc pas : on recale
    # au map-event, puis une derniere fois une demi-seconde plus tard.
    def _pin(self, *_a):
        self.move(*self._pos)
        return False
    def on_draw(self, _w, cr):
        cr.set_operator(cairo.OPERATOR_SOURCE)
        cr.set_source_rgba(0, 0, 0, 0)
        cr.paint()
        return False

    def _render(self):
        self.it.advance(None)
        frame = self.it.get_pixbuf()
        scaled = frame.scale_simple(self.w, self.h, GdkPixbuf.InterpType.BILINEAR)
        self.image.set_from_pixbuf(scaled)
        delay = self.it.get_delay_time()   # ms ; -1 = image fixe
        GLib.timeout_add(delay if delay > 0 else 100, self._render)
        return False


def main():
    Moon()
    Gtk.main()


if __name__ == "__main__":
    main()
```

3.421 /opt/GSM/osmo-operator/tools/osmo-ofono-bridge.py

11097 octets, 277 lignes → 277 lignes

```
#!/usr/bin/env python3
# osmo-ofono-bridge.py - LES APPELS ET LES SMS DU BANC, EN LIGNE DE COMMANDE.
#
# [2026-09-07] CE QUE CE PROGRAMME N EST PLUS. C etait le « raccord mobile »
# vers Waydroid : il portait les SMS et les appels du banc jusqu a un Android
# en conteneur (notification, insertion dans content://sms/inbox), faute d une
# vraie pile radio de ce cote-la. Waydroid est abandonne - le telephone du banc
# est desormais la VM postmarketOS/Phosh, qui a ModemManager et parle AT au
# modem du banc (tools/osmo-phonesim-banc.py --connect). Tout le code Android a
# donc ete retire ; ce qui reste, et qui servait deja, c est le pilotage d oFono
# depuis un terminal ou une icone :
#
#   osmo-sms-send <numero> <texte>   -> org.ofono.MessageManager.SendMessage
#   osmo-call <numero>|answer|hangup -> org.ofono.VoiceCallManager
#
# Lance sans --once, il TIENT la ligne : il journalise les SMS et les appels que
# le banc lui pousse, publie l etat dans /run/osmo-ril/state.json (l encart et
# le Conky le lisent) et execute les commandes deposees dans la file par les
# deux lanceurs ci-dessus.
#
# LA DATA. Elle n est pas encore en service sur le banc (pas de contexte PDP
# etabli). Le chemin est neanmoins CABLE ici : --data active le contexte oFono
# et publie l interface obtenue. Sans --data on ne touche a rien.
#
# Reglage par variables d environnement :
#   OSMO_OFONO_MODEM    chemin du modem oFono   (defaut: le premier trouve)
```

```
#   OSMO_BRIDGE_SPOOL   file des commandes      (defaut: /run/osmo-ril/spool)
#   OSMO_BRIDGE_STATE   etat publie (json)      (defaut: /run/osmo-ril/state.json)
import argparse
import json
import os
import sys
import time

from gi.repository import Gio, GLib

MODEM = os.environ.get("OSMO_OFONO_MODEM", "")
RUN = os.environ.get("OSMO_RIL_DIR", "/run/osmo-ril")
SPOOL = os.environ.get("OSMO_BRIDGE_SPOOL", os.path.join(RUN, "spool"))
STATE = os.environ.get("OSMO_BRIDGE_STATE", os.path.join(RUN, "state.json"))

OFONO = "org.ofono"
IF_MANAGER = "org.ofono.Manager"
IF_MODEM = "org.ofono.Modem"
IF_SMS = "org.ofono.MessageManager"
IF_VOICE = "org.ofono.VoiceCallManager"
IF_CALL = "org.ofono.VoiceCall"
IF_CONN = "org.ofono.ConnectionManager"
IF_CTX = "org.ofono.ConnectionContext"


def log(*a):
    print("[ril]", *a, flush=True)


# ── LE BUS ──────────────────────────────────────────────────────────
class Bus:
    """Le strict necessaire de D-Bus systeme, sans dbus-python (absent de
    l image) : Gio suffit et arrive avec PyGObject, deja la pour l encart."""

    def __init__(self):
        self.c = Gio.bus_get_sync(Gio.BusType.SYSTEM, None)

    def call(self, path, iface, method, params=None, timeout=20000):
        return self.c.call_sync(OFONO, path, iface, method, params, None,
                                Gio.DBusCallFlags.NONE, timeout, None)

    def subscribe(self, iface, signal, cb, path=None):
        return self.c.signal_subscribe(OFONO, iface, signal, path, None,
                                       Gio.DBusSignalFlags.NONE, cb)


# ── L ETAT PUBLIE ───────────────────────────────────────────────────
def publish(**kw):
    """Ce que le reste du banc lit : l encart et le Conky."""
    try:
        os.makedirs(RUN, exist_ok=True)
        cur = {}
        if os.path.exists(STATE):
            with open(STATE) as f:
                cur = json.load(f)
        cur.update(kw)
        cur["ts"] = int(time.time())
        tmp = STATE + ".tmp"
        with open(tmp, "w") as f:
            json.dump(cur, f)
        os.replace(tmp, STATE)
    except Exception as e:                      # jamais fatal
        log("etat non publie :", e)


# ── LE PONT ─────────────────────────────────────────────────────────
```

```
class Bridge:
    def __init__(self, bus, modem, with_data=False):
        self.bus = bus
        self.modem = modem
        self.with_data = with_data
        self.calls = {}

    # -- modem
    def power_on(self):
        for prop in ("Powered", "Online"):
            try:
                self.bus.call(self.modem, IF_MODEM, "SetProperty",
                              GLib.Variant("(sv)", (prop, GLib.Variant("b", True))))
            except GLib.Error as e:
                log("modem %s : %s" % (prop, e.message.split(":")[-1].strip()))

    # -- SMS
    def on_sms(self, _c, _s, _p, _i, _sig, params):
        text = params[0]
        info = params[1]
        sender = info.get("Sender", "?")
        when = info.get("LocalSentTime") or info.get("SentTime") or ""
        log("SMS de %s : %s" % (sender, text))
        publish(last_sms={"from": sender, "text": text, "when": when})

    # -- appels
    def on_call_added(self, _c, _s, _p, _i, _sig, params):
        path, props = params[0], params[1]
        num = props.get("LineIdentification", "?")
        state = props.get("State", "?")
        self.calls[path] = num
        log("appel %s (%s) %s" % (num, state, path))
        publish(call={"number": num, "state": state})

    def on_call_removed(self, _c, _s, _p, _i, _sig, params):
        path = params[0]
        num = self.calls.pop(path, "?")
        log("appel termine %s" % num)
        publish(call={"number": num, "state": "ended"})

    # -- data (cablee, pas allumee : voir l entete)
    def data_up(self):
        try:
            ctxs = self.bus.call(self.modem, IF_CONN, "GetContexts").unpack()[0]
        except GLib.Error as e:
            log("pas de ConnectionManager :", e.message.split(":")[-1].strip())
            return
        if not ctxs:
            log("aucun contexte PDP declare sur le modem")
            return
        path, props = ctxs[0]
        log("contexte %s (apn=%s)" % (path, props.get("AccessPointName", "?")))
        try:
            self.bus.call(path, IF_CTX, "SetProperty",
                          GLib.Variant("(sv)", ("Active", GLib.Variant("b", True))))
            props = self.bus.call(path, IF_CTX, "GetProperties").unpack()[0]
        except GLib.Error as e:
            log("contexte non active :", e.message.split(":")[-1].strip())
            publish(data={"up": False, "error": e.message})
            return
        st = props.get("Settings", {})
        log("data : if=%s addr=%s gw=%s" % (st.get("Interface"), st.get("Address"),
                                            st.get("Gateway")))
        publish(data={"up": True, "interface": st.get("Interface"),
                      "address": st.get("Address"), "gateway": st.get("Gateway"),
                      "dns": st.get("DomainNameServers", [])})
```

```
# -- les commandes venues de osmo-sms-send / osmo-call
def command(self, line):
    try:
        verb, rest = (line.split(" ", 1) + [""])[:2]
    except ValueError:
        return
    verb = verb.strip().lower()
    if verb == "sms":
        num, _, text = rest.partition(" ")
        if not num or not text:
            log("usage: sms <numero> <texte>")
            return
        try:
            self.bus.call(self.modem, IF_SMS, "SendMessage",
                          GLib.Variant("(ss)", (num, text)))
            log("SMS envoye a %s" % num)
        except GLib.Error as e:
            log("envoi refuse :", e.message.split(":")[-1].strip())
    elif verb == "call":
        try:
            self.bus.call(self.modem, IF_VOICE, "Dial",
                          GLib.Variant("(ss)", (rest.strip(), "default")))
        except GLib.Error as e:
            log("appel refuse :", e.message.split(":")[-1].strip())
    elif verb == "answer":
        for path in list(self.calls):
            try:
                self.bus.call(path, IF_CALL, "Answer")
            except GLib.Error as e:
                log("decrochage refuse :", e.message.split(":")[-1].strip())
    elif verb == "hangup":
        try:
            self.bus.call(self.modem, IF_VOICE, "HangupAll")
        except GLib.Error as e:
            log("raccrochage refuse :", e.message.split(":")[-1].strip())
    elif verb == "data-up":
        self.data_up()
    else:
        log("commande inconnue : %s" % line.strip())

def watch_spool(self):
    """La file : une ligne = une commande. Un fifo serait plus joli mais un
    simple fichier se lit et s ecrit depuis n importe quel script, y compris
    a la main pendant une seance."""
    os.makedirs(RUN, exist_ok=True)
    if not os.path.exists(SPOOL):
        open(SPOOL, "w").close()
        os.chmod(SPOOL, 0o666)

    def tick():
        try:
            with open(SPOOL, "r+") as f:
                lines = f.read().splitlines()
                if lines:
                    f.seek(0)
                    f.truncate()
            for line in lines:
                if line.strip():
                    self.command(line)
        except Exception as e:
            log("file :", e)
        return True

    GLib.timeout_add(500, tick)
```

```
def first_modem(bus):
    modems = bus.call("/", IF_MANAGER, "GetModems").unpack()[0]
    if not modems:
        return ""
    return modems[0][0]


def main():
    ap = argparse.ArgumentParser(description="appels et SMS du banc par oFono")
    ap.add_argument("--data", action="store_true",
                    help="active aussi le contexte PDP (pas encore en service sur le banc)")
    ap.add_argument("--once", metavar="CMD",
                    help="passe une commande au pont deja lance (sms/call/answer/hangup)")
    args = ap.parse_args()

    if args.once:
        os.makedirs(RUN, exist_ok=True)
        with open(SPOOL, "a") as f:
            f.write(args.once.rstrip("\n") + "\n")
        return 0

    try:
        bus = Bus()
    except GLib.Error as e:
        print("[ril] bus systeme injoignable : %s" % e, file=sys.stderr)
        return 1

    modem = MODEM or first_modem(bus)
    if not modem:
        print("[ril] aucun modem oFono (ofonod demarre ? modem branche ?)", file=sys.stderr)
        publish(modem=None)
        return 1
    log("modem", modem)
    publish(modem=modem)

    br = Bridge(bus, modem, with_data=args.data)
    br.power_on()
    bus.subscribe(IF_SMS, "IncomingMessage", br.on_sms, path=modem)
    bus.subscribe(IF_SMS, "ImmediateMessage", br.on_sms, path=modem)
    bus.subscribe(IF_VOICE, "CallAdded", br.on_call_added, path=modem)
    bus.subscribe(IF_VOICE, "CallRemoved", br.on_call_removed, path=modem)
    br.watch_spool()
    if args.data:
        br.data_up()

    log("en ligne (SMS + appels ; data %s)" % ("demandee" if args.data else "non cablee"))
    GLib.MainLoop().run()
    return 0


if __name__ == "__main__":
    sys.exit(main())
```

3.422 /opt/GSM/osmo-operator/tools/osmo-op.sh

12159 octets, 241 lignes → 241 lignes

```
#!/bin/bash
# ============================================================================
#  osmo-op - QUEL OPERATEUR L ECRAN REGARDE
# ============================================================================
#  Le bureau du banc a UN seul encart (tools/osmo-panel.py) et UN seul Conky
#  (tools/conky-osmo-status.sh), mais un banc multi en porte trois plus le hub.
```

```
#  Le choix vit dans un unique fichier, /run/osmo-fft/operator, que les deux
#  lisent a chaque rafraichissement :
#
#      OP=2  MODE=docker  IP=172.20.0.12  NAME=osmo-operator-2
#      DASH=http://172.20.0.12:8080
#
#  Ce script est ce qui l ECRIT, et la seule definition de "la liste des
#  operateurs" - l encart et le Conky s y referent, ils ne la recalculent pas.
#
#      osmo-op                l operateur courant
#      osmo-op --list         les operateurs et leur etat
#      osmo-op --list-vue     les arrets VISIBLES (ce qui tourne) et leur etat
#      osmo-op --next / --prev l operateur suivant / precedent  (les fleches)
#      osmo-op --set N        l operateur N
#
#  [2026-09-04] LES OPERATEURS ARRETES COMPTENT AUSSI. L encart ne montrait ses
#  fleches que si au moins deux operateurs etaient EN MARCHE (docker ps) : sur
#  un banc ou op2 et op3 n avaient pas encore demarre, il n y avait aucune
#  fleche - donc aucun moyen d aller voir POURQUOI ils n avaient pas demarre.
#  L ecran doit pouvoir regarder un operateur eteint : c est meme la le seul
#  moment ou on en a besoin. L etat (actif / arrete) est affiche, pas filtre.
#
#  EN REVANCHE, PAS D OPERATEURS INVENTES. Sans /etc/osmocom/osmo-multi.conf -
#  donc sur un banc a un seul operateur - la liste rend le seul natif, et
#  l encart cache ses fleches : il n y a nulle part ou aller. Les fleches
#  n apparaissent que quand la topologie declare vraiment plusieurs operateurs.
# ============================================================================
set -u
RUN="${OSMO_FFT_DIR:-/run/osmo-fft}"
OP_FILE="$RUN/operator"
MULTI_CONF="${MULTI_CONF:-/etc/osmocom/osmo-multi.conf}"
DASH_PORT="${DASH_PORT:-8080}"
HUB_NAME="${MULTI_HUB_NAME:-osmo-inter-stp}"
M3UA_PORT="${MULTI_M3UA_PORT:-2908}"

# -- LA LISTE : "idx mode ip" par ligne ---------------------------------------
# La topologie d abord (MULTI_OPS de osmo-multi.conf, ecrit par addition.sh) ;
# a defaut, op 1 natif + les suivants en docker, sur le plan d adressage du
# hub (172.20.0.1<idx>, cf. start-multi.sh).
liste() {
    local specs="" idx mode ip
    if [ -r "$MULTI_CONF" ]; then
        specs="$(sed -nE 's/^MULTI_OPS="?([^"]*)"?.*/\1/p' "$MULTI_CONF" | head -1)"
    fi
    if [ -n "$specs" ]; then
        for s in $specs; do
            IFS=: read -r idx mode ip _ <<< "$s"
            [ -n "$idx" ] || continue
            echo "$idx ${mode:-docker} ${ip:-}"
        done
        # -- ET LE HUB, EN DERNIERE POSITION -------------------------------
        # [2026-09-04] L inter-STP ne figurait nulle part dans le cycle des
        # fleches : on pouvait regarder les trois operateurs, jamais le noeud
        # qui les relie - alors que c est LUI qu on interroge quand le SS7 va
        # mal. Il devient donc le quatriemme arret, apres op3, avant de revenir
        # a op1. Son "index" est son nom (hub) et non un chiffre : il ne
        # numerote pas un operateur, et l encart le rend par une vue SS7 (la
        # matrice, les ASP, son journal) au lieu du spectre I/Q - un hub M3UA
        # n a pas de radio, une FFT n y voudrait rien dire.
        _hub_ip="$(sed -nE 's/^MULTI_HUB_IP=([^ #]*).*/\1/p' "$MULTI_CONF" | head -1)"
        echo "hub interstp ${_hub_ip:-172.20.0.10}"
    else
        # Pas de topologie : un seul operateur, le natif. Rien a parcourir.
        echo "1 native "
    fi
    # -- ET LA 4G, SI ELLE EST INSTALLEE SUR CETTE MACHINE ------------------
```

```
    # [2026-09-10] Le banc porte une 4G complete (tools/osmo-lte.sh : Open5GS,
    # srsENB et srsUE sur ZeroMQ) et l ecran n avait aucun moyen de la
    # regarder : le cycle des fleches s arretait au hub. Elle devient le
    # dernier arret - apres les operateurs 2G et le hub - et le Conky lui rend
    # une vue a elle (les demons du coeur EPC, l eNB, l UE, le S1, la cellule).
    # Son « index » est « lte », comme celui du hub est « hub » : il ne
    # numerote pas un operateur 2G. Pas de conteneur, pas de FFT : tout est
    # natif sur l hote.
    lte_installee && echo "lte lte "
}

# La 4G est-elle posee ici ? Les binaires, la ou tools/osmo-lte-install.sh les
# met (ou la compilation locale de srsRAN).
lte_installee() {
    [ -x /usr/local/bin/srsenb ] && return 0
    [ -x "${OPEN5GS_PREFIX:-/opt/LTE/open5gs/install}/bin/open5gs-mmed" ] && return 0
    [ -x "${OSMO_SRSRAN_BUILD:-/opt/LTE/srsRAN_4G/build}/srsenb/src/srsenb" ]
}

# actif IDX MODE [IP] : l operateur tourne-t-il ?
#
# [2026-09-04] « ARRETE » VOULAIT PARFOIS DIRE « JE N AI PAS LE DROIT DE VOIR ».
# Le Conky et l encart tournent sous le compte de la SESSION, pas sous root, et
# ce compte n est pas forcement dans le groupe `docker` :
#     $ docker ps
#     permission denied while trying to connect to the docker API at
#     unix:///var/run/docker.sock
# `docker ps 2>/dev/null | grep -qx` rendait alors faux, et l ecran annoncait
# op2 et op3 ARRETES pendant que `docker ps` en root les montrait « Up ». Une
# sonde qui confond « absent » et « invisible » est une sonde qui ment - le
# depot appelle ca ailleurs une sonde mensongere, et c en est une.
#
# On distingue donc les deux cas : si docker ne repond pas A NOUS, on retombe
# sur une sonde qui ne demande aucun privilege - le tableau de bord de
# l operateur, sur son IP du reseau du hub. Le compte est ajoute au groupe
# `docker` par ailleurs (installer/calamares/modules/users.conf, addition.sh) ;
# ce repli couvre la session deja ouverte, ou le groupe n a pas encore pris.
actif() {
    if [ "$2" = lte ]; then
        # La 4G tourne des que le coeur ou la radio tourne (osmo-lte status).
        pgrep -x open5gs-mmed >/dev/null 2>&1 || pgrep -x srsenb >/dev/null 2>&1 \
            || pgrep -x srsue >/dev/null 2>&1
    elif [ "$2" = interstp ]; then
        if docker ps --format '{{.Names}}' 2>/dev/null | grep -q .; then
            docker ps --format '{{.Names}}' 2>/dev/null | grep -qx "${HUB_NAME}"
        else
            # Sans acces docker : le port M3UA du hub, en SCTP - une sonde TCP
            # y serait muette, d ou `ss` plutot qu un /dev/tcp.
            ss -an 2>/dev/null | awk -v p=":${M3UA_PORT}" \
               '$1=="sctp" && index($5,p) {f=1} END{exit !f}'
        fi
    elif [ "$2" = native ]; then
        pgrep -x osmo-bsc >/dev/null 2>&1 || pgrep -x osmo-stp >/dev/null 2>&1
    elif docker ps --format '{{.Names}}' 2>/dev/null | grep -q .; then
        docker ps --format '{{.Names}}' 2>/dev/null | grep -qx "osmo-operator-$1"
    elif [ -n "${3:-}" ]; then
        timeout 1 bash -c "echo >/dev/tcp/$3/${DASH_PORT}" 2>/dev/null
    else
        return 1
    fi
}

courant() { sed -n 's/^OP=//p' "$OP_FILE" 2>/dev/null | head -1; }

ecrire() {
    local idx="$1" mode="$2" ip="$3"
```

```
    # ── LE REPERTOIRE APPARTIENT AU BANC, PAS A CELUI QUI ARRIVE EN PREMIER ──
    # [2026-09-10] /run/osmo-fft est normalement cree par osmo-fft-snap.service
    # (RuntimeDirectory, 0775, groupe sudo) : l encart et les sondes tournent
    # sous le compte de la SESSION et doivent pouvoir y ecrire. Mais ce
    # `mkdir -p` pouvait le creer AVANT lui, sous root, en 0755 root:root - et
    # a partir de la, plus une seule sonde de la session ne pouvait y ecrire.
    # C est ce qui a coute la banniere des timeslots (osmo-ts-probe.py). On
    # pose donc les memes droits que l unite ; en session (non root) le chgrp
    # echoue sans consequence, le repertoire est deja au bon compte.
    if [ ! -d "$RUN" ]; then
        mkdir -p "$RUN" 2>/dev/null || true
        chmod 0775 "$RUN" 2>/dev/null || true
        chgrp sudo "$RUN" 2>/dev/null || true
    fi
    local nom="osmo-operator-$idx"
    [ "$mode" = interstp ] && nom="$HUB_NAME"
    [ "$mode" = lte ] && nom="osmo-lte"
    printf 'OP=%s\nMODE=%s\nIP=%s\nNAME=%s\nDASH=http://%s:%s\n' \
        "$idx" "$mode" "$ip" "$nom" "${ip:-127.0.0.1}" "$DASH_PORT" > "$OP_FILE.tmp" \
        && mv -f "$OP_FILE.tmp" "$OP_FILE"
}

# ── LES ARRETS VISIBLES : CE QUI TOURNE VRAIMENT SUR CETTE MACHINE ──────────
# [2026-09-10] Sur un banc ou seul op1 est monte, l ecran gardait op2, op3 et
# le hub dans le cycle : trois arrets ou il n y a RIEN a regarder, et trois
# quarts de tour de fleche pour revenir chez soi. On ne montre donc que ce qui
# tourne - un seul `docker ps` le dit pour les trois.
#
# Ce que ca coute, et qui etait le motif du 2026-09-04 (« LES OPERATEURS
# ARRETES COMPTENT AUSSI ») : on ne peut plus arriver AUX FLECHES sur un
# operateur eteint pour voir pourquoi il ne demarre pas. `osmo-op --set 2` y
# va toujours, et `osmo-op --list` continue de TOUS les afficher avec leur
# etat - c est la qu on lit « op2 arrete ».
#
# Deux exceptions, qui ne sont pas des operateurs docker : le premier arret
# (l operateur natif, CETTE machine - son banc peut etre a l arret sans
# disparaitre) et la 4G (installee ici, arretee ou non : c est justement la
# vue qui dit comment la demarrer).
liste_vue() {
    local premier=1 noms="" docker_ok=0
    if noms="$(docker ps --format '{{.Names}}' 2>/dev/null)"; then docker_ok=1; fi
    while read -r idx mode ip; do
        if [ "$premier" = 1 ] || [ "$mode" = lte ]; then
            echo "$idx $mode $ip"
        elif [ "$docker_ok" = 0 ]; then
            # docker ne nous repond pas (pas dans le groupe) : on ne SAIT pas,
            # et une sonde qui confond « absent » et « invisible » ment. On
            # garde l arret plutot que de le faire disparaitre a tort.
            echo "$idx $mode $ip"
        elif [ "$mode" = interstp ]; then
            printf '%s\n' "$noms" | grep -qx "$HUB_NAME" && echo "$idx $mode $ip"
        else
            printf '%s\n' "$noms" | grep -qx "osmo-operator-$idx" && echo "$idx $mode $ip"
        fi
        premier=0
    done < <(liste)
}

# decale +1 / -1 dans la liste, en boucle.
decale() {
    local pas="$1" cur n i=0 sel=0
    cur="$(courant)"; cur="${cur:-1}"
    mapfile -t L < <(liste_vue)
    n="${#L[@]}"; [ "$n" -gt 0 ] || return 1
    for i in "${!L[@]}"; do
        [ "${L[$i]%% *}" = "$cur" ] && sel="$i"
```

```
    done
    sel=$(( (sel + pas + n) % n ))
    # shellcheck disable=SC2086
    set -- ${L[$sel]}
    ecrire "$1" "$2" "${3:-}"
    echo "op $1 (${2})"
}

case "${1:-}" in
    --list|-l)
        while read -r idx mode ip; do
            if actif "$idx" "$mode" "$ip"; then e="actif"; else e="arrete"; fi
            [ "$idx" = "$(courant)" ] && e="$e  <- ecran"
            printf '  op %-2s %-7s %-13s %s\n' "$idx" "$mode" "${ip:-(hote)}" "$e"
        done < <(liste) ;;
    --list-vue)
        # « idx mode ip etat » - LA liste sur laquelle le Conky construit son
        # selecteur, pour qu ecran et fleches s arretent aux memes endroits.
        while read -r idx mode ip; do
            if actif "$idx" "$mode" "$ip"; then e=actif; else e=arrete; fi
            # L IP peut etre vide (l hote) : un tiret la remplace, sinon la
            # ligne n a plus que trois champs et le lecteur prend l etat pour
            # une adresse.
            echo "$idx $mode ${ip:--} $e"
        done < <(liste_vue) ;;
    --next|-n) decale 1 ;;
    --prev|-p) decale -1 ;;
    --set|-s)
        [ $# -ge 2 ] || { echo "usage: $0 --set N" >&2; exit 2; }
        while read -r idx mode ip; do
            if [ "$idx" = "$2" ]; then ecrire "$idx" "$mode" "$ip"; echo "op $idx ($mode)"; exit 0; fi
        done < <(liste)
        echo "op $2 : absent de la topologie ($MULTI_CONF)" >&2; exit 1 ;;
    ""|--current|-c)
        [ -r "$OP_FILE" ] && cat "$OP_FILE" || echo "OP=1"$'\n'"MODE=native" ;;
    *) sed -n '2,30p' "$0" | sed 's/^# \{0,1\}//'; exit 2 ;;
esac
```

3.423 /opt/GSM/osmo-operator/tools/osmo-panel.py

46799 octets, 943 lignes → 943 lignes

```
#!/usr/bin/env python3
# osmo-panel.py - l ENCART VIVANT du bureau, version cliquable.
#
# Une fenetre GTK de type BUREAU (le meme mecanisme que les icones DING
# d Ubuntu : sous toutes les fenetres, au-dessus du fond, et elle recoit les
# clics - ce que Conky ne sait pas faire), posee EXACTEMENT sur le cadre Calvin
# & Hobbes du fond d ecran (tools/wallpaper-render.py : boite 510,220-1410,1010
# en 1920x1080), a l echelle de l ecran. Elle affiche /run/osmo-fft/panel.png
# (tools/osmo-fft-snap.py : le strip du jour qui fond vers « FFT du mobile +
# mobile.log » quand le banc est pret) et, en pied, un petit menu :
#
#   [ Dashboard ]  firefox http://<operateur>:8080
#   [ tmux ]       un terminal attache a la session du banc
#   [ VTY 4247 ]   un terminal sur telnet 127.0.0.1 4247 (console du mobile)
#   [ ZMQ ]        l etat de la radio virtuelle srsRAN (ZeroMQ) : vert quand
#                  les points d entree ecoutent, gris sinon. Un clic ouvre un
#                  terminal sur qui ecoute et qui est connecte.
#   IP 192.168.x.y l adresse de CETTE machine sur le reseau - celle qu il faut
#                  ecrire dans un srsRAN qui tourne ailleurs, ou donner a un
#                  autre noeud du banc. Un clic la copie dans le presse-papier.
#   [ < ] op 2/3 [ > ]   SEULEMENT en multi-operateur : passe d un operateur a
#                  l autre. Le choix est ecrit dans /run/osmo-fft/operator ; le
```

```
#                  rendu FFT (osmo-fft-snap.py) et le Conky en haut a droite
#                  (conky-osmo-status.sh) le lisent et suivent.
#
# Multi-operateur = /etc/osmocom/osmo-multi.conf (addition.sh) ET au moins un
# conteneur osmo-operator-N en marche (docker ps). Sans ca : pas de fleches.
#
# Lance par l autostart osmo-conky.desktop (avec le Conky osmo-conky.conf).
#
# [2026-10-12] L ENCART N EST LA QUE SI UN BANC TOURNE. Banc a l arret, la
# fenetre est CACHEE (pas seulement transparente : la barre de boutons
# restait posee au milieu d un bureau vide). Elle apparait des qu une 2G
# (natif osmo-bsc, un conteneur osmo-operator-N, le hub) ou une 4G
# (open5gs-mmed, srsenb, srsue) est debout, et repart quand tout s eteint.
# Le processus, lui, reste : osmo-desktop-panel le compte pour vivant et ne
# le double pas. OSMO_PANEL_TOUJOURS=1 la garde a l ecran quoi qu il arrive.
import os
import re
import shlex
import signal
import socket
import shutil
import subprocess
import sys
import threading
import time

import cairo
# [2026-09-06] LA FENETRE DE BUREAU EXIGE X11, PAS WAYLAND.
# Sous Wayland, GTK3 ignore window.move() ET Gdk.WindowTypeHint.DESKTOP : le
# compositeur place la fenetre ou il veut (au milieu, au premier plan) - d ou
# l encart et le dino qui « demarrent n importe ou ». Le protocole ne donne au
# client aucun moyen de se poser a des coordonnees absolues (il faudrait
# gtk-layer-shell, absent sur mutter). On passe donc par XWayland, ou le
# positionnement X11 est honore, exactement comme le Conky voisin (own_window
# type desktop) qui, lui, se cale bien. On ne force que si un DISPLAY existe :
# sans serveur X, mieux vaut le comportement par defaut qu un echec au demarrage.
X11_FORCE = os.environ.get("XDG_SESSION_TYPE") == "wayland" and bool(os.environ.get("DISPLAY"))
if X11_FORCE:
    os.environ["GDK_BACKEND"] = "x11"

import gi

gi.require_version("Gtk", "3.0")
gi.require_version("Gdk", "3.0")
gi.require_version("GdkPixbuf", "2.0")
from gi.repository import Gdk, GdkPixbuf, Gio, GLib, Gtk  # noqa: E402

# [2026-09-06] ET ALORS LE TYPE BUREAU NE VA PLUS. Sur XWayland, une fenetre
# _NET_WM_WINDOW_TYPE_DESKTOP est rangee par GNOME Shell dans la couche du
# bureau, SOUS la fenetre plein ecran des icones (extension DING) - qui, elle,
# avale tous les clics : l encart s affichait mais ses boutons ne repondaient
# plus. On garde donc une fenetre NORMALE, simplement maintenue en dessous des
# autres (keep_below + stick + hors barre des taches) : meme rendu, et les
# clics arrivent. Sur X11 natif (la cle live) le type BUREAU marchait : on n y
# touche pas.
TYPE_HINT = Gdk.WindowTypeHint.NORMAL if X11_FORCE else Gdk.WindowTypeHint.DESKTOP
# [2026-09-06] ... ET « EN DESSOUS » NON PLUS. Deuxieme moitie du meme piege :
# avec _NET_WM_STATE_BELOW, GNOME Shell range la fenetre XWayland sous sa
# propre couche de bureau (native Wayland), qui est plein ecran et prend tous
# les clics - la fenetre restait visible mais totalement inerte. Sous XWayland
# on renonce donc a « toujours dessous » : fenetre ordinaire, collante et hors
# barre des taches. Elle passera au premier plan quand on clique dedans, ce qui
# est le prix a payer pour qu on puisse justement cliquer dedans. Mettre
# OSMO_DESKTOP_BELOW=1 pour retrouver l ancien comportement (fenetre sous tout,
# clics perdus) ; sur X11 natif rien ne change.
KEEP_BELOW = (not X11_FORCE) or os.environ.get("OSMO_DESKTOP_BELOW") == "1"
```

```
RUN = os.environ.get("OSMO_FFT_DIR", "/run/osmo-fft")
IMG = os.path.join(RUN, "panel.png")
OP_FILE = os.path.join(RUN, "operator")
# [2026-09-04] LE VERROU D INSTANCE EST PAR SESSION, PAS PAR MACHINE. Il etait
# dans /run/osmo-fft, que systemd cree root:root (RuntimeDirectory de
# osmo-fft-snap.service). Sur la cle le bureau tourne en root et l ecriture
# passait ; sur le systeme installe le bureau est celui du compte cree par
# l installeur, et open(PID_FILE, "w") levait PermissionError - apport ouvrait
# "osmo-operator a rencontre une erreur interne" a CHAQUE ouverture de session.
# XDG_RUNTIME_DIR (/run/user/<uid>) appartient a l utilisateur, et un encart
# par session est exactement ce qu on veut compter.
_RUNDIR = os.environ.get("XDG_RUNTIME_DIR") or "/run/user/%d" % os.getuid()
PID_FILE = os.path.join(_RUNDIR if os.path.isdir(_RUNDIR) else RUN, "osmo-panel.pid")
ETAT_FILE = os.path.join(_RUNDIR if os.path.isdir(_RUNDIR) else RUN, "osmo-banc.etat")
REPO_DIR = os.environ.get("OSMO_REPO", "/opt/GSM/osmo-operator")
MULTI_CONF = os.environ.get("MULTI_CONF", "/etc/osmocom/osmo-multi.conf")
TMUX_SESSION = os.environ.get("TMUX_SESSION", "calypso")
DASH_PORT = int(os.environ.get("DASH_PORT", "8080"))
# Le VTY du client mobile (osmocom-bb « mobile »). 4247 pour MS#1 ; MS#2 est sur
# 4248 - voir start-native.sh, qui les attribue.
# [2026-10-12] 4247, ou 4347 en --dsp : start-native.sh publie le port reel dans
# /run/osmo-banc/ms1-vty. MS_VTY_PORT force ; sinon le fichier ; sinon 4247.
def vty_port():
    if os.environ.get("MS_VTY_PORT"):
        return int(os.environ["MS_VTY_PORT"])
    try:
        with open("/run/osmo-banc/ms1-vty") as f:
            return int(f.read().strip())
    except (OSError, ValueError):
        return 4247


VTY_PORT = vty_port()
# La boite de l encart dans le fond (1920x1080) : x, y, largeur, hauteur.
# [2026-09-08] 790 de haut : deux moities (4G en haut, 2G en bas), voir
# tools/osmo-fft-snap.py. Les quatre fichiers qui la connaissent doivent
# rester d accord (liste dans tools/wallpaper-render.py).
BOX = (510, 220, 900, 790)
FW, FH = 1920, 1080

# [2026-09-04] LA BARRE N A PLUS DE FOND. Elle posait un rectangle sombre
# derriere les boutons, EN PLUS du pied deja peint par osmo-fft-snap.py : deux
# fonds l un sur l autre, aux bords qui ne tombaient pas au meme endroit, et une
# arete visible en travers du pied de l encart. Les boutons ont deja leur propre
# fond (`.osmo-bar button`), ils se detachent tres bien seuls.
CSS = b"""
/* [2026-09-04] `background-color: transparent` NE SUFFIT PAS, et le rectangle
   reapparaissait. Adwaita ne peint pas seulement une couleur sur une boite : il
   y pose un `background-image` (un degrade), plus une bordure et une ombre.
   Neutraliser la seule couleur laisse l image - d ou le bandeau plus clair qui
   revenait derriere les boutons, par-dessus le pied deja peint par
   osmo-fft-snap.py. On coupe les quatre.
   `background: none` DOIT venir avant `background-color`, sinon la propriete
   raccourcie remet la couleur par defaut apres coup. */
.osmo-bar { background: none; background-image: none; background-color: transparent;
            border: none; box-shadow: none; padding: 0; margin: 0; }
.osmo-bar button { background: #161b22; color: #e6edf3; border: 1px solid #30363d;
                   border-radius: 6px; padding: 1px 8px; font: 9pt "DejaVu Sans Mono"; min-height: 0; }
.osmo-bar button:hover { background: #21262d; border-color: #58a6ff; }
.osmo-bar button.arrow { font: 15pt "DejaVu Sans Mono"; font-weight: bold;
                         padding: 0 14px; min-width: 34px; color: #3fb950; }
.osmo-bar button.arrow:hover { border-color: #ff9a3c; color: #ff9a3c; }
.osmo-bar label { color: #8b949e; font: 9pt "DejaVu Sans Mono"; }
.osmo-bar label.op { color: #3fb950; font-weight: bold; font-size: 11pt; }
/* La radio virtuelle : vert quand elle ecoute, gris quand elle est a terre.
```

```
   La couleur EST l information - c est ce qu on regarde d un coup d oeil. */
.osmo-bar button.zmq-on  { color: #3fb950; border-color: #238636; }
.osmo-bar button.zmq-off { color: #6e7681; }
.osmo-bar label.ip { color: #58a6ff; font-weight: bold; }
"""


# ── LA GEOMETRIE : OU EST LE CADRE SUR CET ECRAN ──────────────────────
def geometry():
    """(x, y, w, h) du cadre du strip sur l ecran, selon le mode d affichage
    du fond (zoom = couvre et rogne, scaled = contient, stretched)."""
    disp = Gdk.Display.get_default()
    mon = disp.get_primary_monitor() or disp.get_monitor(0)
    g = mon.get_geometry()
    sw, sh = g.width, g.height
    try:
        mode = Gio.Settings.new("org.gnome.desktop.background").get_string("picture-options")
    except Exception:
        mode = "zoom"
    if mode == "stretched":
        sx, sy, ox, oy = sw / FW, sh / FH, 0, 0
    else:
        s = min(sw / FW, sh / FH) if mode == "scaled" else max(sw / FW, sh / FH)
        sx = sy = s
        ox, oy = (FW * s - sw) / 2, (FH * s - sh) / 2
    bx, by, bw, bh = BOX
    return (g.x + int(bx * sx - ox), g.y + int(by * sy - oy), int(bw * sx), int(bh * sy)), (sw, sh, mode)


# ── LES OPERATEURS ────────────────────────────────────────────────────
# [2026-09-04] LES OPERATEURS ARRETES COMPTENT AUSSI.
# Cette fonction ne rendait que les conteneurs EN MARCHE (docker ps), et
# update_op() ne montrait les fleches qu a partir de deux entrees. Consequence :
# sur un banc ou op2 et op3 n avaient pas demarre, l encart n avait AUCUNE
# fleche - donc aucun moyen de regarder l operateur qu on voulait justement
# diagnostiquer. C est l inverse du besoin : l ecran doit pouvoir se poser sur
# un operateur eteint, c est meme la le seul moment ou on en a besoin.
# On rend donc TOUTE la topologie, avec l etat (up) a cote, et le pied de
# l encart ecrit "op 3/3 arrete" au lieu de faire disparaitre le bouton.
# EN REVANCHE, PAS DE FLECHES HORS MULTI-OPERATEUR. Sans
# /etc/osmocom/osmo-multi.conf - donc sur un banc a un seul operateur - il n y a
# rien vers quoi naviguer : la liste rend le seul natif, et update_op() cache
# les fleches. On n invente pas d operateurs 2 et 3 qui n existent pas.
# Meme liste et meme regle que tools/osmo-op.sh, qui est la version en ligne
# de commande (et la definition de reference).


M3UA_PORT = int(os.environ.get("MULTI_M3UA_PORT", "2908"))

# ── LA RADIO VIRTUELLE srsRAN (ZeroMQ) ────────────────────────────────
# srsRAN sans materiel : le pilote « zmq » remplace la carte radio par deux
# sockets ZeroMQ - l un porte les echantillons de la station vers le mobile,
# l autre du mobile vers la station (tcp://IP:2000 et :2001 par defaut). Le
# banc s en sert pour donner une VRAIE couche radio au telephone : srsUE
# s attache a la station, et le modem du telephone (postmarketOS, via
# tools/osmo-pmos.sh) parle a ce mobile-la.
# On ne devine RIEN de l etat : on regarde qui ecoute sur ces ports.
ZMQ_PORTS = [int(x) for x in os.environ.get("OSMO_ZMQ_PORTS", "2000,2001").split(",") if x.strip()]


def _sctp_ecoute(port):
    """Un socket SCTP ecoute-t-il sur ce port ? Lecture directe de
    /proc/net/sctp/eps - aucun privilege, et `ss` peut ne pas etre installe."""
    try:
        with open("/proc/net/sctp/eps") as f:
            for line in f.readlines()[1:]:
```

```
                ch = line.split()
                # colonne LPORT : la 4e dans le format du noyau.
                if len(ch) > 4 and ch[4] == str(port):
                    return True
    except OSError:
        pass
    try:
        out = subprocess.run(["ss", "-an"], capture_output=True, text=True, timeout=3).stdout
        return any(l.startswith("sctp") and f":{port}" in l for l in out.splitlines())
    except Exception:
        return False


def _tcp_ecoute(port):
    """Un socket TCP ecoute-t-il sur ce port ? On lit /proc/net/tcp{,6}
    directement : aucun privilege, et `ss` peut manquer. La 4e colonne vaut
    « 0A » (TCP_LISTEN) et l adresse locale est « HEX:PORT »."""
    hexport = "%04X" % port
    for f in ("/proc/net/tcp", "/proc/net/tcp6"):
        try:
            with open(f) as fh:
                for line in fh.readlines()[1:]:
                    ch = line.split()
                    if len(ch) > 3 and ch[3] == "0A" and ch[1].endswith(":" + hexport):
                        return True
        except OSError:
            continue
    return False


def zmq_etat():
    """(actif, detail) de la radio virtuelle srsRAN.

    Actif = au moins un des ports ZeroMQ ecoute. Le detail nomme ceux qui
    ecoutent : sur un banc a moitie monte (la station lancee, le mobile pas
    encore), c est la seule facon de voir lequel des deux manque.
    """
    ouverts = [p for p in ZMQ_PORTS if _tcp_ecoute(p)]
    if not ouverts:
        return False, "aucun point d entree ZeroMQ (srsRAN arrete)"
    manque = [p for p in ZMQ_PORTS if p not in ouverts]
    detail = "ecoute : " + ", ".join(str(p) for p in ouverts)
    if manque:
        detail += " ; MANQUE : " + ", ".join(str(p) for p in manque)
    return True, detail


def ip_locale():
    """L adresse de cette machine sur le reseau - celle qu on donne aux autres.

    « ip route get » designe l interface qui porte la route par defaut sans
    envoyer le moindre paquet : c est la bonne reponse meme sans reseau au
    bout, et elle ne depend pas d un nom d interface ecrit en dur.
    """
    try:
        out = subprocess.run(["ip", "-4", "route", "get", "1.1.1.1"],
                             capture_output=True, text=True, timeout=2).stdout
        m = re.search(r"\bsrc\s+(\d+\.\d+\.\d+\.\d+)", out)
        if m:
            return m.group(1)
    except Exception:
        pass
    # Pas de route par defaut (banc isole) : la premiere adresse non locale.
    try:
        out = subprocess.run(["ip", "-4", "-br", "addr"], capture_output=True,
                             text=True, timeout=2).stdout
```

```
        for ligne in out.splitlines():
            m = re.search(r"(\d+\.\d+\.\d+\.\d+)/", ligne)
            if m and not m.group(1).startswith("127."):
                return m.group(1)
    except Exception:
        pass
    return "127.0.0.1"


def ip_toutes():
    """Toutes les adresses, pour l infobulle : sur un banc il y en a plusieurs
    (le reseau, les ponts docker, apn0...) et celle qui compte depend de qui
    demande."""
    try:
        out = subprocess.run(["ip", "-4", "-br", "addr"], capture_output=True,
                             text=True, timeout=2).stdout.strip()
        return out or "aucune adresse"
    except Exception:
        return "adresses illisibles"


def operators():
    """Liste [(idx, mode, ip, up)] : toute la topologie, marche ou pas."""
    ops = []
    try:
        with open(MULTI_CONF) as f:
            txt = f.read()
    except OSError:
        txt = ""
    m = re.search(r'^MULTI_OPS="?([^"\n]*)"?', txt, re.M)
    specs = m.group(1).split() if m else []
    if not specs:
        # Pas de topologie multi-operateur : un seul operateur, le natif.
        # update_op() cachera les fleches - il n y a nulle part ou aller.
        specs = ["1:native:"]
    # [2026-09-04] `docker ps` ECHOUE SOUS LE COMPTE DE LA SESSION s il n est pas
    # dans le groupe `docker` ("permission denied ... /var/run/docker.sock"), et
    # l encart annoncait alors « op 3/3 arrete » avec les conteneurs bien en
    # marche. On ne confond plus « absent » et « invisible » : quand docker ne
    # nous repond pas, on sonde le tableau de bord de l operateur, qui ne
    # demande aucun privilege. Meme regle que tools/osmo-op.sh.
    running = set()
    docker_ok = False
    try:
        r = subprocess.run(["docker", "ps", "--format", "{{.Names}}"], capture_output=True,
                           text=True, timeout=5)
        docker_ok = r.returncode == 0
        running = set(r.stdout.split())
    except Exception:
        docker_ok = False

    def reachable(ip):
        if not ip:
            return False
        try:
            with socket.create_connection((ip, DASH_PORT), timeout=0.4):
                return True
        except OSError:
            return False

    for spec in specs:
        parts = spec.split(":")
        if len(parts) < 3:
            continue
        idx, mode, ip = parts[0], parts[1], parts[2]
        if mode == "native":
```

```
            up = subprocess.run(["pgrep", "-x", "osmo-bsc"], capture_output=True).returncode == 0
        elif docker_ok:
            up = f"osmo-operator-{idx}" in running
        else:
            up = reachable(ip)
        ops.append((idx, mode, ip, up))
    if not any(o[1] == "native" for o in ops):
        ops.insert(0, ("1", "native", "", False))
    # ── LE HUB EST LE DERNIER ARRET DU CYCLE ──────────────────────────────
    # [2026-09-04] On pouvait regarder les trois operateurs, jamais le noeud qui
    # les relie - alors que c est lui qu on interroge quand le SS7 va mal.
    # L encart le rend par une vue SS7 (matrice de connectivite + journal du
    # hub) et non par un spectre : un hub M3UA n a pas de radio, une FFT n y
    # voudrait rien dire. Voir tools/osmo-fft-snap.py, render_interstp().
    # Meme regle que tools/osmo-op.sh, la definition de reference.
    if specs and any(o[1] == "docker" for o in ops):
        hub_ip = ""
        m = re.search(r'^MULTI_HUB_IP="?([^"\s#]*)', txt, re.M)
        if m:
            hub_ip = m.group(1)
        hub = os.environ.get("MULTI_HUB_NAME", "osmo-inter-stp")
        # [2026-09-04] LE HUB N A PAS DE TABLEAU DE BORD. Le repli `reachable()`
        # tapait sur 172.20.0.10:8080, que l inter-STP n ecoute pas - il ne
        # route que du M3UA : l encart affichait « inter-STP arrete » sur un hub
        # parfaitement vivant. Sans acces docker, on regarde donc ce qu il
        # ecoute VRAIMENT, un socket SCTP sur le port M3UA. Une sonde TCP y est
        # muette (c est du SCTP), d ou la lecture de /proc/net/sctp.
        up = hub in running if docker_ok else _sctp_ecoute(M3UA_PORT)
        ops.append(("hub", "interstp", hub_ip or "172.20.0.10", up))
    return ops


# ── Y A-T-IL UN BANC A MONTRER ? ──────────────────────────────────────────
# 2G : un operateur de la topologie en marche (natif, conteneur ou hub - la
# liste de operators(), qui sait deja le dire). 4G : l un des demons que le
# Conky regarde pour dire « 4G debout » (tools/conky-osmo-status.sh, lte) -
# memes noms, meme regle. `pgrep -x` : le NOM exact, pas la ligne de commande
# (un `tail -F srsue.log` ou cet encart lui-meme ne comptent pas).
LTE_DEMONS = os.environ.get("OSMO_LTE_DEMONS", "open5gs-mmed|srsenb|srsue")
PANEL_TOUJOURS = os.environ.get("OSMO_PANEL_TOUJOURS") == "1"


def lte_actif():
    try:
        out = subprocess.run(["pgrep", "-x", LTE_DEMONS], capture_output=True,
                             text=True, timeout=5).stdout.split()
    except Exception:
        return False
    # pgrep liste aussi les ZOMBIES (un srsue plante, pas encore recolte par
    # son parent) : un mort ne tient pas de banc, on ne le compte pas.
    for pid in out:
        try:
            with open(f"/proc/{pid}/stat") as f:
                if f.read().rsplit(")", 1)[1].split()[0] != "Z":
                    return True
        except (OSError, IndexError):
            return True       # illisible : on le croit vivant plutot que de cacher un banc
    return False


def banc_actif(ops):
    """Vrai si une 2G (un operateur ou le hub en marche) ou une 4G tourne."""
    return PANEL_TOUJOURS or any(o[3] for o in ops) or lte_actif()


# ── LE FICHIER EST LA SOURCE PARTAGEE, PAS UNE SORTIE ─────────────────────
```

```
# [2026-09-04] Cet encart ECRIVAIT /run/osmo-fft/operator et ne le LISAIT
# jamais. Trois consequences, toutes vues sur le banc :
#   - Ctrl+Alt+O (raccourci) et `osmo-op --next` changeaient bien le fichier -
#     le Conky suivait - mais l encart, lui, restait sur son operateur : ses
#     boutons Dashboard / tmux / VTY continuaient de viser l ancien ;
#   - pire, des qu un conteneur changeait d etat, refresh_ops() rappelait
#     update_op(), qui REECRIVAIT le fichier avec la selection de l encart :
#     le choix fait au clavier etait annule quelques secondes plus tard, sans
#     que rien ne l explique. « ctrl alt O ne marche pas » - il marchait, il
#     etait ecrase.
#   - et deux encarts (deux sessions) se seraient disputes le fichier.
# L encart LIT donc maintenant, et n ECRIT que sur un clic de ses fleches.
def read_operator():
    """L operateur choisi, quel que soit celui qui l a choisi (fleches de
    l encart, raccourci clavier, `osmo-op`). None si le fichier n existe pas."""
    try:
        with open(OP_FILE) as f:
            for line in f:
                if line.startswith("OP="):
                    return line[3:].strip() or None
    except OSError:
        pass
    return None


def write_operator(op):
    idx, mode, ip = op[0], op[1], op[2]
    # NAME est le nom du CONTENEUR : c est par lui que osmo-fft-snap.py et le
    # Conky entrent (docker exec). Pour le hub ce n est pas
    # « osmo-operator-hub » - ce conteneur n existe pas - mais osmo-inter-stp.
    # Meme regle que tools/osmo-op.sh.
    nom = (os.environ.get("MULTI_HUB_NAME", "osmo-inter-stp") if mode == "interstp"
           else f"osmo-operator-{idx}")
    os.makedirs(RUN, exist_ok=True)
    tmp = OP_FILE + ".tmp"
    with open(tmp, "w") as f:
        f.write(f"OP={idx}\nMODE={mode}\nIP={ip}\nNAME={nom}\n"
                f"DASH=http://{ip or '127.0.0.1'}:{DASH_PORT}\n")
    os.replace(tmp, OP_FILE)


def spawn(argv):
    subprocess.Popen(argv, stdin=subprocess.DEVNULL, stdout=subprocess.DEVNULL,
                     stderr=subprocess.DEVNULL, start_new_session=True)


def terminal(cmd, titre="osmo-panel"):
    """Ouvre `cmd` (chaine shell) dans un terminal VTE pose DANS une fenetre GTK
    du banc (tools/osmo-gtk-run.py --tty). [2026-10-01] Plus d emulateur de
    terminal : aucune entree de l encart n en ouvre."""
    root = "" if os.geteuid() == 0 else "sudo -E "
    gtk_run = os.path.join(os.path.dirname(os.path.abspath(__file__)), "osmo-gtk-run.py")
    spawn([sys.executable, gtk_run, "--tty", "--title", titre, "--", "bash", "-c", root + cmd])


class Panel(Gtk.Window):
    def __init__(self):
        super().__init__(title="osmo-panel (osmo-egprs)")
        (x, y, w, h), (sw, sh, mode) = geometry()
        print(f"[panel] ecran {sw}x{sh} ({mode}) : encart a {x},{y} {w}x{h}", flush=True)
        self.geo = (x, y, w, h)
        self.set_type_hint(TYPE_HINT)
        self.set_decorated(False)
        self.set_resizable(False)
        self.set_keep_below(KEEP_BELOW)
        self.set_skip_taskbar_hint(True)
```

```
self.set_skip_pager_hint(True)
self.stick()
self.set_default_size(w, h)
self.set_size_request(w, h)
self.move(x, y)
screen = self.get_screen()
visual = screen.get_rgba_visual()
if visual and screen.is_composited():
    self.set_visual(visual)
self.set_app_paintable(True)
self.connect("draw", self.on_draw)
self.connect("destroy", Gtk.main_quit)

prov = Gtk.CssProvider()
prov.load_from_data(CSS)
Gtk.StyleContext.add_provider_for_screen(screen, prov, Gtk.STYLE_PROVIDER_PRIORITY_APPLICATION)

self.image = Gtk.Image()
overlay = Gtk.Overlay()
overlay.add(self.image)

bar = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=6)
bar.get_style_context().add_class("osmo-bar")
bar.set_halign(Gtk.Align.END)
bar.set_valign(Gtk.Align.END)
scale = w / BOX[2]
bar.set_margin_end(int(18 * scale))
bar.set_margin_bottom(int(14 * scale))
b_dash = self.b_dash = Gtk.Button(label="Dashboard")
b_dash.set_tooltip_text("firefox sur le tableau de bord de l operateur choisi")
b_dash.connect("clicked", self.on_dash)
b_tmux = self.b_tmux = Gtk.Button(label="tmux")
b_tmux.set_tooltip_text("terminal attache a la session du banc (Ctrl-b d pour detacher)")
b_tmux.connect("clicked", self.on_tmux)
# Le VTY du mobile : la console ou l on tape « show ms », « call 100102 »
# ... C est le troisieme endroit qu on ouvre a la main dix fois par
# seance ; il n avait pas de bouton.
b_vty = self.b_vty = Gtk.Button(label=f"VTY {VTY_PORT}")
b_vty.set_tooltip_text(f"terminal : telnet 127.0.0.1 {VTY_PORT} (console du mobile)")
b_vty.connect("clicked", self.on_vty)
# La radio virtuelle srsRAN, et l adresse de la machine. Les deux
# repondent a la meme question - « ou est-ce que je branche le
# telephone ? » - et n avaient aucune place sur le bureau : il fallait
# un terminal et deux commandes pour savoir si la radio ecoutait, et
# une troisieme pour l adresse a mettre en face.
b_zmq = self.b_zmq = Gtk.Button(label="ZMQ")
b_zmq.connect("clicked", self.on_zmq)
self.l_ip = Gtk.Label(label="IP —")
self.l_ip.get_style_context().add_class("ip")
# Un label ne recoit pas les clics : on l habille d une boite d evenements.
ip_box = Gtk.EventBox()
ip_box.add(self.l_ip)
ip_box.connect("button-press-event", self.on_ip)
ip_box.set_tooltip_text("l adresse de cette machine - un clic la copie")
self.ip_box = ip_box

self.b_prev = Gtk.Button(label="◀")
self.b_next = Gtk.Button(label="▶")
self.b_prev.get_style_context().add_class("arrow")
self.b_next.get_style_context().add_class("arrow")
self.b_prev.set_tooltip_text("operateur precedent (op1, op2 ...)")
self.b_next.set_tooltip_text("operateur suivant (op1, op2 ...)")
self.b_prev.connect("clicked", self.on_prev)
self.b_next.connect("clicked", self.on_next)
self.l_op = Gtk.Label(label="")
self.l_op.get_style_context().add_class("op")
```

```
    for wdg in (b_dash, b_tmux, b_vty, b_zmq, ip_box,
                self.b_prev, self.l_op, self.b_next):
        bar.pack_start(wdg, False, False, 0)
    overlay.add_overlay(bar)
    self.add(overlay)

    self.ops = []
    self.cur = 0
    self.mtime = None
    self._pos = (x, y)
    self.connect("map-event", self._pin)
    # Les enfants sont montres, la FENETRE ne l est que si un banc tourne
    # (refresh_ops, ci-dessous) : un show_all() ici l aurait fait passer
    # a l ecran une seconde avant de la cacher.
    overlay.show_all()
    self.update_op()          # show_all vient de tout montrer : on recache les fleches hors multi
    # None : etat inconnu - le premier refresh_ops ecrit l etat du banc
    # (fond d ecran, fichier d etat) meme s il est « a l arret ».
    self._visible = None
    self.refresh_ops()
    self.refresh_image()
    GLib.timeout_add(1000, self.refresh_image)
    GLib.timeout_add(2000, self.refresh_ops)
    self.refresh_zmq()
    GLib.timeout_add(3000, self.refresh_zmq)

# ── montrer / cacher selon le banc ──────────────────────────────
def montrer(self, actif):
    if actif == self._visible:
        return
    self._visible = actif
    self._etat_banc(actif)
    if actif:
        print("[panel] un banc tourne : encart a l ecran", flush=True)
        self.show()
        self._pin()
        GLib.timeout_add(500, self._pin)
    else:
        print("[panel] plus de banc : encart cache", flush=True)
        self.hide()

# [2026-10-12] LE BUREAU SUIT LE BANC, PAS SEULEMENT L ENCART. Au meme
# moment : (1) le fond d ecran passe en fondu de la ville seule a la ville
# avec l antenne (tools/osmo-fond.py), et retour a l arret ; (2) l etat est
# ecrit dans $XDG_RUNTIME_DIR/osmo-banc.etat ("1"/"0"), que osmo-topzone.py
# lit pour fondre le titre « Telecom Operator » vers la banniere LAB GSM.
# osmo-launcher.py lance aussi le fond des le CLIC sur un run (plus tot
# que la detection d ici) ; osmo-fond est idempotent, les deux cohabitent.
def _etat_banc(self, actif):
    try:
        with open(ETAT_FILE + ".tmp", "w") as f:
            f.write("1\n" if actif else "0\n")
        os.replace(ETAT_FILE + ".tmp", ETAT_FILE)
    except OSError as e:
        print("[panel] etat du banc non ecrit : %s" % e, flush=True)
    fond = os.path.join(REPO_DIR, "tools", "osmo-fond.py")
    if os.path.exists(fond):
        spawn([sys.executable, fond, "antenne" if actif else "ville"])
    self._conky_fondu(actif)

# [2026-10-12] LE CONKY SUIT AUSSI, EN FONDU. Banc a l arret, le tableau de
# bord n a rien a dire (« banc a l arret - ./start-native.sh » deux fois) :
# on le fait disparaitre, et revenir avec le banc. Pas de kill : le gardien
# osmo-desktop-panel relancerait un conky dans les cinq secondes. On agit
# sur sa FENETRE X11 : opacite (_NET_WM_WINDOW_OPACITY, que mutter honore)
# qui descend en une douzaine de pas, puis unmap (xdotool) ; au retour, map
```

```
# puis l opacite remonte. Sans xdotool, l opacite 0 suffit a le cacher.
# Le conky peut etre relance entre-temps (nouvelle fenetre) : _conky_veille
# remet la fenetre dans l etat voulu a chaque tour de refresh_ops.
def _conky_fenetre(self):
    try:
        out = subprocess.run(["xdotool", "search", "--classname", "^conky$"],
                             capture_output=True, text=True, timeout=3).stdout.split()
    except (OSError, subprocess.TimeoutExpired):
        out = []
    if not out:
        try:
            for l in subprocess.run(["wmctrl", "-lx"], capture_output=True, text=True,
             ↪  timeout=3).stdout.splitlines():
                f = l.split()
                if len(f) > 2 and f[2].lower().startswith("conky."):
                    out.append(str(int(f[0], 16)))
        except (OSError, subprocess.TimeoutExpired, ValueError):
            pass
    return out[0] if out else None

@staticmethod
def _conky_opacite(win, v):
    subprocess.run(["xprop", "-id", win, "-f", "_NET_WM_WINDOW_OPACITY", "32c",
                    "-set", "_NET_WM_WINDOW_OPACITY", str(int(max(0.0, min(1.0, v)) * 0xFFFFFFFF))],
                   stdout=subprocess.DEVNULL, stderr=subprocess.DEVNULL, timeout=3)

def _conky_fondu(self, actif, pas=12, duree=1.2):
    if os.environ.get("OSMO_CONKY_TOUJOURS") == "1":
        return
    win = self._conky_fenetre()
    self._conky_voulu = actif
    if not win:
        return
    xdo = shutil.which("xdotool")

    def anim():
        try:
            if actif:
                self._conky_opacite(win, 0.0)
                if xdo:
                    subprocess.run([xdo, "windowmap", win], timeout=3)
                for i in range(1, pas + 1):
                    self._conky_opacite(win, i / pas)
                    time.sleep(duree / pas)
                subprocess.run(["xprop", "-id", win, "-remove", "_NET_WM_WINDOW_OPACITY"],
                               stdout=subprocess.DEVNULL, stderr=subprocess.DEVNULL, timeout=3)
            else:
                for i in range(pas - 1, -1, -1):
                    self._conky_opacite(win, i / pas)
                    time.sleep(duree / pas)
                if xdo:
                    subprocess.run([xdo, "windowunmap", win], timeout=3)
        except (OSError, subprocess.TimeoutExpired):
            pass
        self._conky_win = win
    threading.Thread(target=anim, daemon=True).start()

# [2026-10-12] UN FONDU TUE EN VOL (kill -9 : start-native.sh --stop, avant
# que osmo-fond.py soit dans sa liste d epargnes) laissait le fond sur une
# image intermediaire. Toutes les 10 s, on relit l URI du fond : si ce
# n est pas le fichier attendu pour l etat courant, on relance osmo-fond
# (idempotent, il repart de l image ou il en etait).
def _fond_veille(self):
    if self._visible is None or os.environ.get("OSMO_FOND_TOUJOURS") == "1":
        return
    t = time.monotonic()
```

```
    if t - getattr(self, "_fond_t", 0) < 10:
        return
    self._fond_t = t
    attendu = "gsm-lab-antenne.jpg" if self._visible else "gsm-lab-ville.jpg"
    try:
        uri = subprocess.run(["gsettings", "get", "org.gnome.desktop.background", "picture-uri"],
                             capture_output=True, text=True, timeout=5).stdout.strip().strip("'")
    except (OSError, subprocess.TimeoutExpired):
        return
    if uri.endswith("/" + attendu) or "/osmo-fond/" not in uri and "gsm-lab-" not in uri:
        return          # en place, ou un fond choisi par l utilisateur : on ne touche pas
    fond = os.path.join(REPO_DIR, "tools", "osmo-fond.py")
    if os.path.exists(fond):
        print("[panel] fond d ecran a reprendre : %s" % os.path.basename(uri), flush=True)
        spawn([sys.executable, fond, "antenne" if self._visible else "ville"])

def _conky_veille(self):
    # un conky relance (autre fenetre) alors que le banc est a l arret : on
    # le cache aussitot ; et s il a disparu puis reparu au retour, on le montre
    voulu = getattr(self, "_conky_voulu", None)
    if voulu is None or voulu or os.environ.get("OSMO_CONKY_TOUJOURS") == "1":
        return
    win = self._conky_fenetre()
    if not win:
        return
    # l id X peut etre REUTILISE par un conky relance : on regarde l etat
    # reel de la fenetre, pas seulement son numero
    try:
        mappee = "IsViewable" in subprocess.run(["xwininfo", "-id", win], capture_output=True,
                                                text=True, timeout=3).stdout
    except (OSError, subprocess.TimeoutExpired):
        mappee = False
    if win != getattr(self, "_conky_win", None) or (mappee and shutil.which("xdotool")):
        self._conky_fondu(False, pas=1, duree=0)

# [2026-09-06] SE REPOSER APRES COUP. Meme sous X11, le gestionnaire de
# fenetres peut deplacer la fenetre au moment ou il la mappe (mutter le
# fait pour les fenetres non decorees qu il ne reconnait pas comme du
# bureau). Un seul move() avant show_all() ne tient donc pas : on recale
# au map-event, puis une derniere fois une demi-seconde plus tard.
def _pin(self, *_a):
    self.move(*self._pos)
    return False
def on_draw(self, _w, cr):
    cr.set_operator(cairo.OPERATOR_SOURCE)
    cr.set_source_rgba(0, 0, 0, 0)
    cr.paint()
    return False

# ── l image ──────────────────────────────────────────────────────
def refresh_image(self):
    try:
        mt = os.stat(IMG).st_mtime
    except OSError:
        return True
    if mt == self.mtime:
        return True
    self.mtime = mt
    _, _, w, h = self.geo
    try:
        pb = GdkPixbuf.Pixbuf.new_from_file_at_scale(IMG, w, h, False)
        self.image.set_from_pixbuf(pb)
    except GLib.Error:
        pass  # fichier en cours d ecriture : prochaine seconde
    return True
```

```
# ── les operateurs ──────────────────────────────────────────────────
def refresh_ops(self):
    ops = operators()
    changed = ops != self.ops
    if changed:
        sel = self.ops[self.cur] if self.ops and self.cur < len(self.ops) else None
        self.ops = ops
        self.cur = next((i for i, o in enumerate(ops) if sel and o[0] == sel[0]), 0)
    # Quelqu un d autre a-t-il choisi ? (raccourci clavier, `osmo-op`, une
    # autre session). Si oui, on le suit - sans reecrire le fichier.
    want = read_operator()
    if want is not None:
        i = next((i for i, o in enumerate(self.ops) if o[0] == want), None)
        if i is not None and i != self.cur:
            self.cur = i
            changed = True
    if changed:
        self.update_op(write=(want is None))
    self.montrer(banc_actif(self.ops))
    self._conky_veille()
    self._fond_veille()
    return True

def update_op(self, write=True):
    global VTY_PORT
    # Les fleches sont TOUJOURS la des qu il y a plus d un operateur dans la
    # topologie - qu ils tournent ou non (voir operators()). Un operateur
    # arrete se dit dans le libelle, il ne fait pas disparaitre le bouton.
    multi = len(self.ops) > 1
    for wdg in (self.b_prev, self.b_next, self.l_op):
        wdg.set_visible(multi)
    if self.ops:
        op = self.ops[self.cur]
        etat = "" if op[3] else " arrete"
        libelle = "inter-STP" if op[1] == "interstp" else f"op {op[0]}/{len(self.ops)}"
        self.l_op.set_text(f"{libelle}{etat}")
        self.l_op.set_tooltip_text(
            "\n".join(f"op {o[0]} {o[1]} {o[2] or '(hote)'} "
                      f"{'actif' if o[3] else 'arrete'}" for o in self.ops))
        # Les trois boutons visent l operateur choisi : ils le DISENT.
        # Sans ca, "Dashboard / tmux / VTY 4247" ne changeait pas d aspect
        # quand on poussait les fleches, et rien ne disait sur quel
        # operateur on allait tomber.
        hub = op[1] == "interstp"
        suff = (" hub" if hub else f" op{op[0]}") if len(self.ops) > 1 else ""
        if hub:
            cible = os.environ.get("MULTI_HUB_NAME", "osmo-inter-stp")
        elif op[1] == "docker":
            cible = f"osmo-operator-{op[0]}"
        else:
            cible = f"{op[2] or '127.0.0.1'}"
        self.b_dash.set_label("Dashboard" + suff)
        self.b_dash.set_tooltip_text(f"firefox http://{op[2] or '127.0.0.1'}:{DASH_PORT} ({cible})")
        self.b_tmux.set_label("tmux" + suff)
        self.b_tmux.set_tooltip_text(f"terminal sur la session tmux de {cible}")
        VTY_PORT = vty_port()
        self.b_vty.set_label(f"VTY {VTY_PORT}" + suff)
        self.b_vty.set_tooltip_text(f"terminal : telnet 127.0.0.1 {VTY_PORT} dans {cible}")
        # `write=False` quand on ne fait que SUIVRE le fichier : le
        # reecrire a l identique est inutile, et le reecrire avec notre
        # propre idee est ce qui annulait le choix fait au clavier.
        if write:
            write_operator(op)

def on_prev(self, *_):
    if self.ops:
```

```
        self.cur = (self.cur - 1) % len(self.ops)
        self.update_op()

    def on_next(self, *_):
        if self.ops:
            self.cur = (self.cur + 1) % len(self.ops)
            self.update_op()

    # ── la radio virtuelle et l adresse, rafraichies avec le reste ──────────
    def refresh_zmq(self):
        actif, detail = zmq_etat()
        ctx = self.b_zmq.get_style_context()
        ctx.remove_class("zmq-on" if not actif else "zmq-off")
        ctx.add_class("zmq-on" if actif else "zmq-off")
        self.b_zmq.set_label("ZMQ" if actif else "ZMQ ○")
        self.b_zmq.set_tooltip_text("radio virtuelle srsRAN (ZeroMQ)\n%s\nun clic : qui ecoute, qui est connecte"
         ↪  % detail)
        ip = ip_locale()
        self.l_ip.set_text("IP %s" % ip)
        self.l_ip.set_tooltip_text("adresse de cette machine (un clic la copie)\n\n%s" % ip_toutes())
        return True

    def on_zmq(self, *_):
        ports = "|".join(str(p) for p in ZMQ_PORTS)
        # ss d abord (il nomme le processus), netstat en repli.
        cmd = (f"echo 'Radio virtuelle srsRAN (ZeroMQ) - ports {', '.join(str(p) for p in ZMQ_PORTS)}'; echo;"
               f" (ss -lntp 2>/dev/null || netstat -lntp 2>/dev/null) | grep -E ':({ports})\\b' || echo 'personne
                ↪  n ecoute';"
               f" echo; echo 'connexions etablies :';"
               f" (ss -ntp 2>/dev/null || netstat -ntp 2>/dev/null) | grep -E ':({ports})\\b' || echo 'aucune';"
               f" echo; echo 'processus srsRAN :'; pgrep -a 'srsenb|srsue|srsepc|gnb|srsran' || echo 'aucun';"
               f" echo; read -n1 -rsp 'touche pour fermer...'")
        terminal(cmd)

    def on_ip(self, *_):
        """Copier l adresse : c est ce qu on en fait neuf fois sur dix - la
        recopier dans un fichier de configuration srsRAN ou sur un autre poste."""
        ip = ip_locale()
        try:
            Gtk.Clipboard.get(Gdk.SELECTION_CLIPBOARD).set_text(ip, -1)
            self.l_ip.set_text("IP %s ✓" % ip)
            GLib.timeout_add(1200, lambda: (self.l_ip.set_text("IP %s" % ip), False)[1])
        except Exception:
            pass
        return True

    # ── les boutons ─────────────────────────────────────────────────────────
    def on_dash(self, *_):
        op = self.ops[self.cur] if self.ops else ("1", "native", "", False)
        url = f"http://{op[2] or '127.0.0.1'}:{DASH_PORT}"
        for browser in ("firefox", "chromium", "xdg-open"):
            if subprocess.run(["which", browser], capture_output=True).returncode == 0:
                spawn([browser, url])
                return

    def on_vty(self, *_):
        # telnet, et pas nc : c est ce que la documentation Osmocom donne, et le
        # VTY attend un vrai terminal ligne a ligne. Le `read` final garde la
        # fenetre ouverte quand le port est ferme - sinon elle se refermerait
        # avant qu on ait lu pourquoi.
        #
        # [2026-09-04] LE VTY D UN CONTENEUR EST DANS LE CONTENEUR. On faisait
        # « telnet <ip-du-conteneur> 4247 » : ce port n est PAS publie sur le
        # reseau docker de l operateur (start-docker.sh ne publie que ce dont l hote a
        # besoin), et le bouton rendait "Connection refused" sur un operateur
        # parfaitement vivant. On entre donc dans le conteneur pour le natif du
```

```
        # dedans, exactement comme pour tmux.
        VTY_PORT = vty_port()
        op = self.ops[self.cur] if self.ops else ("1", "native", "", False)
        if op[1] == "interstp":
            # Le hub n a ni mobile ni VTY 4247 : sa console, c est le STP (4239).
            c = os.environ.get("MULTI_HUB_NAME", "osmo-inter-stp")
            cmd = (f"docker exec -it {c} telnet 127.0.0.1 4239"
                   f" || {{ echo; echo 'VTY 4239 injoignable dans {c} (hub arrete ?)'; read -r _; }}")
        elif op[1] == "docker":
            c = f"osmo-operator-{op[0]}"
            cmd = (f"docker exec -it {c} telnet 127.0.0.1 {VTY_PORT}"
                   f" || {{ echo; echo 'VTY {VTY_PORT} injoignable dans {c} (operateur arrete ?)';"
                   f" read -r _; }}")
        else:
            cmd = (f"telnet 127.0.0.1 {VTY_PORT}"
                   f" || {{ echo; echo 'VTY 127.0.0.1:{VTY_PORT} injoignable - banc arrete ?'; read -r _; }}")
        terminal(cmd)

    def on_tmux(self, *_):
        # [2026-09-04] LE NOM DE LA SESSION N ETAIT PAS LE BON. On attachait
        # « tmux -S /tmp/osmocom_tmux attach -t osmocom » dans le conteneur,
        # alors que start-docker.sh y ouvre une session nommee « osmo » sur le socket
        # PAR DEFAUT (voir son commentaire : "docker exec -ti osmo-operator-2
        # tmux attach -t osmo"). Le bouton disait donc "pas de session tmux"
        # sur un operateur qui en avait une. On essaie les deux, dans l ordre
        # ou elles existent, avant de conclure.
        op = self.ops[self.cur] if self.ops else ("1", "native", "", False)
        if op[1] == "interstp":
            c = os.environ.get("MULTI_HUB_NAME", "osmo-inter-stp")
            cmd = (f"docker exec -it {c} bash"
                   f" || {{ echo; echo 'hub {c} injoignable'; read -r _; }}")
        elif op[1] == "docker":
            c = f"osmo-operator-{op[0]}"
            cmd = (f"docker exec -it {c} tmux attach -t osmo"
                   f" || docker exec -it {c} tmux -S /tmp/osmocom_tmux attach -t osmocom"
                   f" || docker exec -it {c} tmux attach"
                   f" || {{ echo; echo 'pas de session tmux dans {c} (operateur arrete ?)';"
                   f" echo 'docker ps -a --filter name={c}'; read -r _; }}")
        else:
            cmd = (f"tmux attach -t {TMUX_SESSION}"
                   f" || tmux -S /tmp/osmocom_tmux attach -t osmocom"
                   f" || {{ echo; echo 'pas de session tmux « {TMUX_SESSION} » - banc arrete ?'; read -r _; }}")
        terminal(cmd)


def single_instance():
    try:
        old = int(open(PID_FILE).read().strip())
        if old != os.getpid():
            os.kill(old, signal.SIGTERM)
            time.sleep(0.3)
    except (OSError, ValueError):
        pass
    # Un verrou qu on ne peut pas ecrire n est pas une raison de ne pas afficher
    # l encart : au pire deux fenetres, jamais une boite de crash.
    try:
        with open(PID_FILE, "w") as f:
            f.write(str(os.getpid()))
    except OSError as e:
        print("osmo-panel : verrou %s non ecrit (%s)" % (PID_FILE, e), file=sys.stderr)


if __name__ == "__main__":
    single_instance()
    Panel()
    Gtk.main()
```

3.424 /opt/GSM/osmo-operator/tools/osmo-phonesim-banc.py

135695 octets, 2911 lignes → 2911 lignes

```
#!/usr/bin/env python3
# osmo-phonesim-banc.py - LE MODEM DU BANC, VU PAR oFONO.
#
# LA CHAINE COMPLETE, ET LA PLACE DE CE FICHIER DEDANS :
#
#   Android → rild → libreference-ril.so → pty AT → osmo-ril-atmodem.py → oFono
#                                                                          ↑
#                                                          CE PROGRAMME (AT sur TCP) ┘
#                                                                  ↓
#                                              le banc : VTY osmo-bsc / osmo-msc
#
# oFono ne fabrique pas de modem : il en PILOTE un. Sur ce banc il n y a pas de
# dongle - « l EC25, pour nous, c est oFono ». Il faut donc donner a oFono
# quelque chose qui se comporte comme un modem : c est le role du plugin
# phonesim, qui ouvre une connexion TCP et y parle AT tout court. Ce programme
# tient l autre bout de cette connexion et repond en lisant LE BANC.
#
# Mise en service :
#   1. /etc/ofono/phonesim.conf :
#          [osmo]
#          Address=127.0.0.1
#          Port=12345
#   2. ce programme (il doit ecouter AVANT qu ofonod cherche le modem)
#   3. systemctl restart ofono  ->  un modem apparait dans org.ofono.Manager
#
# Ce que le modem raconte vient d osmo-bsc (VTY 4242) : le reseau (MCC/MNC,
# nom), l ARFCN, l etat d enregistrement. Un banc arrete = un modem « pas
# enregistre », ce qui est la verite, pas une panne.
import json
import os
import re
import socket
import socketserver
import subprocess
import sys
import threading
import time

REPO = os.environ.get("OSMO_REPO", "/opt/GSM/osmo-operator")
HOST = os.environ.get("OSMO_PHONESIM_HOST", "127.0.0.1")
PORT = int(os.environ.get("OSMO_PHONESIM_PORT", "12345"))
BSC_VTY = (os.environ.get("OSMO_BSC_VTY_HOST", "127.0.0.1"),
           int(os.environ.get("OSMO_BSC_VTY_PORT", "4242")))
MSC_VTY = (os.environ.get("OSMO_MSC_VTY_HOST", "127.0.0.1"),
           int(os.environ.get("OSMO_MSC_VTY_PORT", "4254")))
# Le mobile osmocom-bb. C est LUI l abonne 100101, et c est son pont audio
# (gsm_audio / gsm_mic) que la VM relaie : sans lui, pas de voix.
#
# [2026-09-23] LE PORT PAR DEFAUT N'EST PLUS UNE VALEUR, C'EST UNE LISTE.
# Le mobile du fork QEMU (mobile.cfg) lie 4247, celui du banc DSP
# (c54x_exe/mobile_pont.cfg) lie 4347. start-native.sh passe le bon port
# depuis le 22/09, mais osmo-pmos-setup lance l'instance --connect (le modem de
# la VM) sans OSMO_MOB_VTY_PORT : en --dsp elle parlait a 4247, ou personne
# n'ecoute, et le telephone restait sans reseau et sans appel, sans une
# erreur. Sans la variable, on essaie donc 4247 puis 4347 a chaque connexion ;
# la variable, si elle est posee, reste la seule adresse essayee.
_MOB_HOST = os.environ.get("OSMO_MOB_VTY_HOST", "127.0.0.1")
if os.environ.get("OSMO_MOB_VTY_PORT"):
    MOB_VTY = (_MOB_HOST, int(os.environ["OSMO_MOB_VTY_PORT"]))
else:
    MOB_VTY = [(_MOB_HOST, 4247), (_MOB_HOST, 4347)]
MS = os.environ.get("OSMO_MS_NAME", "1")
```

```
# L abonne que ce modem INCARNE : celui du telephone Android. Sur le banc c est
# 100101 (IMSI 001010001000001), releve par « show subscriber msisdn 100101 »
# sur le VTY d osmo-msc.
IMSI = os.environ.get("OSMO_MS_IMSI", "001010001000001")
MSISDN = os.environ.get("OSMO_MS_MSISDN", "100101")
# La reception des SMS passe par le serveur SMPP d osmo-msc (port 2775) : un
# ESME est deja declare dans la conf du banc (esme msc_tester / osmocom1).
SMPP_HOST = os.environ.get("OSMO_SMPP_HOST", "127.0.0.1")
SMPP_PORT = int(os.environ.get("OSMO_SMPP_PORT", "2775"))
SMPP_ID = os.environ.get("OSMO_SMPP_ID", "msc_tester")
SMPP_PW = os.environ.get("OSMO_SMPP_PASSWORD", "osmocom1")
SMPP_ON = os.environ.get("OSMO_SMPP", "1") == "1"
# Un SMS entrant va-t-il AUSSI au mobile osmocom-bb ? Oui par defaut : Android
# et le mobile sont le meme abonne du banc, et on veut les deux en parallele.
DUPLIQUE = os.environ.get("OSMO_SMS_DUPLIQUE", "1") == "1"
# Les boites d entree des SMS remis par la passerelle du banc. Ce programme
# tourne DEUX fois (le modem d Android en mode ecoute, celui de la VM en mode
# --connect) : il leur faut donc un port chacun. C est scripts/sms_notify.py
# qui les previent, et scripts/sms-interop-relay.py qui l appelle.
SMS_IN_HOST = os.environ.get("OSMO_SMS_IN_HOST", "127.0.0.1")
SMS_IN_LISTEN = int(os.environ.get("OSMO_SMS_IN_PORT", "12348"))
SMS_IN_CONNECT = int(os.environ.get("OSMO_SMS_IN_PORT_VM", "12349"))
# Comment un appel demande par le telephone Android devient un appel du banc.
# {num} = le numero compose, {msisdn} = notre numero. Le canal Local entre dans
# le plan de numerotation d Asterisk (contexte « internal », celui qui porte
# 100/500/600 et la passerelle GSM), puis rebascule sur notre propre numero :
# le banc etablit donc un vrai appel entre le numero compose et le mobile.
CTX = os.environ.get("OSMO_CALL_CONTEXT", "internal")
# Et le contexte par lequel le banc nous APPELLE. Releve sur le banc, sur un
# appel de 100102 vers 100101 : « Event: DialBegin / Context: gsm_in /
# DestExten: 100101 ». C est ce qui distingue un appel entrant de la seconde
# jambe de nos propres appels sortants, qui compose aussi notre numero.
# [2026-09-07] UN SEUL CONTEXTE NE SUFFISAIT PAS : LE TELEPHONE NE SONNAIT
# QUE POUR LES APPELS DE SON PROPRE OPERATEUR. « gsm_in » est le contexte de
# l endpoint gsm_msc, celui par lequel arrive un appel VENU DE LA RADIO
# (100102 -> 100101). Un appel d un AUTRE operateur, lui, entre par le trunk
# interop (endpoint interop_trunk_opN, context=interop_in) et un appel d un
# Linphone par « internal » : dans les deux cas le DialBegin ne portait pas
# « gsm_in », entrant() n etait jamais appele, et AUCUN RING ne partait vers
# oFono. Le mobile osmocom-bb, lui, etait bien pagine par le MSC (la radio
# s allume) - mais le combine, cote AT, ne sonnait pas ; personne ne decrochait
# et le MSC finissait par relacher (MNCC cause DEST_OOO), ce qui remonte en
# 503 chez l appelant. D ou « les appels op3 -> op1 ne passent pas ».
# On accepte donc TOUS les contextes par lesquels un appel peut nous arriver.
# Le filtre qui compte reste DestExten == notre MSISDN : la seconde jambe de
# nos propres appels sortants compose le numero DISTANT, jamais le notre.
# [2026-09-09] « zadarma_in » AJOUTE - SANS LUI, PAS D APPEL DE L EXTERIEUR SUR
# LE TELEPHONE. Un appel venu de l operateur Zadarma entre par le trunk
# [zadarma] / [zadarma_pbx] (pjsip.conf), dont le context est « zadarma_in » :
# le DialBegin porte « Context: zadarma_in / DestExten: 100101 ». Ce contexte ne
# figurait pas dans la liste, entrant() n etait donc jamais appele et AUCUN RING
# ne partait vers ModemManager. Exactement le meme symptome que pour l inter-op
# le 07/09 : le mobile osmocom-bb etait bien pagine (la radio s allume) mais le
# combine restait muet, personne ne decrochait, et l appelant exterieur
# entendait la tonalite jusqu au relachement du MSC.
IN_CTX = {c.strip() for c in os.environ.get(
    "OSMO_CALL_IN_CONTEXT", "gsm_in,interop_in,internal,zadarma_in").split(",") if c.strip()}
# [2026-09-06] L APPEL PASSE PAR L AMI, PAS PAR LA CLI. « channel originate »
# vient du module res_clioriginate, absent de cette installation (modules.conf
# est en autoload = no) : la CLI repondait « No such command ». L AMI, lui, est
# deja actif dans manager.conf ; il n y manquait qu un compte, pose par
# start-native.sh depuis configs/manager-osmo.conf.
AMI_HOST = os.environ.get("OSMO_AMI_HOST", "127.0.0.1")
AMI_PORT = int(os.environ.get("OSMO_AMI_PORT", "5038"))
AMI_USER = os.environ.get("OSMO_AMI_USER", "osmo")
AMI_PASS = os.environ.get("OSMO_AMI_PASSWORD", "osmocom1")
```

```
def clip(num):
    """Le numero de l appelant tel qu il doit etre PRESENTE, et son type 27.007.

    [2026-09-09] LE NUMERO S AFFICHAIT SANS SON « + » SUR LE TELEPHONE. Un appel
    venu de l exterieur par Zadarma arrive avec un appelant en E.164, mais le
    dialplan retire le « + » avant de passer l appel au MSC ([zadarma_in] :
    Set(CALLERID(num)=${FILTER(0-9,...)})), parce que la signalisation GSM ne
    sait pas le porter jusqu au combine. L AMI nous donne donc « 33656678074 »,
    et Phosh affichait ca tel quel : ni reconnu comme un numero francais, ni
    rapproche d un contact du carnet d adresses.

    Le « + » se REMET ici, au moment de la presentation, parce que 27.007 a un
    champ pour ca : le type de numero. 145 = international (le « + » est
    implicite et le telephone le reaffiche), 129 = inconnu/national. On rend
    donc les deux, et l appelant retrouve sa forme +33656678074.

    Regle : un numero deja prefixe d un « + » est international. Sinon, dix
    chiffres ou plus SANS zero de tete l est aussi - les numeros du banc font
    six chiffres (100101) et un national francais commence par 0, donc ni l un
    ni l autre ne peut etre pris pour un international par erreur.
    """
    n = (num or "").strip()
    if not n:
        return "", 129
    if n.startswith("+"):
        return n, 145
    if len(n) >= 10 and n.isdigit() and not n.startswith("0"):
        return "+" + n, 145
    return n, 129


_PROMPT = re.compile(rb'[\r\n][A-Za-z0-9_.\-]+(?:\([^)]*\))?[>#]\s*$')


AT_LOG = os.environ.get("OSMO_AT_LOG_OFONO", "/var/log/osmo-at-ofono.log")
_atlog = None


def log(*a):
    print("[banc]", *a, flush=True)


def atlog(sens, texte):
    """Le journal de CE port-ci : ce qu oFono envoie au modem du banc et ce
    qu il repond. Le pendant de /var/log/osmo-at.log, qui est l autre bout de
    la chaine (le RIL d Android). Les deux ensemble racontent tout le trajet."""
    global _atlog
    if _atlog is None:
        # [2026-09-07] SE RABATTRE SUR /tmp. L instance qui sert la VM tourne
        # sous l utilisateur ordinaire et ne peut pas ecrire dans /var/log :
        # elle perdait donc TOUTE sa trace AT, celle qui dit justement ce que
        # ModemManager fait d un RING ou d un NO CARRIER. Sans trace, plus de
        # diagnostic possible sur le seul bout de chaine qui compte ici.
        for chemin in (AT_LOG, "/tmp/osmo-at-ofono-%s.log" % os.environ.get("USER", "x")):
            try:
                _atlog = open(chemin, "a", buffering=1)
                try:
                    os.chmod(chemin, 0o644)
                except OSError:
                    pass
                if chemin != AT_LOG:
                    print("[banc] trace AT dans %s" % chemin, file=sys.stderr)
                break
            except OSError as e:
                print("[banc] journal %s impossible : %s" % (chemin, e), file=sys.stderr)
```

```
            _atlog = False
    if not _atlog:
        return
    stamp = time.strftime("%H:%M:%S") + ".%03d" % (int(time.time() * 1000) % 1000)
    for ligne in texte.replace("\r", "\n").split("\n"):
        if ligne.strip():
            try:
                _atlog.write("%s %s %s\n" % (stamp, sens, ligne.strip()))
            except OSError:
                return


# ── LE BANC, LU PAR SON VTY ──────────────────────────────────────
class Vty:
    """Une connexion VTY qu on garde ouverte (meme mecanique que
    tools/osmo-ts-probe.py : on lit jusqu au prompt, on ne dort pas)."""

    def __init__(self, addr):
        # Une adresse, ou une liste essayee dans l'ordre (voir MOB_VTY).
        self.addrs = addr if isinstance(addr, list) else [addr]
        self.addr = self.addrs[0]
        self.s = None
        self.lock = threading.Lock()

    def _connect(self):
        for addr in self.addrs:
            try:
                self.s = socket.create_connection(addr, timeout=1.0)
                self.s.settimeout(1.0)
                self.addr = addr
                self._read()
                return True
            except OSError:
                self.s = None
        return False

    def _read(self, limit=262144):
        buf = b""
        while True:
            try:
                chunk = self.s.recv(4096)
            except socket.timeout:
                break
            except OSError:
                self.s = None
                return None
            if not chunk:
                self.s = None
                return None
            buf += chunk
            if _PROMPT.search(buf) or len(buf) > limit:
                break
        return buf

    def cmd(self, text):
        with self.lock:
            if self.s is None and not self._connect():
                return None
            try:
                self.s.sendall(text.encode() + b"\r\n")
            except OSError:
                self.s = None
                if not self._connect():
                    return None
                try:
                    self.s.sendall(text.encode() + b"\r\n")
```

```
                except OSError:
                    self.s = None
                    return None
            out = self._read()
            return out.decode(errors="replace") if out else None


BSC = Vty(BSC_VTY)
MSC = Vty(MSC_VTY)
MOB = Vty(MOB_VTY)


def mobile(action, essais=1):
    """Une commande d appel sur le mobile. Rend True si elle a ete acceptee.

    [2026-09-07] POURQUOI LE MOBILE, ET PAS LA VM. Un ATD de la VM fait lancer
    a Asterisk DEUX jambes : l une joue le service demande (l echo du 600),
    l autre appelle le 100101 - c est-a-dire le MOBILE, pas la VM. Releve sur
    le banc, quatorze canaux restes ouverts, tous par paires :
        Local/600@internal-0000000b;2   Application: Echo
        Local/600@internal-0000000b;1   Application: Congestion
    La jambe qui doit joindre l abonne tombait sur Congestion(), faute de
    decroche : l echo parlait dans le vide, gsm_audio restait a zero (crete 0,
    mesuree) et on n entendait rien. Or c est ce pont-la que la VM relaie. Le
    mobile doit donc repondre - et il n y a personne pour appuyer sur la
    touche. On le fait par son VTY.
    """
    # [2026-09-08] ON ATTEND QUE LA JAMBE ARRIVE AVANT DE DECROCHER. Tenter
    # « call 1 answer » toutes les demi-secondes pendant que la jambe est
    # encore en paging faisait ecrire « % No alerting call » une dizaine de
    # fois sur TOUS les VTY du mobile - dont celui de l utilisateur. Le mobile
    # dit ou il en est dans « show ms » : tant que sa couche RR est « idle »,
    # rien n est encore arrive, on se tait.
    if action == "answer":
        for _ in range(essais * 2):
            out = MOB.cmd("show ms %s" % MS) or ""
            if "radio resource layer state: idle" not in out:
                break
            time.sleep(0.5)
        # Entre la reponse au paging et la sonnerie il reste une a trois
        # secondes : on garde toutes les tentatives (essais), pas quatre -
        # avec quatre, le mobile ne decrochait plus et la voix disparaissait.
    # [2026-09-09] LE DECROCHE REUSSI ETAIT PRIS POUR UN ECHEC. Le mobile
    # confirme par « % (MS 1) % Call is connected » - un message qui commence
    # par « % », comme ses erreurs. Le test « pas de % » ne voyait donc jamais
    # le succes : les 30 tentatives etaient jouees jusqu au bout, soit 29 x
    # « % No alerting call » sur tous les VTY du mobile (dont celui de
    # l utilisateur) pendant 15 s, a CHAQUE appel. Seul « No alerting call »
    # (rien a decrocher encore) justifie de reessayer ; « connected » ou une
    # reponse sans erreur, c est gagne.
    for _ in range(essais):
        MOB.cmd("enable")
        out = MOB.cmd("call %s %s" % (MS, action))
        if out is not None and "No alerting call" not in out and "No call" not in out:
            return True
        time.sleep(0.5)
    return False


def mobile_decroche():
    """Le decroche, en insistant : la jambe vers l abonne met un instant a
    faire sonner le mobile, et une seule tentative arrive trop tot."""
    if mobile("answer", essais=30):
        log("mobile : decroche")
        return True
    log("mobile : rien a decrocher")
```

```
    return False


class Banc:
    """L etat du banc, rafraichi a la demande et garde une seconde en cache -
    un modem est interroge en rafale au demarrage du RIL (CREG, COPS, CSQ...),
    inutile de refaire quatre allers-retours VTY pour la meme reponse."""

    def __init__(self):
        self._t = 0
        self._v = {}

    def state(self):
        if time.time() - self._t < 1.0:
            return self._v
        v = {"up": False, "mcc": "001", "mnc": "01", "name": "Osmocom",
             "arfcn": "", "lac": 0, "cid": 0}
        txt = BSC.cmd("show network")
        if txt:
            v["up"] = True
            # [2026-09-06] LE FORMAT REEL D OsmoBSC, releve sur le banc :
            #   « BSC is on MCC-MNC 001-01 and has 2 BTS »
            # (et non les « Country Code: » / « Network Code: » que cherchait la
            # premiere version - elle ne trouvait donc jamais rien et laissait
            # le modem sur les valeurs par defaut).
            m = re.search(r'MCC-MNC\s+(\d+)-(\d+)', txt)
            if m:
                v["mcc"], v["mnc"] = m.group(1), m.group(2)
            m = re.search(r'Network Name.*?:\s*(\S.*)', txt)
            if m:
                v["name"] = m.group(1).strip()
        txt = BSC.cmd("show bts 0")
        if txt:
            # « ARFCNs: 514 » sur ce banc (le « s » et les deux-points
            # comptaient : « ARFCN\s+ » ne trouvait rien).
            m = re.search(r'ARFCNs?:?\s+(\d+)', txt)
            if m:
                v["arfcn"] = m.group(1)
            m = re.search(r'LAC\s+(\d+)', txt)
            if m:
                v["lac"] = int(m.group(1))
            m = re.search(r'(?:Cell ID|CI)\s+(\d+)', txt)
            if m:
                v["cid"] = int(m.group(1))
        self._v, self._t = v, time.time()
        return v


BANC = Banc()


# ── LA 4G : srsRAN SUR LA RADIO VIRTUELLE ZEROMQ ──────────────────────
# [2026-09-07] LE TELEPHONE EST SUR LA 4G, ET REDESCEND EN 2G POUR PARLER.
# Le banc a deux radios : la 2G d osmocom (osmo-bts-trx + le mobile
# osmocom-bb, qui EST l abonne 100101) et une 4G srsRAN sans materiel
# (srsENB <-> srsUE par ZeroMQ, ports 2000/2001, srsEPC derriere). L eNB
# annonce la 2G a ses UE par le SIB7 de sib.conf (la liste GERAN, mappee dans
# le SIB1 par si_mapping_info = [7]) : c est le CSFB, « CS fallback » - un UE
# pose sur la LTE, qui n a pas de voix, redescend sur la GSM le temps d un
# appel ou d un SMS, puis remonte.
#
# Ce modem fait ce que ferait la puce d un vrai telephone devant ces deux
# reseaux : il campe sur la 4G tant qu elle est la (+CEREG enregistre, AcT 7,
# TAC/ECI de l eNB), et des qu un service CS commence - un ATD, un RING, un
# SMS dans un sens ou l autre - il annonce qu il est passe sur la 2G (URC
# +CREG avec LAC/CI de la BTS et AcT 0) ; le service fini, il remonte. Phosh
```

```
# affiche donc « 4G », puis « 2G » pendant l appel, puis « 4G ».
#
# LA VERITE DE LA 4G, ELLE, VIENT DU BANC, comme celle de la 2G vient du VTY
# d osmo-bsc :
#   - l eNB est la si le port ZeroMQ de srsENB (2000) ecoute ;
#   - l UE est ATTACHE si srsUE a monte son tun (tun_srsue) avec une adresse -
#     il le fait dans son propre espace reseau (--gw.netns=ue1), c est la
#     qu on regarde. Pas de droit d y regarder (pas root) : on s en tient a
#     l eNB. OSMO_LTE_TEMOIN=enb pour ne regarder QUE l eNB, meme en root.
# Les identites (MCC/MNC, TAC, ECI, APN, les ARFCN GERAN du SIB7) sont lues
# dans les fichiers srsRAN eux-memes (OSMO_SRSRAN_DIR, sinon ~/.config/srsran
# de root, sinon /etc/srsran) : ce que le modem raconte est ce que l eNB emet.
ZMQ_PORT = int(os.environ.get("OSMO_ZMQ_PORT", "2000"))
LTE_NETNS = os.environ.get("OSMO_LTE_NETNS", "ue1")
LTE_TUN = os.environ.get("OSMO_LTE_TUN", "tun_srsue")
LTE_TEMOIN = os.environ.get("OSMO_LTE_TEMOIN", "ue")          # ue | enb
LTE_ON = os.environ.get("OSMO_LTE", "1") == "1"
# Le nom sous lequel la 4G apparait au telephone. srsRAN n emet pas de nom de
# reseau ; on en donne un, distinct de celui de la 2G (« Osmocom », lu sur
# osmo-bsc), pour que la liste des reseaux montre bien DEUX entrees.
LTE_NAME = os.environ.get("OSMO_LTE_NAME", "Osmocom 4G")
# [2026-10-15] LA DATA EN 2G : LE MODEM GPRS SUR LE CALYPSO (QEMU + ROM TI).
# Sans 4G, la data du telephone sort par tun_modem, le tunnel du `modem`
# osmocom-bb (contexte PDP, netns msgprs), NATe derriere son adresse ; le GGSN
# rend le tout sur apn0, NATe vers Internet comme le SGi de la 4G. Le Calypso
# n a qu UN client de couche 2 : scripts/calypso-mode.sh le donne au modem
# (ATD*99, data) ou au mobile (appel, SMS : voix). OSMO_GPRS_DATA=0 : coupe.
GPRS_DATA = os.environ.get("OSMO_GPRS_DATA", "1") == "1"
GPRS_NETNS = os.environ.get("OSMO_GPRS_NETNS", "msgprs")
GPRS_TUN = os.environ.get("OSMO_GPRS_TUN", "tun_modem")
GPRS_NET = os.environ.get("OSMO_GPRS_NET", "176.16.32.0/24")
CALYPSO_MODE = os.environ.get("OSMO_CALYPSO_MODE",
                              os.path.join(os.path.dirname(os.path.abspath(__file__)), "..", "scripts",
 ↪  "calypso-mode.sh"))
CALYPSO_MODE_F = "/run/osmo-banc/calypso-mode"


def calypso_mode():
    """voix | data : a qui est le Calypso (scripts/calypso-mode.sh)."""
    try:
        with open(CALYPSO_MODE_F) as f:
            return f.read().strip() or "voix"
    except OSError:
        return "voix"


def calypso(mode, attendre):
    """Donne le Calypso au mobile (voix) ou au modem GPRS (data). attendre :
    jusqu au mobile en service / au PDP actif (bloquant, 2 a 4 min max)."""
    if not os.path.exists(CALYPSO_MODE):
        log("calypso : %s absent, pas de bascule %s" % (CALYPSO_MODE, mode))
        return False
    cmd = ["bash", CALYPSO_MODE, mode] + (["--wait"] if attendre else [])
    if not attendre:
        subprocess.Popen(cmd, stdout=subprocess.DEVNULL, stderr=subprocess.DEVNULL, start_new_session=True)
        log("calypso : bascule en %s lancee" % mode)
        return True
    r = subprocess.run(cmd, capture_output=True, text=True, timeout=300)
    for l in (r.stdout or "").splitlines():
        log("calypso : %s" % l)
    return r.returncode == 0
# Le constructeur annonce a AT+CGMI : il decide du greffon ModemManager, donc
# de la facon de faire la data (voir la reponse a +CGMI et a +GTRNDIS).
VENDOR = os.environ.get("OSMO_MODEM_VENDOR", "Fibocom")
```

```
def _tcp_ecoute(port, host="127.0.0.1"):
    try:
        s = socket.create_connection((host, port), timeout=0.3)
        s.close()
        return True
    except OSError:
        return False


class Lte:
    """La 4G du banc : ce que srsRAN emet, et si notre UE y est."""

    def __init__(self):
        self.dir = self._repertoire()
        self.cfg = self._lire_conf()
        self._t = 0
        self._v = {}
        self._dernier = None
        self._geran_verifie = False

    @staticmethod
    def _repertoire():
        cands = [os.environ.get("OSMO_SRSRAN_DIR", ""), "/root/.config/srsran",
                 os.path.expanduser("~/.config/srsran"), "/etc/srsran"]
        for d in cands:
            if d and os.path.isfile(os.path.join(d, "enb.conf")):
                return d
        return None

    def _texte(self, nom):
        if not self.dir:
            return ""
        try:
            with open(os.path.join(self.dir, nom), errors="replace") as f:
                return f.read()
        except OSError:
            return ""

    @staticmethod
    def _val(texte, cle, defaut):
        """« cle = valeur » (ini ou libconfig, 0x accepte), ou le defaut."""
        m = re.search(r'^\s*%s\s*=\s*"?([0-9A-Za-z_.]+)' % re.escape(cle), texte, re.M)
        if not m:
            return defaut
        v = m.group(1)
        if isinstance(defaut, int):
            try:
                return int(v, 0)
            except ValueError:
                return defaut
        return v

    def _lire_conf(self):
        enb, rr, epc, sib = (self._texte(n) for n in ("enb.conf", "rr.conf", "epc.conf", "sib.conf"))
        c = {"mcc": self._val(enb, "mcc", "001"), "mnc": self._val(enb, "mnc", "01"),
             "enb_id": self._val(enb, "enb_id", 0x19B),
             "earfcn": self._val(enb, "dl_earfcn", 3350),
             "tac": self._val(rr, "tac", self._val(epc, "tac", 7)),
             "cell_id": self._val(rr, "cell_id", 1),
             "apn": self._val(epc, "apn", "srsapn"),
             "geran": [], "sib7_annonce": False}
        # Le SIB7 : la liste GERAN que l eNB donne a ses UE pour le CSFB, et
        # le fait qu il soit bien programme dans le SIB1 (sinon il n est pas
        # emis, et un vrai UE ne saurait pas qu il y a une 2G a cote).
        m = re.search(r'sib7\s*=\s*\{(.*?)\n\}', sib, re.S)
        if m:
```

```
            bloc = m.group(1)
            # Le groupe GERAN du SIB7, c est start_arfcn PLUS la liste qui
            # suit (36.331, CarrierFreqsGERAN) : les deux sont annonces.
            deb = re.search(r'start_arfcn\s*=\s*(\d+)', bloc)
            if deb:
                c["geran"].append(int(deb.group(1)))
            lst = re.search(r'explicit_list_of_arfcns\s*=\s*\(([^)]*)\)', bloc)
            if lst:
                c["geran"] += [int(x) for x in re.findall(r'\d+', lst.group(1))]
        m = re.search(r'si_mapping_info\s*=\s*\[([^\]]*)\]', sib)
        c["sib7_annonce"] = bool(m and re.search(r'\b7\b', m.group(1)))
        return c

    def _ue_attache(self):
        """True/False si on a pu regarder le tun de srsUE, None sinon."""
        # Le tun est dans l espace reseau si srsue a ete lance avec
        # --gw.netns, sur l hote sinon (« srsue » tout court) : on regarde
        # les DEUX. True des qu on le voit ; None si on n a pu regarder nulle
        # part (pas root pour l espace, et pas de tun sur l hote).
        essais = []
        if LTE_NETNS:
            essais.append(["ip", "-n", LTE_NETNS, "-4", "-o", "addr", "show"])
        essais.append(["ip", "-4", "-o", "addr", "show"])
        vu_quelque_part = False
        ns_ferme = False
        for cmd in essais:
            try:
                r = subprocess.run(cmd, capture_output=True, text=True, timeout=3)
            except (OSError, subprocess.TimeoutExpired):
                continue
            if r.returncode != 0:
                # [2026-09-08] « ip -n ue1 » refuse sans root (Operation not
                # permitted) alors que l espace existe : le tun peut y etre
                # sans qu on le voie. Ne pas conclure « pas attache » sur la
                # seule vue de l hote.
                if "-n" in cmd and LTE_NETNS and os.path.exists("/run/netns/" + LTE_NETNS):
                    ns_ferme = True
                continue
            vu_quelque_part = True
            if re.search(r'\b%s\b.*\binet\b' % re.escape(LTE_TUN), r.stdout):
                return True
        if ns_ferme:
            # Sans root, on regarde par les fils de srsue : le fil GW a fait
            # setns() dans l espace, et /proc/<pid>/task/<tid>/net/ se lit
            # par tout le monde. fib_trie y liste les adresses locales : le
            # tun y est avec son adresse des que l attach a abouti.
            vu = self._ue_attache_par_proc()
            if vu is not None:
                return vu
            return None
        return False if vu_quelque_part else None

    def _ue_attache_par_proc(self):
        """True/False d apres /proc/<pid srsue>/task/*/net, None si pas de
        srsue ou rien de lisible."""
        try:
            r = subprocess.run(["pgrep", "-x", "srsue"], capture_output=True, text=True, timeout=3)
        except (OSError, subprocess.TimeoutExpired):
            return None
        pids = r.stdout.split()
        if not pids:
            return False
        lu = False
        for pid in pids:
            try:
                tids = os.listdir("/proc/%s/task" % pid)
```

```
            except OSError:
                continue
            for tid in tids:
                base = "/proc/%s/task/%s/net/" % (pid, tid)
                try:
                    with open(base + "dev") as f:
                        dev = f.read()
                except OSError:
                    continue
                lu = True
                if LTE_TUN + ":" not in dev:
                    continue
                try:
                    with open(base + "fib_trie") as f:
                        trie = f.read()
                except OSError:
                    continue
                # Une adresse LOCAL autre que 127.x dans l espace du tun :
                # srsue l a recue de l EPC.
                if re.search(r'^\s+\|--\s+(?!127\.)\d+\.\d+\.\d+\.\d+\n\s+/32 host LOCAL', trie, re.M):
                    return True
        return False if lu else None

    def state(self):
        if time.time() - self._t < 1.0:
            return self._v
        v = dict(self.cfg)
        v["eci"] = ((v["enb_id"] & 0xFFFFF) << 8) | (v["cell_id"] & 0xFF)
        v["enb"] = LTE_ON and _tcp_ecoute(ZMQ_PORT)
        v["ue"] = self._ue_attache() if v["enb"] else False
        if LTE_TEMOIN == "enb":
            v["ok"] = v["enb"]
        else:
            v["ok"] = v["enb"] and v["ue"] is not False
        etat = (v["enb"], v["ok"])
        if etat != self._dernier:
            if v["ok"]:
                log("4G : eNB en ligne (ZeroMQ %d), UE %s - TAC %04X, ECI %08X, APN %s"
                    % (ZMQ_PORT, "attache" if v["ue"] else "non observable, on suit l eNB",
                       v["tac"], v["eci"], v["apn"]))
            elif not v["enb"]:
                log("4G : eNB absent (rien n ecoute sur le port ZeroMQ %d)" % ZMQ_PORT)
            else:
              log("4G : eNB en ligne mais srsUE pas attache (pas de %s avec une adresse, ni dans %s ni sur l hote)"
                    % (LTE_TUN, LTE_NETNS or "-"))
            self._dernier = etat
        self._v, self._t = v, time.time()
        return v

    def resume(self):
        c = self.cfg
        if not self.dir:
            log("4G : aucune configuration srsRAN trouvee (OSMO_SRSRAN_DIR) - valeurs par defaut")
        else:
            log("4G : configuration srsRAN dans %s" % self.dir)
        log("4G : PLMN %s%s, EARFCN %d, TAC %04X, eNB 0x%X, APN %s"
            % (c["mcc"], c["mnc"], c["earfcn"], c["tac"], c["enb_id"], c["apn"]))
        if c["geran"]:
            log("4G : SIB7 annonce la 2G sur ARFCN %s%s"
                % (", ".join(str(a) for a in c["geran"]),
                   "" if c["sib7_annonce"] else
                   " - MAIS le SIB7 n est pas dans si_mapping_info du SIB1 : il n est pas emis"))
        else:
            log("4G : pas de SIB7 (liste GERAN) dans sib.conf - pas de CSFB annonce aux UE")

    def verifier_geran(self, arfcn_bts):
```

```
        """Une fois : la 2G que le SIB7 annonce est-elle celle du banc ?"""
        if self._geran_verifie or not arfcn_bts:
            return
        self._geran_verifie = True
        try:
            a = int(arfcn_bts)
        except ValueError:
            return
        if self.cfg["geran"] and a not in self.cfg["geran"]:
            log("4G : ATTENTION, le SIB7 annonce ARFCN %s mais la BTS du banc est sur %d - "
                "un vrai UE ne trouverait pas la 2G pour son CSFB (sib.conf : explicit_list_of_arfcns)"
                % (", ".join(str(x) for x in self.cfg["geran"]), a))


LTE = Lte()


# ── ASTERISK, PAR L AMI ──────────────────────────────────────────────
def _ami(actions, wait=3.0):
    """Ouvre une session AMI, joue les actions, rend la reponse brute."""
    try:
        s = socket.create_connection((AMI_HOST, AMI_PORT), timeout=5)
    except OSError as e:
        return None, "AMI injoignable (%s)" % e
    try:
        s.settimeout(wait)
        s.recv(4096)                                  # la banniere
        s.sendall(("Action: Login\r\nUsername: %s\r\nSecret: %s\r\n\r\n"
                   % (AMI_USER, AMI_PASS)).encode())
        rep = s.recv(4096).decode(errors="replace")
        if "Success" not in rep:
            return None, "connexion AMI refusee"
        for act in actions:
            s.sendall(act.encode())
        out = b""
        end = time.time() + wait
        while time.time() < end:
            try:
                chunk = s.recv(4096)
            except socket.timeout:
                break
            if not chunk:
                break
            out += chunk
            if b"Response: Success" in out or b"Response: Error" in out:
                break
        return out.decode(errors="replace"), ""
    except OSError as e:
        return None, str(e)
    finally:
        try:
            s.sendall(b"Action: Logoff\r\n\r\n")
            s.close()
        except OSError:
            pass


def ami_originate(num):
    """Fait naitre l appel : le numero compose est appele dans le plan du banc,
    et une fois decroche il est mis en relation avec notre propre numero - donc
    avec le mobile de l abonne. Rend (ok, canal ou raison)."""
    chan = "Local/%s@%s" % (num, CTX)
    # [2026-09-09] ECHO_NO_PROMPT=1 : le 600 d Asterisk saute son annonce
    # (demo-echotest) et passe droit a Echo. Avec l enregistrement de la
    # jambe ;1 (sub-record), l enchainement Playback -> Echo faisait lacher a
    # Asterisk une rafale de milliers de paquets RTP de silence en 60 ms vers
```

```
    # le BTS - « une partie de l appel avec un sale son ». Voir le commentaire
    # du 600 dans configs/extensions.conf ; sans effet sur les autres numeros.
    # L annonce de l echo test est rejouee sur la patte du mobile (option A()
    # du Dial, via DIAL_OPTS) : la, elle ne declenche pas la rafale.
    extra = "Variable: DIAL_OPTS=A(demo-echotest)\r\n" if num == "600" else ""
    act = ("Action: Originate\r\n"
           "Channel: %s\r\n"
           "Context: %s\r\n"
           "Exten: %s\r\n"
           "Priority: 1\r\n"
           "CallerID: %s\r\n"
           "Variable: ECHO_NO_PROMPT=1\r\n"
           "%s"
           "Async: true\r\n\r\n" % (chan, CTX, MSISDN, MSISDN, extra))
    out, err = _ami([act], wait=5.0)
    if out is None:
        return False, err
    if "Response: Error" in out:
        m = re.search(r"Message: (.+)", out)
        return False, (m.group(1).strip() if m else "refus d Asterisk")
    return True, chan


def ami_hangup(channel):
    if not channel:
        return
    _ami(["Action: Hangup\r\nChannel: %s\r\n\r\n" % channel], wait=2.0)


def _bloc(txt):
    """Un bloc d evenement AMI (« Cle: valeur » par ligne) rendu en dict."""
    d = {}
    for ligne in txt.split("\r\n"):
        cle, sep, val = ligne.partition(":")
        if sep:
            d[cle.strip()] = val.strip()
    return d


class AmiEcoute(threading.Thread):
    """L oreille sur Asterisk.

    [2026-09-07] SANS ELLE, NI SONNERIE NI RACCROCHE. Ce modem ne savait que
    COMPOSER. Il n emettait aucun RING : ModemManager n avait donc rien a
    montrer et les appels entrants n existaient pas dans l interface. Et il
    n apprenait jamais qu un appel s etait termine : quand le correspondant
    raccrochait, l ecran d appel restait affiche. Les deux manques ont la meme
    cause - personne n ecoutait Asterisk - et donc le meme remede.

    Ce qu on guette, releve sur le banc pour un appel de 100102 vers 100101 :
        Event: DialBegin      Context: gsm_in
        CallerIDNum: 100102   DestExten: 100101
        DestChannel: PJSIP/gsm_msc-0000000f
    puis Hangup sur ce meme canal pour la fin. Le contexte compte : nos propres
    appels sortants font eux aussi composer notre numero, mais depuis
    « internal » - sans ce filtre le modem se sonnerait lui-meme.
    """

    daemon = True

    def run(self):
        while True:
            try:
                self._session()
            except OSError as e:
                log("AMI : lien perdu (%s)" % e)
```

```
            time.sleep(5)

    def _session(self):
        s = socket.create_connection((AMI_HOST, AMI_PORT), timeout=5)
        try:
            s.settimeout(None)
            s.recv(4096)                                    # la banniere
            s.sendall(("Action: Login\r\nUsername: %s\r\nSecret: %s\r\n"
                       "Events: on\r\n\r\n" % (AMI_USER, AMI_PASS)).encode())
            log("AMI : a l ecoute des appels")
            buf = b""
            while True:
                d = s.recv(8192)
                if not d:
                    return
                buf += d
                while b"\r\n\r\n" in buf:
                    blk, buf = buf.split(b"\r\n\r\n", 1)
                    self._evenement(_bloc(blk.decode(errors="replace")))
        finally:
            try:
                s.close()
            except OSError:
                pass

    def _evenement(self, e):
        modem = CURRENT
        if modem is None:
            return
        ev = e.get("Event")
        if ev == "DialBegin" and e.get("Context") in IN_CTX \
                and e.get("DestExten") == MSISDN:
            # La seconde jambe de nos propres appels compose NOTRE numero : sans
            # ce test, le telephone sonne pour l appel qu il vient de passer.
            if modem.est_notre_jambe(e.get("Channel") or ""):
                return
            modem.entrant(e.get("CallerIDNum") or "", e.get("DestChannel") or "")
        elif ev == "Newchannel":
            # [2026-09-07] LE NOM QU ON GARDAIT N EN ETAIT PAS UN.
            # ami_originate range « Local/600@internal », qui est la DEMANDE.
            # Asterisk, lui, cree « Local/600@internal-0000000b;1 » : le
            # Action: Hangup sur l ancien nom ne trouvait rien et ne raccrochait
            # donc jamais - d ou les quatorze canaux empiles sur le banc. On
            # note le vrai nom des qu il apparait.
            # [2026-09-07] ON LATCHE LE PREMIER, PAS LE DERNIER. Un canal Local
            # arrive par PAIRES (« ...;1 » et « ...;2 », les deux evenements a
            # la meme milliseconde) : la boucle acceptait les deux et le nom
            # retenu finissait sur « ;2 ». On ne remplace donc le nom que tant
            # qu il vaut encore la DEMANDE (c["origine"]).
            chan = e.get("Channel") or ""
            for c in modem.calls:
                ref = c.get("channel")
                if ref and chan.startswith(ref) and chan != ref \
                        and c.get("channel") == c.get("origine"):
                    c["channel"] = chan
        elif ev == "Hangup" or (ev == "DialEnd" and e.get("DialStatus") != "ANSWER"):
            for chan in (e.get("Channel"), e.get("DestChannel")):
                if chan:
                    modem.fin_appel(chan)


# ── LA DATA : UN SERVEUR PPP AU BOUT DU PORT AT ──────────────────────────
# [2026-09-07] POURQUOI PPP, ET PAS AUTRE CHOSE. ModemManager, devant un modem
# qui n a qu un port AT, ne connait qu une facon de faire la data : composer
# ATD*99***1#, attendre CONNECT, et laisser NetworkManager lancer pppd sur le
# port serie. Les greffons qui savent se passer de PPP (Fibocom +GTRNDIS,
```

```
# Huawei ^NDISDUP, Cinterion ^SWWAN) exigent tous un port reseau USB - soit
# pour reconnaitre le modem a son identifiant USB, soit pour numeroter le port
# (bInterfaceNumber) ; sur un port serie PCI de QEMU, aucun ne s applique.
# Verifie dans les sources de ModemManager le 2026-09-07. Il faut donc que le
# telephone AIT du PPP (les modules ppp_generic/ppp_async, compiles pour son
# noyau) et que ce modem parle PPP a l autre bout : c est ce bloc.
#
# Ce qu on fait, et rien de plus (RFC 1661, 1662, 1332) :
#   - le cadrage HDLC asynchrone (0x7e, echappement 0x7d, FCS-16) ;
#   - LCP : on accepte MRU, ACCM, nombre magique, PFC et ACFC, on rejette le
#     reste (dont l authentification : le banc n en demande pas) ; on repond
#     aux Echo-Request ; un Terminate-Request finit la session ;
#   - IPCP : on DONNE au telephone son adresse et ses DNS par Configure-Nak
#     (c est ainsi qu un operateur les attribue), on refuse la compression VJ ;
#   - IPv6CP et CCP : Protocol-Reject, pppd s en passe ;
#   - les paquets IP vont dans un tun, et ce tun est mis DANS L ESPACE RESEAU
#     DE srsUE (--gw.netns) avec un NAT vers tun_srsue : chaque paquet du
#     telephone traverse alors la radio 4G, l eNB, l EPC et le SGi. Si srsUE
#     tourne sur l hote (sans netns), on route quand meme par tun_srsue, mais
#     le retour, lui, prend le raccourci local - l UE et l EPC etant la meme
#     machine, le noyau livre en interne ce qui est adresse a une adresse
#     locale. On le dit dans le journal.
import fcntl
import random
import struct

# [2026-09-08] Les adresses du lien PPP ne doivent PAS etre celles que l EPC
# donne a l UE. Avec open5gs, srsUE recoit 10.45.0.2 ; le lien PPP en
# 10.45.0.1/.2 mettait alors l adresse du telephone en LOCAL sur tun_srsue,
# dans le meme espace reseau : le retour du NAT etait livre a l hote au lieu
# de repartir vers le telephone. Un /30 a part, que rien d autre n emploie.
PPP_LOCAL = os.environ.get("OSMO_PPP_LOCAL", "10.99.0.1")      # nous, le reseau
PPP_PEER = os.environ.get("OSMO_PPP_PEER", "10.99.0.2")        # le telephone
PPP_DNS = [d.strip() for d in os.environ.get("OSMO_PPP_DNS", "8.8.8.8,8.8.4.4").split(",") if d.strip()]
PPP_IF = os.environ.get("OSMO_PPP_IF", "ppp-pmos")


def _sgi_net_defaut():
    """Le sous-reseau des UE, pour la sortie SGi -> Internet de l hote :
    celui de smf.yaml avec open5gs (10.45.0.0/16), 172.16.0.0/24 avec srsEPC."""
    # [2026-09-16] Le prefixe du banc (/opt/LTE/open5gs/install, celui d
    # osmo-epc.sh) manquait : on retombait sur 172.16.0.0/24 et la MASQUERADE
    # posee ici ne couvrait pas les UE open5gs (10.45.0.0/16).
    for f in ([os.path.join(os.environ["OPEN5GS_PREFIX"], "etc/open5gs/smf.yaml")] if
     ↪  os.environ.get("OPEN5GS_PREFIX") else []) + [
            "/opt/LTE/open5gs/install/etc/open5gs/smf.yaml",
            "/root/open5gs/install/etc/open5gs/smf.yaml", "/etc/open5gs/smf.yaml"]:
        try:
            m = re.search(r'^\s*-\s*subnet:\s*(\d+\.\d+\.\d+\.\d+/\d+)', open(f).read(), re.M)
        except OSError:
            continue
        if m:
            return m.group(1)
    return "172.16.0.0/24"


PPP_SGI_NET = os.environ.get("OSMO_LTE_SGI_NET") or _sgi_net_defaut()

_FCS_TAB = []
for _b in range(256):
    _v = _b
    for _ in range(8):
        _v = (_v >> 1) ^ 0x8408 if _v & 1 else _v >> 1
    _FCS_TAB.append(_v)
```

```
def ppp_fcs(data, fcs=0xFFFF):
    for b in data:
        fcs = (fcs >> 8) ^ _FCS_TAB[(fcs ^ b) & 0xFF]
    return fcs


def _sh(*cmd):
    """Une commande systeme, sans lever : rend (rc, sortie)."""
    try:
        r = subprocess.run(cmd, capture_output=True, text=True, timeout=5)
        return r.returncode, (r.stdout + r.stderr).strip()
    except (OSError, subprocess.TimeoutExpired) as e:
        return 1, str(e)


class Ppp:
    LCP, IPCP, IP, IPV6CP, CCP = 0xC021, 0x8021, 0x0021, 0x8057, 0x80FD
    CONF_REQ, CONF_ACK, CONF_NAK, CONF_REJ = 1, 2, 3, 4
    TERM_REQ, TERM_ACK, CODE_REJ, PROTO_REJ, ECHO_REQ, ECHO_REP = 5, 6, 7, 8, 9, 10

    def __init__(self, modem):
        self.modem = modem
        self.magic = struct.pack(">I", random.getrandbits(32))
        self.buf = bytearray()
        self.in_frame = False
        self.esc = False
        self.brut = bytearray()             # ce qui arrive HORS trame (un AT ?)
        self.ident = 0
        self.lcp_ack_recu = self.lcp_ack_donne = self.lcp_ouvert = False
        self.ipcp_ack_recu = self.ipcp_ack_donne = self.ipcp_ouvert = False
        self.tun = None
        self.ns = None
        self.fini = False
        self.lock = threading.Lock()
        self.octets = [0, 0]                # montant, descendant
        self._motif_fin = None              # pose par terminer()

    def terminer(self, motif):
        """C est NOUS qui fermons la session : Terminate-Request a pppd, qui
        repond Terminate-Ack et s arrete ; s il ne repond pas dans la seconde,
        on ferme quand meme. fin() dira NO CARRIER au port, ModemManager
        reprend alors la main en mode commande."""
        if self.fini or self._motif_fin:
            return
        self._motif_fin = motif
        self._paquet(self.LCP, self.TERM_REQ, self._prochain(), b"")
        t = threading.Timer(1.0, lambda: self.fin(motif))
        t.daemon = True
        t.start()

    # -- la couche HDLC
    def feed(self, data):
        for b in data:
            if b == 0x7E:
                if self.in_frame and len(self.buf) >= 4:
                    self._trame(bytes(self.buf))
                self.buf.clear()
                self.in_frame = True
                self.esc = False
                self.brut.clear()
                continue
            if not self.in_frame:
                # Hors trame : le port a pu repasser en AT (ModemManager qui
                # reprend la main apres la mort de pppd). « AT » suivi d une
                # fin de ligne, et on rend le port.
                self.brut += bytes([b])
```

```
            if b in (0x0D, 0x0A):
                ligne = self.brut.decode(errors="replace").strip().upper()
                self.brut.clear()
                if ligne.startswith("AT") or ligne == "+++":
                    self.fin("commande AT recue en mode donnees (%s)" % ligne)
                    return
            if len(self.brut) > 64:
                del self.brut[:32]
            continue
        if self.esc:
            self.buf.append(b ^ 0x20)
            self.esc = False
        elif b == 0x7D:
            self.esc = True
        else:
            self.buf.append(b)

def _trame(self, f):
    if ppp_fcs(f) != 0xF0B8:
        return                          # FCS faux : on jette
    f = f[:-2]
    if f[:2] == b"\xff\x03":
        f = f[2:]
    if not f:
        return
    if f[0] & 1:                        # protocole sur un octet (PFC)
        proto, pkt = f[0], f[1:]
    else:
        proto, pkt = struct.unpack(">H", f[:2])[0], f[2:]
    if proto == self.IP:
        self._vers_tun(pkt)
    elif proto == self.LCP:
        self._lcp(pkt)
    elif proto == self.IPCP:
        self._ipcp(pkt)
    else:
        self._proto_rejet(proto, pkt)

def envoyer(self, proto, payload):
    f = b"\xff\x03" + struct.pack(">H", proto) + payload
    fcs = ppp_fcs(f) ^ 0xFFFF
    f += bytes([fcs & 0xFF, fcs >> 8])
    out = bytearray([0x7E])
    for b in f:
        if b < 0x20 or b in (0x7D, 0x7E):
            out += bytes([0x7D, b ^ 0x20])
        else:
            out.append(b)
    out.append(0x7E)
    with self.lock:
        try:
            self.modem.wfile.write(bytes(out))
            self.modem.wfile.flush()
        except OSError:
            self.fin("port ferme")

# -- les paquets de negociation
@staticmethod
def _options(data):
    opts, i = [], 0
    while i + 2 <= len(data):
        t, ln = data[i], data[i + 1]
        if ln < 2 or i + ln > len(data):
            break
        opts.append((t, data[i + 2:i + ln]))
        i += ln
```

```
    return opts

@staticmethod
def _opt(t, v):
    return bytes([t, 2 + len(v)]) + v

def _paquet(self, proto, code, ident, body):
    self.envoyer(proto, bytes([code, ident]) + struct.pack(">H", 4 + len(body)) + body)

def _prochain(self):
    self.ident = (self.ident + 1) & 0xFF or 1
    return self.ident

def demarrer(self):
    log("PPP : session ouverte (CONNECT), le telephone doit lancer pppd")
    self._lcp_requete()

def _lcp_requete(self):
    self._paquet(self.LCP, self.CONF_REQ, self._prochain(),
                 self._opt(2, b"\x00\x00\x00\x00") + self._opt(5, self.magic))

def _lcp(self, pkt):
    if len(pkt) < 4:
        return
    code, ident, ln = pkt[0], pkt[1], struct.unpack(">H", pkt[2:4])[0]
    body = pkt[4:ln]
    if code == self.CONF_REQ:
        rejet = b""
        for t, v in self._options(body):
            if t not in (1, 2, 5, 7, 8):     # MRU, ACCM, magique, PFC, ACFC
                rejet += self._opt(t, v)
        if rejet:
            self._paquet(self.LCP, self.CONF_REJ, ident, rejet)
        else:
            self._paquet(self.LCP, self.CONF_ACK, ident, body)
            self.lcp_ack_donne = True
        if not self.lcp_ack_recu:
            self._lcp_requete()
    elif code == self.CONF_ACK:
        self.lcp_ack_recu = True
    elif code in (self.CONF_NAK, self.CONF_REJ):
        # On n insiste pas sur l ACCM : le nombre magique suffit.
        self._paquet(self.LCP, self.CONF_REQ, self._prochain(), self._opt(5, self.magic))
    elif code == self.TERM_REQ:
        self._paquet(self.LCP, self.TERM_ACK, ident, body)
        self.fin("le telephone a termine la session PPP")
        return
    elif code == self.TERM_ACK and self._motif_fin:
        self.fin(self._motif_fin)
        return
    elif code == self.ECHO_REQ:
        self._paquet(self.LCP, self.ECHO_REP, ident, self.magic + body[4:])
    if self.lcp_ack_recu and self.lcp_ack_donne and not self.lcp_ouvert:
        self.lcp_ouvert = True
        log("PPP : LCP ouvert, on passe a IPCP")
        self._ipcp_requete()

def _proto_rejet(self, proto, pkt):
    if proto in (self.IPV6CP, self.CCP) or self.lcp_ouvert:
        self._paquet(self.LCP, self.PROTO_REJ, self._prochain(),
                     struct.pack(">H", proto) + pkt[:8])

def _ipcp_requete(self):
    self._paquet(self.IPCP, self.CONF_REQ, self._prochain(),
                 self._opt(3, socket.inet_aton(PPP_LOCAL)))
```

```
def _ipcp(self, pkt):
    if len(pkt) < 4:
        return
    code, ident, ln = pkt[0], pkt[1], struct.unpack(">H", pkt[2:4])[0]
    body = pkt[4:ln]
    if code == self.CONF_REQ:
        rejet, nak = b"", b""
        dns = PPP_DNS + PPP_DNS[:1]
        for t, v in self._options(body):
            if t == 3:                    # l adresse du telephone
                if v != socket.inet_aton(PPP_PEER):
                    nak += self._opt(3, socket.inet_aton(PPP_PEER))
            elif t == 129:                # DNS primaire
                if v != socket.inet_aton(dns[0]):
                    nak += self._opt(129, socket.inet_aton(dns[0]))
            elif t == 131:                # DNS secondaire
                if v != socket.inet_aton(dns[1]):
                    nak += self._opt(131, socket.inet_aton(dns[1]))
            else:                         # compression VJ et le reste
                rejet += self._opt(t, v)
        if rejet:
            self._paquet(self.IPCP, self.CONF_REJ, ident, rejet)
        elif nak:
            self._paquet(self.IPCP, self.CONF_NAK, ident, nak)
        else:
            self._paquet(self.IPCP, self.CONF_ACK, ident, body)
            self.ipcp_ack_donne = True
        if not self.ipcp_ack_recu:
            self._ipcp_requete()
    elif code == self.CONF_ACK:
        self.ipcp_ack_recu = True
    elif code in (self.CONF_NAK, self.CONF_REJ):
        self._ipcp_requete()
    elif code == self.TERM_REQ:
        self._paquet(self.IPCP, self.TERM_ACK, ident, body)
    if self.ipcp_ack_recu and self.ipcp_ack_donne and not self.ipcp_ouvert:
        self.ipcp_ouvert = True
        self._tun_monter()

# -- le tun, et son branchement sur la radio
def _tun_monter(self):
    TUNSETIFF, IFF_TUN, IFF_NO_PI = 0x400454CA, 0x0001, 0x1000
    try:
        fd = os.open("/dev/net/tun", os.O_RDWR)
        fcntl.ioctl(fd, TUNSETIFF, struct.pack("16sH", PPP_IF.encode(), IFF_TUN | IFF_NO_PI))
    except OSError as e:
        log("PPP : impossible de creer %s (%s) - il faut root ; la session reste sans IP" % (PPP_IF, e))
        return
    self.tun = fd
    self.ns = None
    _sh("ip", "addr", "replace", PPP_LOCAL, "peer", PPP_PEER + "/32", "dev", PPP_IF)
    _sh("ip", "link", "set", PPP_IF, "up", "mtu", "1500")
    if not self._radio_brancher():
        if GPRS_DATA:
            log("PPP : IPCP ouvert, %s au telephone ; sortie radio en attente du PDP GPRS (%s dans %s)"
                % (PPP_PEER, GPRS_TUN, GPRS_NETNS))
            threading.Thread(target=self._attendre_radio, daemon=True).start()
        elif not self._raccourci_test():
            log("PPP : IPCP ouvert, %s au telephone, mais pas de %s : la data n a pas de sortie radio"
                % (PPP_PEER, LTE_TUN))
    threading.Thread(target=self._depuis_tun, daemon=True).start()

def _raccourci_test(self):
    """RACCOURCI DE TEST (sans radio, sans GGSN), OSMO_PPP_SHORTCUT=1 : le
    telephone sort par la carte par defaut de l hote, en NAT derriere son
    adresse, comme le SGi de srsEPC le ferait. Le vrai EGPRS passe par le
```

```
        SGSN/GGSN ; ce chemin ne sert qu a valider DNS, NAT et pile IP du
        telephone. (repris d origin/main, fusion du 2026-10-16)"""
        if os.environ.get("OSMO_PPP_SHORTCUT", "0") != "1":
            return False
        rc, up = _sh("sh", "-c", "ip -4 route show default | awk '{for(i=1;i<=NF;i++) if($i==\"dev\") print
↪ $(i+1); exit}'")
        if not up:
            return False
        _sh("sysctl", "-qw", "net.ipv4.ip_forward=1")
        rc, _ = _sh("iptables", "-w", "-t", "nat", "-C", "POSTROUTING", "-s", PPP_PEER, "-o", up, "-j",
↪ "MASQUERADE")
        if rc != 0:
            _sh("iptables", "-w", "-t", "nat", "-A", "POSTROUTING", "-s", PPP_PEER, "-o", up, "-j", "MASQUERADE")
        _sh("iptables", "-w", "-N", "DOCKER-USER")
        for spec in (("-s", PPP_PEER), ("-d", PPP_PEER)):
            rc, _ = _sh("iptables", "-w", "-C", "DOCKER-USER", *spec, "-j", "ACCEPT")
            if rc != 0:
                _sh("iptables", "-w", "-I", "DOCKER-USER", "1", *spec, "-j", "ACCEPT")
        rc, _ = _sh("iptables", "-w", "-C", "FORWARD", "-j", "DOCKER-USER")
        if rc != 0:
            _sh("iptables", "-w", "-I", "FORWARD", "1", "-j", "DOCKER-USER")
        log("PPP : IPCP ouvert, %s au telephone - RACCOURCI DE TEST : sortie directe par %s (sans GGSN)"
            % (PPP_PEER, up))
        return True

    def _radio_choix(self):
        """(netns ou None, tun, reseau a NATer vers Internet) : la 4G si srsUE a
        son tun, sinon le PDP GPRS du modem sur le Calypso, sinon None."""
        if LTE_NETNS:
            rc, o = _sh("ip", "-n", LTE_NETNS, "-4", "-o", "addr", "show", LTE_TUN)
            if rc == 0 and "inet" in o:
                return LTE_NETNS, LTE_TUN, PPP_SGI_NET
        rc, o = _sh("ip", "-4", "-o", "addr", "show", LTE_TUN)
        if rc == 0 and "inet" in o:
            return None, LTE_TUN, PPP_SGI_NET
        if GPRS_DATA and GPRS_NETNS:
            rc, o = _sh("ip", "-n", GPRS_NETNS, "-4", "-o", "addr", "show", GPRS_TUN)
            if rc == 0 and "inet" in o:
                return GPRS_NETNS, GPRS_TUN, GPRS_NET
        return None

    def _radio_brancher(self):
        """Branche PPP_IF sur la sortie radio disponible. False si aucune."""
        choix = self._radio_choix()
        if choix is None or self.tun is None:
            return False
        ns, tun, sgi = choix
        if ns != self.ns:
            cur = ["ip", "-n", self.ns] if self.ns else ["ip"]
            _sh(*cur, "link", "set", PPP_IF, "netns", ns if ns else "1")
            self.ns = ns
        ip = ["ip", "-n", self.ns] if self.ns else ["ip"]
        ex = ["ip", "netns", "exec", self.ns] if self.ns else []
        _sh(*ip, "addr", "replace", PPP_LOCAL, "peer", PPP_PEER + "/32", "dev", PPP_IF)
        _sh(*ip, "link", "set", PPP_IF, "up", "mtu", "1500")
        _sh(*ex, "sysctl", "-qw", "net.ipv4.ip_forward=1")
        _sh(*ex, "sysctl", "-qw", "net.ipv4.conf.%s.rp_filter=0" % PPP_IF)
        if self.ns:
            _sh(*ip, "route", "replace", "default", "dev", tun)
        else:
            _sh("ip", "route", "replace", "default", "dev", tun, "table", "45")
            rc, o = _sh("ip", "rule", "show")
            if PPP_PEER not in o:
                _sh("ip", "rule", "add", "from", PPP_PEER, "lookup", "45", "priority", "4500")
        rc, _ = _sh(*ex, "iptables", "-w", "-t", "nat", "-C", "POSTROUTING", "-s", PPP_PEER, "-o", tun, "-j",
↪ "MASQUERADE")
```

```
        if rc != 0:
            _sh(*ex, "iptables", "-w", "-t", "nat", "-A", "POSTROUTING", "-s", PPP_PEER, "-o", tun, "-j",
↪   "MASQUERADE")
        # La sortie vers Internet du reseau de l UE (SGi 4G ou pool du GGSN)
        rc, up = _sh("sh", "-c", "ip -4 route show default | awk '{for(i=1;i<=NF;i++) if($i==\"dev\") print
↪   $(i+1); exit}'")
        if up:
            _sh("sysctl", "-qw", "net.ipv4.ip_forward=1")
           rc, _ = _sh("iptables", "-w", "-t", "nat", "-C", "POSTROUTING", "-s", sgi, "-o", up, "-j", "MASQUERADE")
            if rc != 0:
                _sh("iptables", "-w", "-t", "nat", "-A", "POSTROUTING", "-s", sgi, "-o", up, "-j", "MASQUERADE")
            _sh("iptables", "-w", "-N", "DOCKER-USER")
            for spec in (("-s", sgi), ("-d", sgi)):
                rc, _ = _sh("iptables", "-w", "-C", "DOCKER-USER", *spec, "-j", "ACCEPT")
                if rc != 0:
                    _sh("iptables", "-w", "-I", "DOCKER-USER", "1", *spec, "-j", "ACCEPT")
            rc, _ = _sh("iptables", "-w", "-C", "FORWARD", "-j", "DOCKER-USER")
            if rc != 0:
                _sh("iptables", "-w", "-I", "FORWARD", "1", "-j", "DOCKER-USER")
        log("PPP : IPCP ouvert, %s au telephone ; %s %s, la data passe par %s (%s)"
            % (PPP_PEER, PPP_IF, "dans l espace %s" % self.ns if self.ns else "sur l hote",
               tun, "GPRS, Calypso + ROM TI" if tun == GPRS_TUN else "4G, srsUE"))
        return True

    def _attendre_radio(self):
        """Le PDP du modem GPRS arrive 30 a 120 s apres ATD*99 (bascule du
        Calypso, attach, PDP) : on branche la sortie des qu il est la."""
        t0 = time.time()
        while not self.fini and self.tun is not None and time.time() - t0 < 300:
            if self._radio_brancher():
                return
            time.sleep(2)
        if not self.fini and self.tun is not None:
            log("PPP : toujours pas de PDP GPRS apres 300 s (scripts/calypso-mode.sh etat)")

    def _vers_tun(self, pkt):
        if self.tun is None:
            return
        try:
            os.write(self.tun, pkt)
            self.octets[0] += len(pkt)
        except OSError:
            pass

    def _depuis_tun(self):
        fd = self.tun
        while not self.fini and fd is not None:
            try:
                pkt = os.read(fd, 2048)
            except OSError:
                return
            if not pkt:
                return
            self.octets[1] += len(pkt)
            self.envoyer(self.IP, pkt)

    def _tun_demonter(self):
        fd, self.tun = self.tun, None
        if fd is None:
            return
        ip = ["ip", "-n", self.ns] if self.ns else ["ip"]
        _sh(*ip, "link", "del", PPP_IF)
        if not self.ns:
            _sh("ip", "rule", "del", "from", PPP_PEER, "lookup", "45")
        try:
            os.close(fd)
```

```
        except OSError:
            pass

    def fin(self, motif):
        if self.fini:
            return
        self.fini = True
        log("PPP : fin (%s) - %d octets montes, %d descendus" % (motif, self.octets[0], self.octets[1]))
        self._tun_demonter()
        self.modem.ppp_fini()


# ── LE MODEM AT ──────────────────────────────────────────────────────
class AtHandler(socketserver.StreamRequestHandler):
    """Un modem par connexion : oFono en ouvre une seule."""

    role = "commande"             # « commande » (hvc0) ou « data » (hvc1, voir DATA)

    def setup(self):
        super().setup()
        global CURRENT, DATA
        if self.role == "data":
            DATA = self
        else:
            CURRENT = self        # oFono n ouvre qu une connexion : c est celle-ci
        self.echo = True          # un vrai modem demarre en echo
        self.cmee = 1
        self.creg_mode = 0
        self.cops_format = 2          # 0 = nom long, 1 = nom court, 2 = MCC/MNC
        self.calls = []               # les appels en cours, pour +CLCC
        self.sms_target = None
        # Comment ce systeme veut qu on lui annonce les SMS. mt=2 : le message
        # pousse tel quel ; mt=1 : rangé, et on ne signale que l index. On
        # part sur 2, et AT+CNMI= vient dire la verite.
        self.cnmi_mt = 2
        self.cmgf = 0             # 0 = PDU (ce qu utilise ModemManager), 1 = texte
        # La 4G et le CSFB (voir la classe Lte). Trois modes d URC, un par
        # enregistrement (CS, GPRS, EPS) ; ws46 = les technologies que le
        # systeme AUTORISE (27.007 § 5.9 : 12 GSM seule, 28 E-UTRAN seule,
        # 31 les deux) - c est le reglage « reseau prefere » de Phosh ;
        # _cs_hold tient la 2G le temps du service CS, _retour est le
        # minuteur qui nous remonte sur la 4G ; _rat_annonce, la derniere
        # technologie annoncee, pour ne signaler que les changements.
        self.cgreg_mode = 0
        self.cereg_mode = 0
        self.ws46 = 31
        self._cs_hold = False
        self._retour = None
        self._rat_annonce = None
        self.data_cid = 0           # le contexte PDP « connecte » par +GTRNDIS (0 = aucun)
        self.ppp = None             # la session PPP en cours (ATD*99), voir la classe Ppp
        self._urc_attente = []      # les URC retenus pendant la data (voir out)
        self._urc_lock = threading.Lock()
        self.port_commande = True   # False entre NO CARRIER et la commande AT suivante
        log("oFono s est connecte depuis %s (port de %s)" % (self.client_address, self.role))
        if self.role != "data":
            threading.Thread(target=self._veille, daemon=True).start()

    def finish(self):
        """Le lien est tombe : ce modem n est plus « le » modem.

        [2026-09-07] CURRENT RESTAIT SUR UN MORT. Il etait pose dans setup() et
        jamais repris : apres une deconnexion, on_passerelle_sms() croyait
        avoir un correspondant, deliver_sms() ecrivait dans une douille fermee
        (out() avale l OSError) et journalisait « SMS remis » - un message
        annonce comme remis a personne. Meme piege pour les RING d AmiEcoute.
```

```
    La boite, elle, garde le message : il ressortira au prochain AT+CMGL.
    """
    global CURRENT, DATA
    if CURRENT is self:
        CURRENT = None
    if DATA is self:
        DATA = None
    if self.ppp is not None:
        self.ppp.fin("lien AT ferme")
    # [2026-09-08] LES APPELS MEURENT AVEC LE LIEN. Un appel arrive sur ce
    # gestionnaire, la VM redemarre : le lien tombe, un nouveau
    # gestionnaire nait, et l AMI ne parle qu a CURRENT - le raccroche
    # n arrivait donc jamais ici. La boucle _sonne tournait pour toujours
    # (RING toutes les 3 s dans le journal, vers une douille fermee) et le
    # correspondant restait pendu. On raccroche cote Asterisk et on vide.
    for c in list(self.calls):
        log("lien ferme : appel %s abandonne, raccroche cote Asterisk"
            % (c.get("channel") or c.get("num") or "?"))
        ami_hangup(c.get("channel"))
    self.calls.clear()
    t, self._retour = self._retour, None
    if t:
        t.cancel()
    try:
        super().finish()
    except OSError:
        pass

def ppp_fini(self):
    """La session PPP est finie : le port redevient un port AT, et on le
    dit comme un modem (NO CARRIER) - ModemManager relance alors sa
    sonde et le porteur passe « deconnecte »."""
    self.ppp = None
    self.data_cid = 0
    # NO CARRIER part tel quel : c est lui qui rend le port. Ensuite, tant
    # que le telephone n a pas envoye sa premiere commande AT, le port est
    # encore a pppd qui s arrete : ce qu on emettrait partirait dans le
    # vide (les +CREG du retour 4G, un +CMTI). On retient (voir out).
    self.port_commande = False
    atlog(">", "NO CARRIER")
    try:
        self.wfile.write(b"\r\nNO CARRIER\r\n")
        self.wfile.flush()
    except OSError:
        pass

def _tag(self, sens):
    return sens if self.role != "data" else sens + "d"      # « >d » : port de donnees

def out(self, text):
    # [2026-09-08] UN SEUL PORT, DONC PAS D URC PENDANT LE PPP. Le modem du
    # banc n a qu un port serie : une fois ATD*99 passe, tout ce qu on y
    # ecrit entre dans le flux PPP, et pppd jette RING, +CLIP, +CMTI comme
    # du bruit. Le telephone ne voyait donc ni appel ni SMS entrant des
    # que la data etait montee. Ce qui doit etre annonce pendant la data
    # attend ici, et part des que le port est revenu en mode commande
    # (voir _urc_liberer, appele apres la premiere commande AT qui suit).
    # csfb_debut() coupe la data pour cela : sur un vrai reseau aussi, le
    # CSFB suspend la data le temps du service CS.
    if self.ppp is not None or not self.port_commande:
        with self._urc_lock:
            if text not in self._urc_attente:
                self._urc_attente.append(text)
        atlog(self._tag("~"), text + "   (retenu : port en mode donnees)")
        return
    atlog(self._tag(">"), text)
```

```
    try:
        self.wfile.write(("\r\n%s\r\n" % text).encode())
        self.wfile.flush()
    except OSError:
        pass

def _urc_liberer(self):
    """Le port est en mode commande : on emet ce qui attendait."""
    with self._urc_lock:
        attente, self._urc_attente = self._urc_attente, []
    for text in attente:
        self.out(text)

# -- la technologie d acces : 4G au repos, 2G le temps d un service CS
def _modem(self):
    """L etat du telephone vit sur le port de commande : le port de
    donnees s y reporte."""
    if self.role == "data" and CURRENT is not None:
        return CURRENT
    return self

def cs_actif(self):
    m = self._modem()
    return bool(m.calls) or m.sms_target is not None or m._cs_hold

def appel_en_cours(self):
    return bool(self._modem().calls)

def rat(self):
    """« lte » ou « gsm » : ou le telephone est pose en ce moment."""
    if self.role == "data" and CURRENT is not None:
        return CURRENT.rat()
    if self.ws46 == 12:                  # le systeme a demande GSM seule
        return "gsm"
    if not LTE.state()["ok"]:
        return "gsm"
    if self.cs_actif():
        return "gsm"                     # CSFB : on est descendu
    return "lte"

def _reg(self, tag):
    """(stat, lac ou tac, ci ou eci, act) pour +CREG, +CGREG ou +CEREG
    dans la technologie courante. act None = pas de position a donner."""
    st = BANC.state()
    lte = LTE.state()
    if self.rat() == "lte":
        # Rattachement combine (EPS + IMSI attach par SGs) : le CS est
        # enregistre si la 2G est la, et tout se dit avec TAC/ECI, AcT 7.
        stat_cs = 1 if st["up"] else (2 if lte["ok"] else 0)
        stat = stat_cs if tag == "+CREG" else 1
        return stat, lte["tac"], lte["eci"], 7
    reg = 1 if st["up"] else 0
    if tag == "+CEREG":
        # Sur la 2G, l EPS est suspendu (CSFB) ou absent : pas de position.
        return (2 if lte["ok"] else 0), 0, 0, None
    return reg, st["lac"], st["cid"], 0

def _ligne_reg(self, tag, mode, urc=False):
    stat, lac, ci, act = self._reg(tag)
    tete = "%s: " % tag + ("" if urc else "%d," % mode)
    if mode >= 2 and act is not None and stat in (1, 5):
        return '%s%d,"%04X","%08X",%d' % (tete, stat, lac, ci, act)
    return "%s%d" % (tete, stat)

def annonce_rat(self):
    """Signale un changement de technologie par les URC d enregistrement
```

```
    que le systeme a demandes (AT+CREG=2 etc.). L ordre compte : c est la
    DERNIERE annonce qui donne l AcT que ModemManager retient, on finit
    donc par celle qui porte la technologie du moment."""
    if self.role == "data":
        return                          # les URC vont sur le port de commande
    rat = self.rat()
    prev, self._rat_annonce = self._rat_annonce, rat
    if prev == rat:
        return
    if prev is not None:
        log("technologie : %s -> %s" % ("4G" if prev == "lte" else "2G",
                                        "4G" if rat == "lte" else "2G"))
    ordre = ("+CEREG", "+CGREG", "+CREG") if rat == "gsm" else ("+CREG", "+CGREG", "+CEREG")
    modes = {"+CREG": self.creg_mode, "+CGREG": self.cgreg_mode, "+CEREG": self.cereg_mode}
    for tag in ordre:
        if modes[tag] >= 1:
            self.out(self._ligne_reg(tag, modes[tag], urc=True))

def _veille(self):
    """La 4G peut se lever ou tomber pendant que le telephone ne fait
    rien : on regarde toutes les deux secondes, et on verifie une fois que
    la 2G annoncee par le SIB7 est bien celle du banc."""
    while CURRENT is self:
        try:
            st = BANC.state()
            if st["up"]:
                LTE.verifier_geran(st["arfcn"])
            self.annonce_rat()
        except Exception as e:      # jamais au prix du modem lui-meme
            log("veille 4G : %s" % e)
        time.sleep(2)

def csfb_debut(self, motif, couper_data=True):
    """Un service CS commence : on descend sur la 2G, tout de suite, et on
    annule un retour 4G qui serait en route. couper_data : la session PPP
    est fermee (un appel : la 2G n a pas de data pendant la voix ; un SMS
    en port unique : il faut rendre le port aux URC)."""
    t, self._retour = self._retour, None
    if t:
        t.cancel()
    self._cs_hold = True
    if GPRS_DATA and calypso_mode() == "data":
        # [2026-10-15] Classe B : la data GPRS du Calypso laisse la place a
        # la voix. Le mobile revient (meme ligne que c54x_exe/run.sh) et on
        # attend qu il soit en service avant de laisser partir l appel.
        log("calypso : %s - le Calypso repasse au mobile (voix)" % motif)
        for m in (self, DATA):
            if m is not None and m.ppp is not None:
                m.ppp.terminer("voix : le Calypso quitte le GPRS")
        calypso("voix", attendre=True)
    if self._rat_annonce == "lte":
        st = BANC.state()
        log("CSFB : %s - on quitte la 4G pour la 2G (ARFCN %s, LAC %d, CI %d)"
            % (motif, st["arfcn"] or "?", st["lac"], st["cid"]))
    # La data est suspendue le temps du service CS - et c est surtout le
    # seul moyen de rendre le port aux URC (voir out). NetworkManager
    # relancera ATD*99 de lui-meme une fois l UE revenu sur la 4G.
    if couper_data:
        for m in (self, DATA):
            if m is not None and m.ppp is not None:
                log("CSFB : data suspendue (fin de la session PPP) le temps du service CS")
                m.ppp.terminer("CSFB : data suspendue le temps du service CS")
    self.annonce_rat()

def csfb_fin_apres(self, delai=2.0):
    """Le service CS est fini : dans <delai> secondes, s il n en a pas
```

```
    commence un autre, on remonte sur la 4G - le temps qu un vrai UE met
    a reselectionner sa cellule LTE une fois l appel raccroche."""
    t, self._retour = self._retour, None
    if t:
        t.cancel()
    self._retour = threading.Timer(delai, self._retour_lte)
    self._retour.daemon = True
    self._retour.start()

def _retour_lte(self):
    self._retour = None
    if self.calls or self.sms_target is not None:
        return                          # un autre service CS a pris le relais
    self._cs_hold = False
    if self._rat_annonce == "gsm" and LTE.state()["ok"] and self.ws46 != 12:
        log("CSFB : service CS termine, retour sur la 4G")
    self.annonce_rat()

def ok(self):
    self.out("OK")

def err(self, code=100):
    self.out("+CME ERROR: %d" % code if self.cmee else "ERROR")

def handle(self):
    buf = b""
    while True:
        try:
            data = self.rfile.read1(4096)
        except (OSError, ValueError):
            break
        if not data:
            break
        if self.ppp is not None:
            # Mode donnees : tout va au PPP, rien n est repete en echo.
            self.ppp.feed(data)
            buf = b""
            continue
        if self.echo:
            try:
                self.wfile.write(data)
                self.wfile.flush()
            except OSError:
                pass
        buf += data
        while True:
            m = re.search(rb"[\r\n\x1a]", buf)
            if not m:
                break
            line, buf = buf[:m.start()], buf[m.end():]
            text = line.decode(errors="replace").strip()
            if text:
                atlog(self._tag("<"), text)
            if text:
                self.port_commande = True     # le telephone parle AT : le port est a nous
                # Les URC retenus partent AVANT la commande : si c est
                # ATD*99, le port repasse en data juste apres, et un
                # +CEREG « retour 4G » retenu resterait enferme derriere
                # le PPP - Phosh gardait la 2G a l ecran.
                if self.ppp is None and self._urc_attente:
                    self._urc_liberer()
            if self.sms_target is not None and m.group() == b"\x1a":
                self.sms(text)
            elif text:
                self.command(text)
    log("oFono s est deconnecte")
```

```
# -- un SMS venu du banc, remis au modem
# -- les appels entrants, pousses par AmiEcoute
def est_notre_jambe(self, chan):
    """Ce canal est-il une jambe d un appel que NOUS avons lance ?

    [2026-09-07] LE TELEPHONE SE SONNAIT LUI-MEME, ET RACCROCHAIT SEUL.
    ami_originate() lance « Local/<num>@internal » d un cote et, de
    l autre, fait composer NOTRE PROPRE numero (Exten: MSISDN) - c est
    ainsi que le banc met le correspondant en relation avec l abonne. Cette
    seconde jambe produit donc, pour chacun de nos appels sortants :
        DialBegin  Context: internal  DestExten: 100101
    soit exactement la signature d un appel entrant (« internal » fait
    partie d IN_CTX depuis qu un appel Linphone doit pouvoir nous joindre).
    Le filtre « DestExten == notre MSISDN » ne separait donc rien : la
    remarque en tete d IN_CTX - « la seconde jambe compose le numero
    DISTANT, jamais le notre » - est fausse pour CE code.

    Consequence, reproduite sur le banc le 2026-09-07 (Originate vers 600,
    evenements AMI horodates) : des le ATD, entrant() ajoutait un DEUXIEME
    appel et envoyait RING - le telephone sonnait pour l appel qu il venait
    de passer ; +CLCC en annoncait deux ; et a la fin de la jambe,
    fin_appel() emettait NO CARRIER. ModemManager terminait alors l appel :
    l ecran d appel de Phosh se fermait tout seul, quelques dixiemes de
    seconde apres la numerotation. « Ca me disco quand je passe un appel ».
    On reconnait donc nos jambes au canal QUI COMPOSE (le Local que nous
    avons demande), et non au contexte - un vrai appel entrant, lui, arrive
    toujours par un autre canal.
    """
    if not chan:
        return False
    for c in self.calls:
        ref = c.get("origine")
        if ref and chan.startswith(ref):
            return True
    return False

def entrant(self, num, chan):
    """Le banc nous appelle : on sonne jusqu a ce que ca cesse."""
    if any(c.get("channel") == chan for c in self.calls):
        return
    c = {"num": num, "state": "incoming", "channel": chan}
    # Le paging arrive par la 4G (SGs), la sonnerie se joue sur la 2G :
    # on descend AVANT le premier RING. Avec un port de donnees a part, la
    # data reste : refuser ATD*99 pendant la sonnerie faisait passer
    # NetworkManager en repli (4 echecs, puis 5 min sans reessayer).
    self.csfb_debut("appel entrant de %s" % (num or "inconnu"), couper_data=(DATA is None))
    self.calls.append(c)
    log("appel entrant de %s (%s)" % (num or "inconnu", chan))
    threading.Thread(target=self._sonne, args=(c,), daemon=True).start()

def _sonne(self, c):
    """RING toutes les trois secondes, et +CLIP juste derriere pour donner
    le numero - c est ce que ModemManager attend pour presenter l appel
    (AT+CLIP=1 fait partie des reglages qu on accepte)."""
    while c in self.calls and c["state"] == "incoming" and CURRENT is self:
        self.out("RING")
        if c["num"]:
            _n, _t = clip(c["num"])
            self.out('+CLIP: "%s",%d,,,,0' % (_n, _t))
        time.sleep(3)

def fin_appel(self, chan):
    """Asterisk a raccroche ce canal. Sans ce chemin, l ecran d appel de
    Phosh restait affiche quand le correspondant raccrochait."""
    for c in list(self.calls):
```

```
            ref = c.get("channel")
            # Nos appels sortants gardent « Local/600@internal », qu Asterisk
            # decline en « Local/600@internal-0000001a;1 » : on compare donc
            # par le debut. Les entrants, eux, portent le nom exact.
            if ref and chan.startswith(ref):
                self.calls.remove(c)
                log("appel termine (%s)" % chan)
                self.out("NO CARRIER")
        if not self.calls:
            self.csfb_fin_apres()

    def deliver_sms(self, sender, text):
        """Remet un SMS entrant DE LA FACON QUE LE SYSTEME A DEMANDEE.

        On range toujours le message : meme en mt=2 le systeme peut vouloir le
        relire (AT+CMGL au demarrage). Ensuite seulement on choisit l URC.
        """
        pdu = pdu_deliver(sender, text)
        idx = inbox_ranger(sender, text, pdu)
        # Un SMS est un service CS sur ce banc (il vient du MSC) : on descend
        # sur la 2G le temps de le recevoir, on remonte trois secondes apres.
        # Avec un port de donnees a part, la data reste montee : les URC ont
        # leur port. En port unique, il faut couper pour que +CMTI passe.
        self.csfb_debut("SMS entrant de %s" % (sender or "inconnu"), couper_data=(DATA is None))
        self.csfb_fin_apres(3.0)
        if self.cnmi_mt == 2:
            tpdu_len = len(pdu) // 2 - 1      # sans l octet de SMSC en tete
            self.out("+CMT: ,%d\r\n%s" % (tpdu_len, pdu))
            log("SMS remis directement (+CMT) : %s" % text)
        else:
            self.out('+CMTI: "SM",%d' % idx)
            log("SMS range en SM[%d], index signale (+CMTI) : %s" % (idx, text))

    # -- les trois commandes qui vont avec la boite
    def _sms_ligne(self, m, entete):
        """Une entree de CMGR/CMGL, dans le mode courant."""
        if self.cmgf:                         # mode texte
            stat = "REC READ" if m["lu"] else "REC UNREAD"
            return '%s"%s","%s",,"%s"\r\n%s' % (entete, stat, m["sender"],
                                                m["ts"], m["text"])
        stat = 1 if m["lu"] else 0            # mode PDU : 0 non lu, 1 lu
        return "%s%d,,%d\r\n%s" % (entete, stat, len(m["pdu"]) // 2 - 1,
                                   m["pdu"])

    # -- l envoi d un SMS par le banc
    def sms(self, text):
        num, self.sms_target = self.sms_target, None
        if num == "<pdu>":
            try:
                num, text = parse_submit(text.strip())
            except (ValueError, IndexError) as e:
                log("PDU illisible (%s) : %s" % (e, text[:60]))
                self.err()
                return
        self.csfb_debut("SMS vers %s" % num, couper_data=(DATA is None))
        out = MSC.cmd('subscriber msisdn %s sms sender msisdn %s send %s'
                      % (num, MSISDN, text))
        self.csfb_fin_apres(3.0)
        if out is None:
            log("SMS non envoye : osmo-msc injoignable")
            self.err()
            return
        log("SMS a %s : %s" % (num, text))
        self.out("+CMGS: 1")
        self.ok()
```

```
def command(self, line):
    cmd = line.upper()
    if not cmd.startswith("AT"):
        return
    ub = cmd[2:]
    body = line[2:]
    st = BANC.state()

    if ub in ("", "&F", "Z", "V1", "Q0", "S0=0", "+CMEE=1", "+CMEE=2",
              "+CLIP=1", "+COLP=0", "+CSSN=0,0", "+CMOD=0", "+CSNS=0",
              "+CSCS=\"IRA\"",
              "+CSCS=\"UCS2\"", "+CGEREP=1,0", "+CVHU=0", "&C1", "&D2"):
        self.ok()
        return
    if ub == "E0":
        self.echo = False; self.ok(); return
    if ub == "E1":
        self.echo = True; self.ok(); return

    if ub in ("+CGMI", "+GMI"):
        # [2026-09-07] « Fibocom », ET POURQUOI. ModemManager choisit son
        # greffon d apres ce nom. Le greffon generique ne sait faire la
        # data que par PPP sur le port AT - et le noyau de la VM n a pas
        # de PPP. Le greffon Fibocom, lui, quand le modem a un PORT RESEAU
        # a cote du port AT et repond a AT+GTRNDIS=?, monte un porteur
        # ECM : +GTRNDIS=1,<cid> pour connecter, adresse par DHCP sur le
        # port reseau. C est ce que fait ce banc : la seconde carte reseau
        # de QEMU (tools/osmo-pmos-data.sh) est ce port, et elle debouche
        # dans l espace reseau de srsUE - donc sur la radio 4G.
        self.out(VENDOR); self.ok(); return
    if ub in ("+CGMM", "+GMM"):
        self.out("osmo-banc"); self.ok(); return
    if ub in ("+CGMR", "+GMR"):
        self.out("1.0"); self.ok(); return
    if ub in ("+CGSN", "+GSN"):
        self.out("350000000000000"); self.ok(); return
    if ub == "+CIMI":
        self.out(IMSI); self.ok(); return
    if ub == "+CPIN?":
        self.out("+CPIN: READY"); self.ok(); return
    if ub.startswith("+CNUM"):
        self.out('+CNUM: "","%s",129' % MSISDN); self.ok(); return

    if ub.startswith("+CFUN"):
        if ub.endswith("?"):
            self.out("+CFUN: %d" % (1 if st["up"] else 4))
        self.ok(); return

    if ub.startswith(("+CREG", "+CGREG", "+CEREG")):
        # [2026-09-07] LES TROIS ENREGISTREMENTS, ET LA 4G. +CREG est le
        # circuit (voix, SMS), +CGREG le paquet 2G/3G, +CEREG l EPS - la
        # LTE. Ce modem repondait ERROR a +CEREG pour rester un modem 2G ;
        # depuis que le banc a une 4G (srsRAN, voir la classe Lte), il
        # dit la verite : enregistre sur l EPS quand srsUE est attache, et
        # chaque reponse porte la technologie du moment (dernier champ :
        # 0 = GSM, 7 = E-UTRAN). Pendant un appel ou un SMS, la reponse
        # bascule sur la 2G (CSFB) et l EPS passe « en recherche ».
        # [2026-09-06] LA FORME « TEST » N EST PAS UN REGLAGE. Le driver
        # atmodem sonde d abord AT+CREG=? et attend la LISTE des modes
        # supportes ; un simple OK lui fait conclure que le modem ne sait
        # pas s enregistrer - oFono journalisait « Unable to initialize
        # Network Registration » et n envoyait plus jamais AT+CREG?, donc
        # pas d interface NetworkRegistration du tout. Meme piege pour
        # +CGREG (la data), d ou « GPRS not supported on this device ».
        tag = ub.split("=")[0].rstrip("?")
        if ub.endswith("=?"):
```

```
            self.out("%s: (0-2)" % tag); self.ok(); return
        if "=" in ub:
            try:
                mode = int(ub.split("=")[1][0])
            except (ValueError, IndexError):
                mode = 0
            if tag == "+CREG":
                self.creg_mode = mode
            elif tag == "+CGREG":
                self.cgreg_mode = mode
            else:
                self.cereg_mode = mode
            self.ok(); return
        mode = {"+CREG": self.creg_mode, "+CGREG": self.cgreg_mode,
                "+CEREG": self.cereg_mode}[tag]
        self.out(self._ligne_reg(tag, mode))
        self.ok(); return

    if ub.startswith("+COPS"):
        # oFono interroge l operateur DEUX FOIS, dans deux formats : il pose
        # AT+COPS=3,2 puis AT+COPS? pour le code numerique (MCC/MNC), et
        # AT+COPS=3,0 puis AT+COPS? pour le nom. Repondre toujours en
        # numerique laissait le nom vide et MCC/MNC a None cote oFono - le
        # format demande doit etre retenu.
        # Le dernier champ est la technologie : 7 sur la 4G, 0 sur la 2G.
        rat = self.rat()
        act = 7 if rat == "lte" else 0
        nom = LTE_NAME if rat == "lte" else st["name"]
        if ub.endswith("=?"):
            # LA RECHERCHE DE RESEAUX (« Selectionner un reseau » dans
            # Phosh). Un vrai scan voit une cellule des qu elle EMET, pas
            # seulement quand on y est attache : la 4G est donc listee des
            # que l eNB est en ligne. Elle porte un nom distinct de la 2G
            # (LTE_NAME) : deux entrees « Osmocom » n en feraient qu une
            # a l ecran. stat : 2 = courant, 1 = disponible.
            lte = LTE.state()
            liste = []
            if lte["enb"]:
                liste.append('(%d,"%s","%s","%s%s",7)'
                             % (2 if rat == "lte" else 1, LTE_NAME, LTE_NAME, lte["mcc"], lte["mnc"]))
            if st["up"]:
                liste.append('(%d,"%s","%s","%s%s",0)'
                             % (2 if rat == "gsm" else 1, st["name"], st["name"], st["mcc"], st["mnc"]))
            self.out('+COPS: %s,,(0,1,3,4),(0,1,2)' % ",".join(liste))
            self.ok(); return
        if ub.endswith("?"):
            if self.cops_format == 2:
                self.out('+COPS: 0,2,"%s%s",%d' % (st["mcc"], st["mnc"], act))
            else:
                self.out('+COPS: 0,%d,"%s",%d' % (self.cops_format, nom, act))
            self.ok(); return
        m = re.match(r'\+COPS=3,(\d)', ub)
        if m:
            self.cops_format = int(m.group(1))
            self.ok(); return
        # La SELECTION : AT+COPS=0 (automatique) ou AT+COPS=1,<fmt>,<oper>
        # [,<act>]. Choisir l entree 4G de la liste, c est camper en LTE ;
        # l entree 2G, en GSM ; l automatique rend les deux.
        #
        # [2026-09-07] C EST LE SEUL LEVIER QU A L UTILISATEUR. Le reglage
        # « type de reseau » de Phosh passe par ModemManager, qui REFUSE de
        # le poser sur un modem a commandes AT : « Setting allowed modes not
        # supported » (MM 1.25.95 - il lit AT+WS46=? pour la liste, mais
        # n emet jamais AT+WS46=<n>). Reste la selection manuelle d un
        # reseau, et MM la fait toujours en NUMERIQUE, SANS technologie :
        # « AT+COPS=1,2,"00101" ». Or nos deux radios portent le meme
```

```
        # MCC/MNC - le numero ne dit donc pas laquelle on veut.
        # On tranche par l usage : l automatique (+COPS=0), c est deja
        # « la 4G tant qu elle est la » ; demander LA MAIN, ici, ne peut
        # vouloir dire qu une chose - rester sur la 2G. Un nom (format 0
        # ou 1) ou une technologie explicite, eux, sont pris au mot.
        m = re.match(r'\+COPS=(\d)(?:,(\d),"?([^",]*)"?(?:,(\d))?)?', ub)
        if m:
            mode, fmt, oper, actsel = (int(m.group(1)), m.group(2),
                                       m.group(3) or "", m.group(4))
            if mode == 0:
                self.ws46 = 31
            elif mode == 1:
                if actsel is not None:
                    self.ws46 = 28 if actsel == "7" else 12
                elif fmt in ("0", "1") and oper:
                    self.ws46 = 28 if oper.strip().upper() == LTE_NAME.upper() else 12
                else:
                    self.ws46 = 12
           log("selection de reseau : %s" % {31: "automatique (2G + 4G)", 28: "la 4G", 12: "la 2G"}[self.ws46])
            self.annonce_rat()
        self.ok(); return

    if ub.startswith("+WS46"):
        # 27.007 § 5.9 : la technologie AUTORISEE par le systeme - c est
        # le reglage « type de reseau » de Phosh (2G / 4G / automatique).
        # 12 = GSM seule, 28 = E-UTRAN seule, 31 = GERAN + E-UTRAN.
        if ub.endswith("=?"):
            self.out("+WS46: (12,28,31)"); self.ok(); return
        if ub.endswith("?"):
            self.out("+WS46: %d" % self.ws46); self.ok(); return
        try:
            v = int(re.sub(r"\D", "", ub.split("=", 1)[1]) or "31")
        except (ValueError, IndexError):
            self.err(); return
        if v not in (12, 28, 31):
            self.err(); return
        self.ws46 = v
        log("technologies autorisees : %s" % {12: "2G seule", 28: "4G seule", 31: "2G + 4G"}[v])
        self.annonce_rat()
        self.ok(); return

    if ub.startswith("+CSQ"):
        # Banc en marche = signal franc ; banc arrete = pas de signal.
        # La 4G virtuelle est « plus forte » que la 2G : ca se voit.
        rat = self.rat()
        self.out("+CSQ: %d,99" % (28 if rat == "lte" else 24 if st["up"] else 99))
        self.ok(); return

    if ub.startswith("+CESQ"):
        # La qualite etendue (27.007 § 8.69) : rxlev/ber pour la 2G,
        # rsrq/rsrp pour la 4G, 255 = sans objet. Sur la 4G : rsrq 20
        # (-9,5 dB), rsrp 70 (-71 dBm) ; sur la 2G : rxlev 24.
        if ub.endswith("=?"):
            self.out("+CESQ: (0-63,99),(0-7,99),(0-96,255),(0-49,255),(0-34,255),(0-97,255)")
            self.ok(); return
        if self.rat() == "lte":
            self.out("+CESQ: 99,99,255,255,20,70")
        else:
            self.out("+CESQ: %d,99,255,255,255,255" % (24 if st["up"] else 99))
        self.ok(); return

    if ub.startswith("+GTRNDIS"):
        # LE PORTEUR ECM DE FIBOCOM (voir +CGMI). ModemManager sonde
        # AT+GTRNDIS=? a l initialisation ; si on repond, et qu il a un
        # port reseau groupe avec ce port AT, la data ne passe plus par
        # PPP : +GTRNDIS=1,<cid> « connecte », +GTRNDIS=0,<cid> coupe,
```

```
    # +GTRNDIS? donne l etat, et l adresse vient par DHCP sur le port
    # reseau (dnsmasq dans l espace de srsUE, tools/osmo-pmos-data.sh).
    if ub.endswith("=?"):
        self.out("+GTRNDIS: (0,1),(1-11)"); self.ok(); return
    if ub.endswith("?"):
        self.out("+GTRNDIS: %d,%d" % (1 if self.data_cid else 0, self.data_cid or 1))
        self.ok(); return
    m = re.match(r'\+GTRNDIS=(\d)(?:,(\d+))?', ub)
    if not m:
        self.err(); return
    cid = int(m.group(2) or 1)
    if m.group(1) == "1":
        if not LTE.state()["ok"] and not st["up"]:
            log("data : +GTRNDIS refuse, aucun reseau")
            self.err(30)                # 30 = pas de service reseau
            return
        self.data_cid = cid
        log("data : porteur ECM connecte (cid %d, APN %s) - l adresse vient par DHCP "
            "sur le port reseau, la data passe par %s"
            % (cid, LTE.state()["apn"], "la 4G (srsUE)" if self.rat() == "lte" else "la 2G"))
    else:
        self.data_cid = 0
        log("data : porteur ECM coupe (cid %d)" % cid)
    self.ok(); return

if ub == "+CGDCONT?":
    # Le contexte PDP : l APN est celui de srsEPC (epc.conf), la donnee
    # du telephone est sur la 4G - meme si, sans PPP, le systeme ne
    # peut pas l activer par ce port (voir ATD*99).
    self.out('+CGDCONT: 1,"IP","%s","0.0.0.0",0,0' % LTE.state()["apn"])
    self.ok(); return
if ub == "+CGATT?":
    self.out("+CGATT: %d" % (1 if (LTE.state()["ok"] or st["up"]) else 0))
    self.ok(); return
if ub == "+CGACT?":
    self.out("+CGACT: 1,%d" % (1 if (self.data_cid or self.ppp) else 0)); self.ok(); return
if ub == "+CGACT=?":
    self.out("+CGACT: (0,1)"); self.ok(); return
if ub.startswith("+CEMODE"):
    # Le mode d exploitation EPS (24.301 § 4.3) : 1 = « CS/PS mode 1,
    # voice centric » - le telephone qui redescend en 2G pour parler.
    # C est exactement ce banc.
    if ub.endswith("=?"):
        self.out("+CEMODE: (0-3)")
    elif ub.endswith("?"):
        self.out("+CEMODE: 1")
    self.ok(); return

# ── CE QU oFONO DEMANDE A L INITIALISATION ──────────────────────
# [2026-09-06] Releve sur le banc : des qu on passe Powered=true, oFono
# deroule tout l inventaire d un vrai modem - jeux de caracteres,
# services SMS, et surtout une longue serie d AT+CRSM (la lecture des
# fichiers de la carte SIM). Repondre ERROR a tout ca fait abandonner
# oFono ; on repond donc comme un modem dont la SIM n a pas ces
# fichiers, ce qui est LA VERITE ici : il n y a pas de carte, l identite
# vient du banc (AT+CIMI) et non d une SIM physique.
if ub == "+CSCS=?":
    self.out('+CSCS: ("IRA","GSM","UCS2")'); self.ok(); return
if ub.startswith("+CSMS"):
    # 1,1,1 : SMS emis, recus et diffuses pris en charge.
    self.out("+CSMS: 1,1,1"); self.ok(); return
if ub.startswith("+CSIM"):
    # [2026-09-07] REFUSER FRANCHEMENT L ACCES APDU. ModemManager sonde
    # la carte avec AT+CSIM (des commandes APDU brutes : selection de
    # fichiers, lecture de l IMSI...). Repondre « OK » sans donnee le
    # laisse attendre une reponse qui ne vient jamais et il n enregistre
```

```
    # aucun modem. En annoncant que la commande n est pas geree, il se
    # rabat sur les commandes ordinaires (+CIMI, +CPIN?), auxquelles on
    # sait repondre - il n y a pas de vraie carte ici, l identite vient
    # du banc.
    self.err(4)          # 4 = operation non supportee
    return
if ub.startswith("+CRSM"):
    # 148,4 = SW1 0x94 / SW2 0x04, « fichier introuvable » : la reponse
    # normale d une SIM qui ne porte pas le fichier demande.
    self.out("+CRSM: 148,4"); self.ok(); return
# Les lectures dont oFono a besoin pour monter MessageManager et la
# data : refusees, elles laissaient ces interfaces absentes.
# [2026-09-06] LA COMMANDE QUI DECIDE DE TOUT : AT+GCAP. C est par elle
# qu oFono demande au modem ce qu il sait faire ; sans « +CGSM » dans la
# reponse, il n active pas la couche reseau et n envoie meme JAMAIS
# AT+CREG - l interface NetworkRegistration reste alors absente, ce
# qu on a observe tel quel sur le banc (aucun CREG dans le journal).
if ub in ("+GCAP", "+CGCAP"):
    self.out("+GCAP: +CGSM,+FCLASS,+DS,+ES"); self.ok(); return

# ── LA BOITE DE RECEPTION, VUE EN AT ──────────────────────────
# [2026-09-07] LA FORME TEST D ABORD. « AT+CMGF=? » commence par
# « +CMGF= » : le prefixe ci-dessous l attrapait, repondait OK tout nu
# et laissait la vraie reponse (plus bas dans cette fonction) en code
# mort - releve dans la trace AT, « < AT+CMGF=? » suivi d un « > OK ».
# Meme piege pour AT+CNMI=?, qui en prime remettait cnmi_mt a 2.
if ub == "+CMGF=?":
    self.out("+CMGF: (0,1)"); self.ok(); return
if ub == "+CNMI=?":
    self.out("+CNMI: (0-2),(0-3),(0,2),(0,1),(0,1)"); self.ok(); return
if ub.startswith("+CMGF="):
    # Retenir le mode : CMGR et CMGL ne repondent pas pareil en texte
    # et en PDU, et repondre dans le mauvais mode fait jeter le message.
    self.cmgf = 1 if ub[6:].strip().startswith("1") else 0
    self.ok(); return
if ub == "+CMGF?":
    self.out("+CMGF: %d" % self.cmgf); self.ok(); return

if ub.startswith("+CNMI="):
    # LA COMMANDE QUI DECIDE DE TOUT POUR LES ENTRANTS. Le deuxieme
    # champ (mt) dit si le systeme veut le message pousse (2) ou
    # seulement son index (1). ModemManager demande 1.
    champs = ub[6:].split(",")
    try:
        self.cnmi_mt = int(champs[1]) if len(champs) > 1 else 2
    except ValueError:
        self.cnmi_mt = 2
    log("annonce des SMS entrants : mt=%d (%s)"
        % (self.cnmi_mt,
           "+CMT direct" if self.cnmi_mt == 2 else "+CMTI puis CMGR"))
    self.ok(); return
if ub == "+CNMI?":
    self.out("+CNMI: 2,%d,2,1,0" % self.cnmi_mt); self.ok(); return

if ub.startswith("+CNMA"):
    # L accuse de reception d un « +CMT: ». Rien a faire : le message
    # est deja parti du banc. Mais il FAUT repondre OK, sinon le
    # systeme croit la remise ratee et coupe l annonce directe.
    self.ok(); return

if ub.startswith("+CMGR="):
    try:
        idx = int(re.sub(r"\D", "", ub[6:]) or "0")
    except ValueError:
        self.err(); return
    m = inbox_get(idx)
```

```
    if m is None:
        self.err(321)            # 321 = index de memoire invalide
        return
    self.out(self._sms_ligne(m, "+CMGR: "))
    m["lu"] = True
    self.ok(); return

if ub.startswith("+CMGL"):
    if ub.endswith("=?"):
        self.out('+CMGL: (0,1,2,3,4)' if not self.cmgf else
                 '+CMGL: ("REC UNREAD","REC READ","ALL")')
        self.ok(); return
    arg = ub[6:].strip().strip('"').upper() if "=" in ub else "4"
    # 0 / « REC UNREAD » : seulement les non lus. Tout le reste : tout.
    que_non_lus = arg in ("0", "REC UNREAD")
    with INBOX_LOCK:
        liste = [m for m in INBOX if not (que_non_lus and m["lu"])]
    for m in liste:
        self.out(self._sms_ligne(m, "+CMGL: %d," % m["idx"]))
    for m in liste:
        m["lu"] = True
    self.ok(); return

if ub.startswith("+CMGD"):
    if ub.endswith("=?"):
        with INBOX_LOCK:
            idxs = ",".join(str(m["idx"]) for m in INBOX)
        self.out("+CMGD: (%s),(0-4)" % idxs); self.ok(); return
    champs = ub[6:].split(",")
    try:
        idx = int(re.sub(r"\D", "", champs[0]) or "0")
    except ValueError:
        self.err(); return
    drapeau = champs[1].strip() if len(champs) > 1 else "0"
    with INBOX_LOCK:
        if drapeau in ("1", "2", "3", "4"):
            INBOX[:] = []            # « efface tout », selon le drapeau
        else:
            INBOX[:] = [m for m in INBOX if m["idx"] != idx]
    self.ok(); return

if ub == "+CSCS?":
    self.out('+CSCS: "IRA"'); self.ok(); return
if ub.startswith("+CPMS"):
    # Trois memoires SMS : on n en tient aucune (les SMS passent par le
    # banc), on annonce donc des boites vides mais existantes.
    # Le compte doit etre VRAI : un systeme qui lit « 0 message »
    # ne va pas chercher ce qu on vient de ranger.
    with INBOX_LOCK:
        n = len(INBOX)
    if ub.endswith("=?"):
        self.out('+CPMS: ("SM"),("SM"),("SM")')
    elif ub.startswith("+CPMS="):
        self.out('+CPMS: %d,%d,%d,%d,%d,%d'
                 % (n, INBOX_MAX, n, INBOX_MAX, n, INBOX_MAX))
    else:
        self.out('+CPMS: "SM",%d,%d,"SM",%d,%d,"SM",%d,%d'
                 % (n, INBOX_MAX, n, INBOX_MAX, n, INBOX_MAX))
    self.ok(); return
if ub == "+CSMS?":
    self.out("+CSMS: 0,1,1,1"); self.ok(); return
if ub == "+CGDCONT=?":
    self.out('+CGDCONT: (1-4),"IP",,,(0-2),(0-4)'); self.ok(); return
if ub in ("+CMUT?", "+CMUT=?"):
    self.out("+CMUT: 0"); self.ok(); return
if ub in ("+CLVL?", "+CLVL=?"):
```

```
    self.out("+CLVL: 5" if ub.endswith("?") and "=" not in ub else "+CLVL: (0-9)")
    self.ok(); return

if ub in ("+CPINR", "+CUAD") or ub.startswith(("+PTTY", "+VTD", "+CCWA",
                                               "+CIND", "+CMER", "+CPBS",
                                               "+CSCB", "+CGATT", "+CPMS")):
    self.ok(); return

if ub == "+CLCC=?":
    # [2026-09-07] LA COMMANDE QUI DECIDE DU RACCROCHAGE. Releve dans
    # la trace AT de la VM : ModemManager n envoie JAMAIS AT+CLCC. Il
    # commence par demander AT+CLCC=? et, sur ERROR, conclut que ce
    # modem ne sait pas rendre sa liste d appels - il coupe alors la
    # relecture periodique et ne lui reste plus rien pour apprendre
    # qu un appel s est termine. Quand le correspondant raccrochait,
    # l ecran d appel restait donc affiche. Un vrai modem repond OK.
    self.ok(); return

if ub == "+CLCC":
    for i, c in enumerate(self.calls, start=1):
        # 27.007 : idx, direction, etat, mode, multipartie, numero,
        # type. Direction 0 = sortant, 1 = entrant. Etat 0 = actif,
        # 2 = en cours d appel, 4 = entrant qui sonne.
        sens = 1 if c["state"] == "incoming" else 0
        etat = {"active": 0, "incoming": 4}.get(c["state"], 2)
        _n, _t = clip(c["num"])
        self.out('+CLCC: %d,%d,%d,0,0,"%s",%d'
                 % (i, sens, etat, _n, _t))
    self.ok(); return

if ub.startswith("+CMGS="):
    # Deux formes : mode texte (le numero entre guillemets, pour un
    # essai a la main) et mode PDU (une longueur - c est ce qu envoie
    # oFono). Dans les deux cas la suite arrive apres l invite « > ».
    m = re.search(r'"([^"]+)"', body)
    self.sms_target = m.group(1) if m else "<pdu>"
    try:
        self.wfile.write(b"\r\n> ")
        self.wfile.flush()
    except OSError:
        pass
    return

if ub == "A":
    # [2026-09-07] DECROCHER. L appel va DEJA vers l abonne du banc -
    # la VM et le mobile sont le meme 100101, comme pour les SMS. On ne
    # peut donc pas « prendre » la ligne : on note que l appel est en
    # cours, et la voix arrive par le pont audio (gsm_audio), que la VM
    # relaie deja.
    for c in self.calls:
        if c["state"] == "incoming":
            c["state"] = "active"
            log("appel de %s decroche" % (c["num"] or "inconnu"))
            threading.Thread(target=mobile_decroche, daemon=True).start()
            self.ok(); return
    self.err(); return

if cmd.startswith("ATD"):
    num = re.sub(r"[^0-9+*#]", "", body.split(";")[0][1:])
    if not num:
        self.err(); return
    if not num.startswith("*99") and ("*" in num or "#" in num):
        # [2026-09-08] UN CODE USSD N EST PAS UN APPEL. Le composeur
        # de Phosh envoie « *#100# » en ATD : on le lancait dans
        # Asterisk, on descendait en 2G, et on essayait de faire
        # decrocher le mobile pour rien. Refuse (27.007 : operation
```

```
    # non permise) ; l USSD passe par AT+CUSD, plus bas.
    log("ATD%s : code USSD/SS, pas un appel - refuse" % num)
    self.err(3); return
if num.startswith("*99"):
    # ATD*99***<cid># : la DATA, par PPP sur ce port (voir la
    # classe Ppp). On repond CONNECT et le port passe en mode
    # donnees : NetworkManager lance pppd dans le telephone, et
    # c est nous qui tenons l autre bout. Le noyau du telephone
    # doit avoir ppp_generic/ppp_async (compiles pour lui, voir
    # tools/osmo-pmos.sh) - sans eux pppd meurt et le port
    # revient en AT tout seul (Ppp.feed reconnait un « AT »).
    if not LTE.state()["ok"] and not st["up"]:
        log("data : ATD%s refuse, aucun reseau" % num)
        self.err(30)                # 30 = pas de service reseau
        return
    # [2026-09-08] PAS DE DATA PENDANT UN SERVICE CS. Sur la 2G,
    # un mobile en appel n a pas de data (classe B) ; et ici, avec
    # UN SEUL port, accepter ATD*99 en plein appel remettait le
    # port en mode donnees : le NO CARRIER du correspondant qui
    # raccroche et le +CEREG du retour sur la 4G restaient retenus
    # derriere le PPP - ecran d appel fige, telephone bloque en
    # 2G. NetworkManager reessaie de lui-meme apres le service.
    if DATA is None and self.cs_actif():
        log("data : ATD%s refuse le temps du service CS (CSFB)" % num)
        self.out("NO CARRIER")
        return
    m = re.search(r"\*\*\*(\d+)#", num)
    self.data_cid = int(m.group(1)) if m else 1
    log("data : ATD%s -> CONNECT, PPP sur le port AT (cid %d, APN %s), par %s"
        % (num, self.data_cid, LTE.state()["apn"],
           "la 4G (srsUE)" if self.rat() == "lte" else "la 2G"))
    # [2026-10-15] CONNECT APRES LE PDP, comme un vrai modem. Repondre
    # CONNECT tout de suite laissait la session PPP du telephone sans
    # sortie pendant l attach + PDP (1 a 3 min sur le Calypso) : elle
    # abandonnait (300 s) et il fallait relancer la data a la main.
    # Ici le port de donnees attend la bascule et le PDP ; en cas
    # d echec, NO CARRIER, et NetworkManager redemande de lui-meme.
    if GPRS_DATA and self.rat() == "gsm":
        pdp = _sh("ip", "-n", GPRS_NETNS, "-4", "-o", "addr", "show", GPRS_TUN)
        if not (pdp[0] == 0 and "inet" in pdp[1]):
            log("data : ATD%s - le Calypso passe au GPRS, CONNECT a l arrivee du PDP" % num)
            if not calypso("data", attendre=True):
                log("data : pas de PDP GPRS, NO CARRIER (le telephone redemandera)")
                self.out("NO CARRIER")
                return
    self.out("CONNECT")
    self.port_commande = False
    self.ppp = Ppp(self)
    self.ppp.demarrer()
    return
# CSFB : l appel se fait sur la 2G. On descend AVANT de composer,
# comme un vrai UE quitte la LTE sur l Extended Service Request.
self.csfb_debut("appel vers %s" % num, couper_data=(DATA is None))
# « origine » = le nom de DEMANDE du canal Local (« Local/600@internal »),
# qu Asterisk decline en « ...-0000000b;1 » et « ;2 ». C est par lui
# qu on reconnait nos propres jambes dans les evenements AMI
# (cf. est_notre_jambe) ; « channel » sera remplace par le vrai nom.
# [2026-09-07] NOTE AVANT DE COMPOSER, PAS APRES. L appel etait
# ajoute a la liste au retour d ami_originate ; or les evenements
# AMI arrivent sur une AUTRE connexion (AmiEcoute) et le DialBegin
# de notre seconde jambe (Exten 100101) peut précéder ce retour :
# est_notre_jambe ne trouvait rien, entrant() sonnait le telephone
# avec son propre numero, et le NO CARRIER qui suivait fermait
# l appel qu on venait de passer. Releve dans la trace AT :
# « ATD600; » puis RING +CLIP "100101" et NO CARRIER dans la ms.
chan = "Local/%s@%s" % (num, CTX)
```

```
        appel = {"num": num, "state": "active", "channel": chan, "origine": chan}
        self.calls.append(appel)
        ok, detail = ami_originate(num)
        if not ok:
            log("appel vers %s refuse : %s" % (num, detail))
            if appel in self.calls:
                self.calls.remove(appel)
            self.csfb_fin_apres()
            self.err(); return
        log("appel vers %s etabli par Asterisk" % num)
        threading.Thread(target=mobile_decroche, daemon=True).start()
        self.ok()
        return

    if ub == "+CHLD=?":
        # 0 rejeter/liberer, 1 liberer les actifs, 1x liberer l appel x.
        self.out('+CHLD: (0,1,1x)'); self.ok(); return

    if ub in ("H", "+CHUP") or ub.startswith("+CHLD="):
        # [2026-09-07] ET C EST +CHLD QUI RACCROCHE, PAS ATH.
        # Trace de la VM postmarketOS (/tmp/osmo-at-ofono-*.log) :
        #     06:53:04.174 < AT+CHLD=11      ModemManager raccroche
        #     06:53:04.207 > RING            ... et on sonnait toujours
        #     06:53:04.824 < AT+CHUP         il en fallait une SECONDE
        # +CHLD n etait traite nulle part ici (osmo-ril-atmodem.py, lui,
        # l a toujours reconnu). Il tombait donc dans la regle generale
        # « toute AFFECTATION est acceptee sans effet » : on repondait OK -
        # ModemManager croyait avoir raccroche - et l appel continuait de
        # sonner jusqu au +CHUP suivant. D ou le double raccrochage.
        # 27.007 § 7.13 : 0 = rejeter l entrant, 1 = liberer les actifs,
        # 1x = liberer l appel x. Sans mise en attente sur ce banc, les
        # trois reviennent au meme - on raccroche.
        #
        # IL FALLAIT RACCROCHER DEUX FOIS DEPUIS pmOS (suite).
        # On vidait self.calls ici meme : quand Asterisk demolissait
        # ensuite le canal pour de bon, fin_appel() ne trouvait plus rien a
        # retirer et n emettait donc JAMAIS le « NO CARRIER ». Cote VM,
        # ModemManager n apprenait pas la fin de l appel - l ecran d appel
        # restait, et il fallait un second ATH (que le premier avait deja
        # rendu sans objet) pour que l interface se resigne.
        # C est nous qui raccrochons : c est donc a nous de l annoncer,
        # tout de suite, une fois par appel. fin_appel() reste en place
        # pour l autre sens (le correspondant raccroche le premier) et ne
        # fera pas doublon : la liste est vide quand son evenement arrive.
        for c in self.calls:
            ami_hangup(c.get("channel", ""))
        n = len(self.calls)
        self.calls.clear()
        mobile("hangup")            # l abonne, c est le mobile : lui aussi
        for _ in range(n):
            self.out("NO CARRIER")
        self.csfb_fin_apres()       # plus rien en CS : retour sur la 4G
        self.ok(); return

    if ub.startswith("+CUSD=1,"):
        # USSD, un minimum : *#100# rend le numero de l abonne (comme sur
        # le mobile osmocom-bb) ; tout autre code est « libere par le
        # reseau » (+CUSD: 2), sans erreur ni descente en 2G.
        m = re.search(r'"([^"]*)"', body)
        code = m.group(1) if m else ""
        self.ok()
        if code.replace("*", "").replace("#", "") == "100":
            self.out('+CUSD: 0,"Votre numero : %s",15' % MSISDN)
        else:
            log("USSD %s : pas de service, on libere" % code)
            self.out("+CUSD: 2")
```

```
            return
        if ub == "+CBC":
            self.out("+CBC: 0,80"); self.ok(); return       # alimente, 80 %
        if ub == "+CLCK=?":
            self.out('+CLCK: ("SC","AO","OI","AI")'); self.ok(); return

        # [2026-09-06] LA REGLE GENERALE, ET POURQUOI ELLE EST SURE. oFono
        # deroule a l initialisation une trentaine de REGLAGES (AT+CRC=1,
        # AT+CNAP=1, AT+CDIP=1, AT+CSCS="GSM"...) dont il ne lit jamais la
        # reponse au-dela du OK : refuser ceux-la faisait abandonner
        # l initialisation entiere. On accepte donc tout ce qui est une
        # AFFECTATION - un modem qui accepte un reglage qu il n applique pas ne
        # ment a personne, il n a rien a rapporter dessus. En revanche une
        # LECTURE inconnue (AT+XXX? ou AT+XXX=?) reste refusee : la, une reponse
        # inventee ferait croire a oFono a une capacite qui n existe pas.
        if "=" in ub and not ub.endswith("=?"):
            log("reglage accepte sans effet : %s" % line)
            self.ok(); return

        log("lecture non geree (refusee) : %s" % line)
        self.err()


# ── LA RECEPTION DES SMS : ESME SMPP SUR osmo-msc ──────────────────────
# Emettre un SMS se fait par le VTY (« subscriber ... sms ... send »), mais
# RECEVOIR demande que le MSC nous pousse quelque chose : le VTY ne notifie
# rien. osmo-msc expose pour cela un serveur SMPP (port 2775) auquel on se
# connecte comme ESME. Chaque SMS route vers nous arrive en deliver_sm, et on
# le presente au modem sous la forme que le monde AT attend : « +CMT: » avec le
# message en PDU, precede de l URC.
#
# Le protocole est binaire mais on n en utilise qu une poignee de messages :
# bind_transceiver, enquire_link (le battement de coeur), deliver_sm et sa
# reponse. Pas de bibliotheque : ce serait une dependance de plus sur l image
# pour trois structures.
SMPP_BIND_TRANSCEIVER = 0x00000009
SMPP_BIND_TRANSCEIVER_RESP = 0x80000009
SMPP_DELIVER_SM = 0x00000005
SMPP_DELIVER_SM_RESP = 0x80000005
SMPP_ENQUIRE_LINK = 0x00000015
SMPP_ENQUIRE_LINK_RESP = 0x80000015


def _cstr(b, i):
    """Une chaine terminee par zero, et l indice juste apres."""
    j = b.index(b"\x00", i)
    return b[i:j].decode(errors="replace"), j + 1


def gsm7_pack(text):
    """L alphabet GSM 7 bits, tasse par paquets de sept octets pour huit
    caracteres - l encodage par defaut d un SMS."""
    # [2026-09-06] Le miroir EXACT de gsm7_unpack : on empile les septets dans
    # un tampon de bits et on sort un octet des qu il y en a huit. La premiere
    # version decalait a l envers et produisait un texte illisible - verifie en
    # relisant son propre resultat, qui ne redonnait pas le message d origine.
    septets = [ord(c) & 0x7F for c in text]
    out = bytearray()
    cur = bits = 0
    for sep in septets:
        cur |= sep << bits
        bits += 7
        while bits >= 8:
            out.append(cur & 0xFF)
            cur >>= 8
            bits -= 8
```

```
    if bits:
        out.append(cur & 0xFF)
    return bytes(out), len(septets)


# ── LA BOITE DE RECEPTION ──────────────────────────────────────────
# [2026-09-07] POURQUOI IL FAUT STOCKER LES SMS. Le modem ne decide pas seul
# comment il annonce un SMS entrant : c est le systeme qui le lui dit par
# AT+CNMI=<mode>,<mt>,... Deux mondes s y opposent :
#   mt=2 : « pousse-moi le message tout de suite »   -> URC « +CMT: » + le PDU
#   mt=1 : « range-le et dis-moi seulement ou »      -> URC « +CMTI: "SM",<i> »
#          puis le systeme vient le lire par AT+CMGR=<i>.
# ModemManager, celui de la VM postmarketOS, negocie mt=1 - releve tel quel
# dans le journal : « AT+CNMI=2,1,2,1,0 ». Il liste meme la boite au demarrage
# par AT+CMGL=4. Or ce programme n emettait QUE « +CMT: » et ne tenait aucune
# boite : les SMS entrants partaient donc dans le vide, et rien n arrivait dans
# la VM. D ou ce magasin, et les commandes qui vont avec (CMGR, CMGL, CMGD).
INBOX = []                      # [{idx, pdu, sender, text, lu, ts}]
INBOX_LOCK = threading.Lock()
INBOX_MAX = 20                  # ce que +CPMS annonce comme capacite


INBOX_DERNIER = [0]             # le dernier index attribue (voir inbox_ranger)


def inbox_ranger(sender, text, pdu):
    """Range un SMS et rend son index, comme une memoire « SM ».

    [2026-09-08] LES INDEX TOURNENT. On reprenait l index 1 des que le
    telephone l avait efface (AT+CMGD=1) ; or ModemManager garde parfois ce
    message dans SA liste - son effacement a echoue chez lui (« SMS storage
    currently locked, try again later ») bien que le modem ait dit OK - et un
    +CMTI sur un index qu il croit deja connu est ignore : le SMS suivant
    passait a la trappe, une fois sur deux ou trois. On prend donc toujours
    l index libre SUIVANT, en tournant sur les 20 : un index n est repris
    qu apres 19 autres."""
    with INBOX_LOCK:
        pris = {m["idx"] for m in INBOX}
        if len(pris) >= INBOX_MAX:
            INBOX.pop(0)        # boite pleine : le plus ancien cede la place
            pris = {m["idx"] for m in INBOX}
        idx = INBOX_DERNIER[0]
        for _ in range(INBOX_MAX):
            idx = idx % INBOX_MAX + 1
            if idx not in pris:
                break
        INBOX_DERNIER[0] = idx
        INBOX.append({"idx": idx, "pdu": pdu, "sender": sender, "text": text,
                      "lu": False, "ts": time.strftime("%y/%m/%d,%H:%M:%S+08")})
        INBOX.sort(key=lambda m: m["idx"])
    return idx


def inbox_get(idx):
    with INBOX_LOCK:
        for m in INBOX:
            if m["idx"] == idx:
                return m
    return None


def semi_octets(num):
    """Un numero en demi-octets inverses, comme dans un PDU."""
    digits = re.sub(r"\D", "", num)
    if len(digits) % 2:
        digits += "F"
```

```
    return bytes(int(digits[i + 1] + digits[i], 16) for i in range(0, len(digits), 2))


def scts_maintenant():
    """L horodatage d un SMS-DELIVER (23.040 § 9.2.3.11) : six paires de
    chiffres en demi-octets inverses, puis le fuseau en quarts d heure.

    [2026-09-07] POURQUOI PAS DES ZEROS. pdu_deliver posait « bytes(7) », soit
    an 00 / mois 00 / jour 00 : une date qui n existe pas. ModemManager la
    transmet telle quelle et Chatty peut jeter le message en la lisant. Un
    modem donne l heure du reseau ; on donne celle du banc, qui est la meme.
    """
    t = time.localtime()
    out = bytearray(int(("%02d" % v)[::-1], 16) for v in
                    (t.tm_year % 100, t.tm_mon, t.tm_mday,
                     t.tm_hour, t.tm_min, t.tm_sec))
    quarts = -(time.altzone if t.tm_isdst else time.timezone) // 900
    octet = int(("%02d" % abs(quarts))[::-1], 16)
    if quarts < 0:
        octet |= 0x08                  # le signe, bit 3 du premier demi-octet
    out.append(octet)
    return bytes(out)


def pdu_deliver(sender, text):
    """Un SMS-DELIVER complet, tel qu un modem le rend sur « +CMT: ».."""
    digits = re.sub(r"\D", "", sender)
    oa = bytes([len(digits), 0x81]) + semi_octets(sender)   # 0x81 : numero national
    ud, n = gsm7_pack(text)
    scts = scts_maintenant()
    pdu = b"\x00" + b"\x04" + oa + b"\x00\x00" + scts + bytes([n]) + ud
    return pdu.hex().upper()


def gsm7_unpack(data, n):
    """L inverse de gsm7_pack : les septets tasses redeviennent des caracteres."""
    out, cur, bits = [], 0, 0
    for b in data:
        cur |= b << bits
        bits += 8
        while bits >= 7 and len(out) < n:
            out.append(chr(cur & 0x7F))
            cur >>= 7
            bits -= 7
    return "".join(out)


def parse_submit(pdu_hex):
    """Lit un SMS-SUBMIT et rend (numero, texte).

    [2026-09-06] POURQUOI IL FAUT DECODER ICI. oFono n envoie PAS les SMS en
    mode texte : il envoie « AT+CMGS=<longueur> » puis un PDU. La premiere
    version cherchait un numero entre guillemets et refusait donc tout ce qui
    venait d oFono - c est-a-dire tout ce qui vient d Android.
    """
    b = bytes.fromhex(pdu_hex)
    i = 0
    i += 1 + b[0]                      # longueur du SMSC, puis on le saute
    first = b[i]; i += 1               # octet de type
    i += 1                             # reference du message
    ndigits = b[i]; i += 1
    i += 1                             # type d adresse
    nbytes = (ndigits + 1) // 2
    num = ""
    for octet in b[i:i + nbytes]:
        num += "%d%d" % (octet & 0x0F, octet >> 4)
```

```
    num = num[:ndigits]
    i += nbytes
    i += 1                              # protocole
    dcs = b[i]; i += 1
    if first & 0x10:                    # periode de validite presente
        i += 1
    udl = b[i]; i += 1
    ud = b[i:]
    if dcs & 0x0C == 0x08:              # UCS2
        text = ud[:udl].decode("utf-16-be", errors="replace")
    else:
        text = gsm7_unpack(ud, udl)
    return num, text


class Smpp(threading.Thread):
    """L ESME : se lie au MSC, tient la liaison, et remet les SMS au modem."""

    daemon = True

    def __init__(self, deliver_cb):
        super().__init__(name="smpp")
        self.deliver = deliver_cb
        self.seq = 0
        self.s = None

    def _pack(self, cmd_id, body=b"", status=0):
        self.seq += 1
        return (len(body) + 16).to_bytes(4, "big") + cmd_id.to_bytes(4, "big") + \
            status.to_bytes(4, "big") + self.seq.to_bytes(4, "big") + body

    def _bind(self):
        body = (SMPP_ID.encode() + b"\x00" + SMPP_PW.encode() + b"\x00" +
                b"osmo\x00" + b"\x34" + b"\x00\x00" + b"\x00")
        self.s.sendall(self._pack(SMPP_BIND_TRANSCEIVER, body))
        head = self._recv(16)
        if not head:
            return False
        ln = int.from_bytes(head[:4], "big")
        cmd = int.from_bytes(head[4:8], "big")
        status = int.from_bytes(head[8:12], "big")
        self._recv(ln - 16)
        if cmd == SMPP_BIND_TRANSCEIVER_RESP and status == 0:
            log("SMPP : lie a osmo-msc comme « %s »" % SMPP_ID)
            return True
        log("SMPP : liaison refusee (statut %d)" % status)
        return False

    def _recv(self, n):
        buf = b""
        while len(buf) < n:
            try:
                chunk = self.s.recv(n - len(buf))
            except OSError:
                return None
            if not chunk:
                return None
            buf += chunk
        return buf

    def run(self):
        while True:
            try:
                self.s = socket.create_connection((SMPP_HOST, SMPP_PORT), timeout=5)
                self.s.settimeout(None)
                if not self._bind():
```

```
                    self.s.close()
                    time.sleep(5)
                    continue
                self._loop()
            except OSError as e:
                log("SMPP : %s" % e)
            finally:
                try:
                    if self.s:
                        self.s.close()
                except OSError:
                    pass
            time.sleep(5)          # le MSC redemarre, on repasse le voir

    def _loop(self):
        while True:
            head = self._recv(16)
            if not head:
                return
            ln = int.from_bytes(head[:4], "big")
            cmd = int.from_bytes(head[4:8], "big")
            seq = head[12:16]
            body = self._recv(ln - 16) if ln > 16 else b""
            if body is None:
                return
            if cmd == SMPP_ENQUIRE_LINK:
                self.s.sendall(b"\x00\x00\x00\x10" +
                               SMPP_ENQUIRE_LINK_RESP.to_bytes(4, "big") +
                               b"\x00\x00\x00\x00" + seq)
            elif cmd == SMPP_DELIVER_SM:
                try:
                    self._on_deliver(body)
                except (ValueError, IndexError) as e:
                    log("SMPP : deliver_sm illisible (%s)" % e)
                self.s.sendall(b"\x00\x00\x00\x11" +
                               SMPP_DELIVER_SM_RESP.to_bytes(4, "big") +
                               b"\x00\x00\x00\x00" + seq + b"\x00")

    def _on_deliver(self, b):
        i = 0
        _svc, i = _cstr(b, i)
        i += 2                                  # ton et npi de la source
        src, i = _cstr(b, i)
        i += 2                                  # ton et npi du destinataire
        dst, i = _cstr(b, i)
        i += 3                                  # esm_class, protocole, priorite
        _sched, i = _cstr(b, i)
        _valid, i = _cstr(b, i)
        i += 3                                  # registered, replace, coding
        i += 1                                  # sm_default_msg_id
        ln = b[i]; i += 1
        text = b[i:i + ln].decode("latin-1", errors="replace")
        log("SMS entrant de %s pour %s : %s" % (src, dst, text))
        self.deliver(src, text)


class Server(socketserver.ThreadingTCPServer):
    allow_reuse_address = True
    daemon_threads = True


CURRENT = None                 # le modem actuellement ouvert par oFono
# [2026-09-08] LE MODEM A DEUX PORTS. Sur un seul port, une fois ATD*99
# passe, plus rien ne circule en AT : le telephone ne peut ni appeler ni
# envoyer un SMS tant que la data est montee (« No AT port available »), et
# les entrants n arrivent qu en coupant le PPP. Un vrai modem a un port de
```

```
# commande et un port de donnees. Ici aussi, desormais : CURRENT tient le
# port de commande (hvc0, URC, appels, SMS) ; DATA tient le port de donnees
# (hvc1, celui que ModemManager marque ID_MM_PORT_TYPE_AT_PPP et sur lequel il
# compose ATD*99). Les deux sont des AtHandler sur le meme etat du banc ; le
# port de donnees renvoie sur CURRENT pour ce qui est l etat du telephone
# (technologie, appel en cours). Sans --data, on reste en port unique.
DATA = None                    # le port de donnees, s il y en a un


def on_smpp_sms(sender, text):
    """Un SMS arrive du banc. Deux destinataires, pas un :

    [2026-09-06] LE TELEPHONE ANDROID EST LE MEME ABONNE QUE LE MOBILE. Le banc
    n a qu un 100101, et on veut qu il sonne des DEUX cotes - dans Android par
    le RIL, et sur le mobile osmocom-bb comme avant. Or « smpp-first » detourne
    le SMS vers nous AVANT la livraison interne : le mobile ne verrait plus
    rien. On le lui remet donc nous-memes, par la commande VTY du MSC, qui elle
    livre directement au mobile sans repasser par SMPP - pas de boucle
    possible. OSMO_SMS_DUPLIQUE=0 pour n en garder qu un (Android seul).
    """
    if CURRENT is not None:
        CURRENT.deliver_sms(sender, text)
    else:
        log("oFono n est pas connecte : rien a remettre a Android")
    if DUPLIQUE:
        out = MSC.cmd("subscriber msisdn %s sms sender msisdn %s send %s"
                      % (MSISDN, sender, text))
        if out is None:
            log("copie au mobile impossible : osmo-msc injoignable")
        else:
            log("copie remise au mobile %s" % MSISDN)


# ── LA BOITE D ENTREE : LES SMS QUE LA PASSERELLE NOUS REMET ──────────────
# [2026-09-07] POURQUOI UNE SECONDE VOIE, A COTE DU SMPP. Le banc tourne avec
# « sms-over-gsup » (osmo-msc.cfg). Dans ce mode, un SM-RP recu de la radio
# part au HLR en gsm_04_11.c:742 SANS jamais passer par gsm340_rx_tpdu() :
# sms_route_mt_sms() n est pas appele, donc smpp_try_deliver() non plus. Le
# « smpp-first » et les quatre « route prefix ... 100101 » de la conf sont donc
# de la configuration morte tant que sms-over-gsup est la - l ESME ci-dessus
# est bien LIE au port 2775, et il ne recoit RIEN. Et le MT qui revient par
# GSUP est pousse directement a l abonne par la voie radio : le mobile
# osmocom-bb l a (on le lit dans mobile.log), les modems logiciels non.
#
# La passerelle previent donc les modems la ou elle decide la remise locale
# (scripts/sms-interop-relay.py, inject_mt_sms), par scripts/sms_notify.py.
# On ne duplique RIEN ici, contrairement a la voie SMPP : le mobile a deja son
# exemplaire par la radio, le lui renvoyer le ferait sonner deux fois.
class SmsInHandler(socketserver.StreamRequestHandler):
    """Une ligne JSON : {"imsi": ..., "from": ..., "text": ...}."""

    def handle(self):
        try:
            ligne = self.rfile.readline(65536)
        except OSError:
            return
        try:
            msg = json.loads(ligne.decode("utf-8", errors="replace"))
        except ValueError as e:
            log("passerelle : message illisible (%s)" % e)
            return
        imsi = str(msg.get("imsi") or "")
        if imsi and imsi != IMSI:
            return                      # ce SMS-la est pour un autre abonne
        on_passerelle_sms(str(msg.get("from") or ""), str(msg.get("text") or ""))
        try:
```

```
            self.wfile.write(b'{"ok":true}\n')
        except OSError:
            pass


def on_passerelle_sms(sender, text):
    """Aucun systeme au bout du fil ? On range quand meme : la VM peut etre en
    train de demarrer, et ModemManager liste la boite par AT+CMGL=4 des qu il
    se branche. Un SMS n a pas a se perdre parce que le modem etait absent."""
    if CURRENT is not None:
        CURRENT.deliver_sms(sender, text)
        return
    idx = inbox_ranger(sender, text, pdu_deliver(sender, text))
    log("SMS de la passerelle range en SM[%d], personne au bout du fil : %s"
        % (idx, text))


def boite_entree(port):
    """Ouvre la boite. Port deja pris = c est l autre instance du modem qui l
    a : on le dit et on continue, tout le reste du modem marche sans."""
    try:
        srv = Server((SMS_IN_HOST, port), SmsInHandler)
    except OSError as e:
        log("boite d entree SMS sur %d impossible (%s)" % (port, e))
        return
    log("boite d entree SMS sur %s:%d" % (SMS_IN_HOST, port))
    threading.Thread(target=srv.serve_forever, daemon=True).start()


class ClientHandler(AtHandler):
    """Le meme modem, mais c est NOUS qui allons vers le port.

    [2026-09-07] POURQUOI CE MODE. Quand le modem est branche sur un port serie
    de machine virtuelle, le firmware UEFI de la VM traite ce port comme une
    console : il lit les reponses du modem comme des touches pressees, croit
    qu on interrompt le demarrage et ouvre son menu - la VM ne demarre jamais.
    Le remede est qu il n y ait PERSONNE au bout du fil tant que le firmware
    travaille : QEMU ecoute (server=on,wait=off), et on vient s y connecter
    une fois le systeme charge.
    """

    def __init__(self, sock, addr, role="commande"):
        # StreamRequestHandler.setup() attend self.request et fabrique lui-meme
        # connection/rfile/wfile : on lui donne juste la douille.
        self.role = role
        self.request = sock
        self.client_address = addr
        self.server = None
        self.setup()
        try:
            self.handle()
        finally:
            self.finish()


def vm_prete(port):
    """Le systeme de la VM repond-il ? On se sert d un port temoin (le SSH de
    la machine virtuelle, 2222 par defaut).

    [2026-09-07] POURQUOI ATTENDRE. Le firmware UEFI de la VM lit tout port
    serie comme une console : si le modem se branche pendant qu il travaille,
    ses reponses passent pour des touches et le demarrage s arrete sur le menu.
    Le mode --connect rebouclant toutes les cinq secondes, il retombait dedans
    a chaque redemarrage de la VM. On ne se branche donc QUE lorsque le systeme
    est leve.
    """
```

```
    try:
        s = socket.create_connection(("127.0.0.1", port), timeout=2)
        s.close()
        return True
    except OSError:
        return False


def connect_to(dest, role="commande"):
    """dest = « hote:port » : on se connecte et on tient le modem sur ce lien."""
    host, _, port = dest.rpartition(":")
    temoin = int(os.environ.get("OSMO_VM_READY_PORT", "2222"))
    deja_dit = False
    while True:
        if temoin and not vm_prete(temoin):
            log("la VM n est pas prete (port %d ferme) - on patiente" % temoin)
            while not vm_prete(temoin):
                time.sleep(5)
            log("VM prete")
        try:
            s = socket.create_connection((host or "127.0.0.1", int(port)), timeout=10)
        except OSError as e:
            if not deja_dit:
                log("connexion a %s (port de %s) impossible (%s) - on reessaie toutes les 5 s"
                    % (dest, role, e))
                deja_dit = True
            time.sleep(5)
            continue
        deja_dit = False
        # [2026-09-07] LE DELAI SERT A SE CONNECTER, PAS A TENIR LE LIEN.
        # create_connection(timeout=10) laisse ce delai SUR la douille : la
        # lecture suivante levait socket.timeout apres dix secondes de silence,
        # handle() rendait la main et le modem se debranchait tout seul. Or un
        # modem se tait la plupart du temps - ModemManager ne l interroge que
        # par a-coups. D ou des appels qui marchent puis disparaissent, et la
        # ronde « oFono s est deconnecte / lien ferme » dans ce journal. On
        # repasse donc en bloquant, comme le lien SMPP plus haut.
        s.settimeout(None)
        log("connecte a %s (port de %s)" % (dest, role))
        try:
            ClientHandler(s, (host, int(port)), role)
        except OSError as e:
            log("lien rompu : %s" % e)
        finally:
            try:
                s.close()
            except OSError:
                pass
        log("lien ferme, on repasse dans 5 s")
        time.sleep(5)


def main():
    LTE.resume()
    if SMPP_ON:
        Smpp(on_smpp_sms).start()
    AmiEcoute().start()
    data = None
    for i, a in enumerate(sys.argv):
        if a == "--data" and i + 1 < len(sys.argv):
            data = sys.argv[i + 1]
    for i, a in enumerate(sys.argv):
        if a == "--connect" and i + 1 < len(sys.argv):
            boite_entree(SMS_IN_CONNECT)
            if data:
                threading.Thread(target=connect_to, args=(data, "data"), daemon=True).start()
```

```
            return connect_to(sys.argv[i + 1])
    boite_entree(SMS_IN_LISTEN)
    srv = Server((HOST, PORT), AtHandler)
    log("modem du banc en ecoute sur %s:%d" % (HOST, PORT))
    log("declarer dans /etc/ofono/phonesim.conf :")
    log("  [osmo]")
    log("  Address=%s" % HOST)
    log("  Port=%d" % PORT)
    log("puis : systemctl restart ofono")
    try:
        srv.serve_forever()
    except KeyboardInterrupt:
        pass
    return 0


if __name__ == "__main__":
    sys.exit(main())
```

3.425 /opt/GSM/osmo-operator/tools/osmo-pmos-build.sh

16758 octets, 284 lignes → 284 lignes

```
#!/bin/bash
# ============================================================================
# osmo-pmos-build.sh - LE NOYAU postmarketOS AVEC PPP, L IMAGE, ET LA VM QUI
# VA AVEC. (ex ~nirvana/build-pm.sh, entre dans le depot le 2026-09-08.)
#
# [2026-09-07] POURQUOI. La data 4G du telephone (ATD*99 -> CONNECT) meurt en
# NO CARRIER : le noyau linux-postmarketos-stable n a pas PPP (« # CONFIG_PPP
# is not set ») et refuse les modules non signes (lockdown). Les modules
# construits AVEC le noyau sont signes par la cle du build : il suffit donc de
# reconstruire le noyau avec CONFIG_PPP/CONFIG_PPP_ASYNC, puis l image.
#
# [2026-09-08] DEUX CHEMINS VERS CE NOYAU :
#   1. LE RACCOURCI : /opt/user_interface/kernel/pmos/ (pose par le .deb
#      osmo-build-pmos-kernel, dans l ISO et dans /var/cache/osmo-debs) porte
#      l apk deja construit, son index, sa cle publique et le commit pmaports.
#      On les copie dans le dossier de travail pmbootstrap et « pmbootstrap
#      install » prend ce noyau-la : ZERO compilation.
#   2. LES SOURCES : pas d apk, ou --rebuild -> pmbootstrap build du noyau
#      (15 a 40 min), comme avant.
#
# A lancer sous le COMPTE DE SESSION (pmbootstrap refuse root ; il demande
# sudo lui-meme), dans un terminal :
#   osmo-pmos-build                 noyau + image + VM smartphone (720x1440)
#   osmo-pmos-build tablette        idem, VM au format tablette (1280x800)
#   osmo-pmos-build --no-install    le noyau seulement
#   osmo-pmos-build --no-launch     noyau + image, sans allumer la VM
#   osmo-pmos-build --rebuild       recompiler meme si l apk avec PPP est deja la
#   osmo-pmos-build --init          seulement preparer pmbootstrap (config, pmaports, noyau)
#   JOBS=8 osmo-pmos-build          moins de coeurs (srsRAN tourne a cote)
#
# PMBOOTSTRAP ET SON DOSSIER DE TRAVAIL. pmbootstrap est celui de
# /opt/user_interface/pmos/pmbootstrap (patche : patches/pmbootstrap-osmo-
# bench-qemu.patch - le modem sur un port serie PCI, la carte son du banc, la
# taille d ecran, et le port 5038 qui n est pas un adb). Le dossier de travail
# est celui du pmbootstrap_v3.cfg du compte, sinon ~/.local/var/pmbootstrap
# (OSMO_PMB_WORK pour un autre). Sans config, on en ecrit une depuis le
# gabarit /opt/user_interface/pmos/pmbootstrap_v3.cfg : Phosh, fr_FR,
# hostname pmos-gsm, et les paquets du modem (ppp, networkmanager-ppp...).
# pmaports est clone au commit du noyau (pmaports.commit) et recoit le patch
# PPP : meme pkgver/pkgrel que l apk, sinon pmbootstrap ne le prefere pas.
# ============================================================================
```

```
set -u

UI="${OSMO_UI_ROOT:-/opt/user_interface}"
PM="$UI/pmos"
KD="$UI/kernel/pmos"
PMB="${PMB:-}"
for _p in "$PMB" "$PM/pmbootstrap/pmbootstrap.py" "$(command -v pmbootstrap 2>/dev/null)"
 ↪  "$HOME/.local/bin/pmbootstrap"; do
    [ -n "$_p" ] && [ -x "$_p" ] && { PMB="$_p"; break; }
done
CFG="$HOME/.config/pmbootstrap_v3.cfg"
WORK="${OSMO_PMB_WORK:-}"
[ -z "$WORK" ] && [ -f "$CFG" ] && WORK="$(sed -n 's/^work *= *//p' "$CFG" | head -1)"
[ -z "$WORK" ] && WORK="$HOME/.local/var/pmbootstrap"
PMAPORTS="$(sed -n 's/^aports *= *//p' "$CFG" 2>/dev/null | head -1)"
[ -n "$PMAPORTS" ] || PMAPORTS="$WORK/cache_git/pmaports"
KCFG="$PMAPORTS/device/main/linux-postmarketos-stable/config-stable.x86_64"
IMG="$WORK/chroot_native/home/pmos/rootfs/qemu-amd64.img"
JOBS="${JOBS:-$(nproc)}"
PASS="${OSMO_PMOS_PASSWORD:-147147}"
FORMAT=smartphone
INSTALL=1
LAUNCH=1
REBUILD=0
INIT_ONLY=0

for a in "$@"; do
    case "$a" in
        smartphone|tablette) FORMAT="$a" ;;
        --no-install) INSTALL=0 ;;
        --no-launch)  LAUNCH=0 ;;
        --rebuild)    REBUILD=1 ;;
        --init)       INIT_ONLY=1 ;;
        *) echo "argument inconnu : $a (smartphone|tablette|--no-install|--no-launch|--rebuild|--init)" >&2; exit
         ↪  2 ;;
    esac
done

RED='\033[0;31m'; GREEN='\033[0;32m'; YELLOW='\033[1;33m'; CYAN='\033[0;36m'; NC='\033[0m'
say()  { echo -e "${CYAN}→${NC} $*"; }
ok()   { echo -e "${GREEN}✓${NC} $*"; }
warn() { echo -e "${YELLOW}!${NC} $*"; }
die()  { echo -e "${RED}✗${NC} $*" >&2; exit 1; }

# [2026-09-09] ROOT ACCEPTE. La session de l ISO EST root (voir osmo-pmos-
# qemu) : refuser root, c etait refuser l icone. pmbootstrap accepte
# --as-root ; sous un compte, rien ne change.
PMB_OPTS=(); [ "$(id -u)" -eq 0 ] && { PMB_OPTS=(--as-root); warn "en root : pmbootstrap --as-root (dossier de
 ↪  travail de root)"; }
pmb() { "$PMB" "${PMB_OPTS[@]}" "$@"; }
[ -n "$PMB" ] && [ -x "$PMB" ] || die "pmbootstrap introuvable ($PM/pmbootstrap manquant : osmo-pmos-install)"

# ── 0. LA CONFIG pmbootstrap DU COMPTE ──────────────────────────────
if [ ! -f "$CFG" ]; then
    [ -f "$PM/pmbootstrap_v3.cfg" ] || die "pas de $CFG ni de gabarit $PM/pmbootstrap_v3.cfg"
    mkdir -p "$(dirname "$CFG")" "$WORK"
    sed "s#@WORK@#$WORK#g; s#@APORTS@#$PMAPORTS#g" "$PM/pmbootstrap_v3.cfg" > "$CFG"
    ok "config pmbootstrap ecrite : $CFG (work = $WORK)"
fi
mkdir -p "$WORK"
# Le numero de version du dossier de travail : sans lui, pmbootstrap demande
# « init ». C est celui de pmb/config/__init__.py (work_version).
if [ ! -f "$WORK/version" ]; then
    _wv="$(grep -m1 -oE '^work_version *= *[0-9]+' "$(dirname "$PMB")/pmb/config/__init__.py" 2>/dev/null | grep
     ↪  -oE '[0-9]+$')"
    echo "${_wv:-8}" > "$WORK/version"
```

```
fi

# ── 1. pmaports, AU COMMIT DU NOYAU, avec le patch PPP ──────────────────
_commit="$(cat "$KD/pmaports.commit" 2>/dev/null || true)"
# [2026-09-09] UN SEUL osmo-pmos-build A LA FOIS. Deux lancements (l icone
# cliquee pendant qu un premier clone tourne) se marchaient dessus : le second
# voyait un .git sans commit - celui du clone EN COURS - et l effacait
# (« fatal: could not open .../tmp_pack_... » chez le premier). Verrou sur le
# dossier de travail, et on ne touche jamais a un clone qui a encore son git.
exec 9>"$WORK/.osmo-pmos-build.lock"
flock -n 9 || die "un autre osmo-pmos-build tourne deja (verrou $WORK/.osmo-pmos-build.lock) : attends-le, ou
 ↪  ferme-le"
# Un clone interrompu (reseau coupe, fenetre fermee) laisse un .git SANS
# commit : on passait le clone, et on mourait plus bas sur « config noyau
# introuvable ». Un depot sans HEAD, et sans git dessus, est un depot absent.
if [ -d "$PMAPORTS/.git" ] && ! git -C "$PMAPORTS" rev-parse --verify HEAD >/dev/null 2>&1 \
   && ! pgrep -f "git(-remote-https)? .*$PMAPORTS" >/dev/null 2>&1; then
    warn "pmaports : clone vide ou interrompu, on le refait ($PMAPORTS)"
    rm -rf "$PMAPORTS"
fi
# [2026-09-09] PMAPORTS SANS GITLAB QUAND C EST POSSIBLE. Le clone complet
# (des centaines de Mo, GitLab lent) a echoue deux fois sur trois depuis le
# live : git efface alors tout et le telephone ne s allume jamais. Dans
# l ordre : la copie posee par l ISO (88-lte-pmos.sh : $PM/pmaports, au commit
# du noyau, sans reseau) ; sinon UN SEUL commit recupere par fetch --depth 1
# (quelques dizaines de Mo) ; le clone complet seulement si le commit est
# inconnu.
if [ ! -d "$PMAPORTS/.git" ]; then
    mkdir -p "$(dirname "$PMAPORTS")"
    if [ -d "$PM/pmaports/.git" ] && git -C "$PM/pmaports" rev-parse --verify HEAD >/dev/null 2>&1; then
        say "pmaports : copie de celui de l ISO ($PM/pmaports -> $PMAPORTS)..."
        cp -a "$PM/pmaports" "$PMAPORTS" || die "copie de pmaports"
        # pmbootstrap lit channels.cfg dans origin/main : la ref doit exister.
        git -C "$PMAPORTS" rev-parse -q --verify origin/main >/dev/null 2>&1 \
            || git -C "$PMAPORTS" update-ref refs/remotes/origin/main HEAD
        ok "pmaports au commit $(git -C "$PMAPORTS" rev-parse --short HEAD) (celui de l ISO)"
    elif [ -n "$_commit" ] && [ "$_commit" != inconnu ]; then
        say "pmaports : le commit $_commit du noyau, seul (fetch --depth 1)..."
        ( git init -q "$PMAPORTS" && cd "$PMAPORTS" \
          && git remote add origin https://gitlab.postmarketos.org/postmarketOS/pmaports.git \
          && git fetch -q --depth 1 origin "$_commit" && git checkout -q -B main FETCH_HEAD \
          && git update-ref refs/remotes/origin/main FETCH_HEAD ) \
          || { rm -rf "$PMAPORTS"; die "pmaports : fetch du commit $_commit impossible (reseau ?) - relance quand
           ↪  GitLab repond"; }
        ok "pmaports au commit $_commit (celui du noyau)"
    else
        say "clone de pmaports ($PMAPORTS)..."
        git clone https://gitlab.postmarketos.org/postmarketOS/pmaports.git "$PMAPORTS" || die "clone pmaports"
    fi
fi
[ -f "$KCFG" ] || die "config noyau introuvable : $KCFG"
if grep -q '^CONFIG_PPP_ASYNC=m' "$KCFG"; then
    ok "config noyau : PPP deja active ($(grep -c '^CONFIG_PPP' "$KCFG") symboles)"
else
    say "config noyau : ajout de PPP"
    cp -a "$KCFG" "$KCFG.avant-ppp"
    _patch=""
    for _p in "$KD/pmaports-linux-postmarketos-stable-ppp.patch"
     ↪  "$PM/patches/pmaports-linux-postmarketos-stable-ppp.patch"; do
        [ -f "$_p" ] && { _patch="$_p"; break; }
    done
    if [ -n "$_patch" ] && git -C "$PMAPORTS" apply --check "$_patch" 2>/dev/null; then
        git -C "$PMAPORTS" apply "$_patch" && ok "patch pmaports applique ($_patch)"
    else
        python3 - "$KCFG" <<'PY' || die "patch de la config impossible"
import sys
```

```
p = sys.argv[1]; s = open(p).read()
old = "# CONFIG_PPP is not set\n"
new = ("CONFIG_PPP=m\nCONFIG_PPP_BSDCOMP=m\nCONFIG_PPP_DEFLATE=m\nCONFIG_PPP_FILTER=y\n"
       "CONFIG_PPP_MULTILINK=y\nCONFIG_PPP_ASYNC=m\nCONFIG_PPP_SYNC_TTY=m\n")
if old not in s:
    sys.exit("pas de « # CONFIG_PPP is not set » dans la config : a regarder a la main")
open(p, "w").write(s.replace(old, new, 1))
PY
        ok "config noyau : PPP=m, PPP_ASYNC=m (sauvegarde : $KCFG.avant-ppp)"
    fi
fi
# La somme de controle : abuild verifie le sha512 de chaque fichier source, et
# la config modifiee n a plus celui de l APKBUILD (« FAILED, Use 'abuild
# checksum' »). On le remet a jour, toujours, ca ne coute rien.
APKB="$(dirname "$KCFG")/APKBUILD"
somme="$(sha512sum "$KCFG" | cut -d' ' -f1)"
if grep -q "^$somme  $(basename "$KCFG")\$" "$APKB"; then
    ok "APKBUILD : sha512 de la config a jour"
else
    sed -i "s/^[0-9a-f]\{128\}  $(basename "$KCFG")\$/$somme  $(basename "$KCFG")/" "$APKB" \
        && ok "APKBUILD : sha512 de la config mis a jour" \
        || die "sha512 de la config non remplace dans $APKB"
fi

# ── 2. LES REGLAGES pmbootstrap : Phosh, pas la console ──────────────────
ui="$(pmb config ui 2>/dev/null | tail -1)"
if [ "$ui" != "phosh" ]; then
    warn "pmbootstrap config ui = « $ui » : un install ferait une VM sans bureau"
    pmb config ui phosh || die "impossible de poser ui=phosh"
fi
ok "ui = phosh, device = $(pmb config device 2>/dev/null | tail -1), jobs = $JOBS, work = $WORK"

# ── 3. LE NOYAU : LE RACCOURCI D ABORD ───────────────────────────────────
# L apk du .deb osmo-build-pmos-kernel, copie dans le depot local de
# pmbootstrap avec son index et sa cle : « pmbootstrap install » le prefere
# au depot en ligne (meme version, paquet local). Sinon, un apk deja
# construit ici, plus recent que la config, qui a ppp_async.ko. Sinon on
# compile. --rebuild force la compilation.
_a_ppp() { grep -q 'ppp_async\.ko' < <(tar tzf "$1" 2>/dev/null); }
if [ "$REBUILD" = 0 ] && [ -f "$KD"/linux-postmarketos-stable-*.apk ]; then
    read -r _chan _parch _ver < "$KD/apk.info" 2>/dev/null || { _chan=edge; _parch=x86_64; }
    _dst="$WORK/packages/$_chan/$_parch"
    mkdir -p "$_dst" "$WORK/config_apk_keys"
    for f in "$KD"/linux-postmarketos-stable-*.apk "$KD"/APKINDEX.tar.gz; do
        [ -f "$f" ] && [ ! -f "$_dst/$(basename "$f")" ] && cp -f "$f" "$_dst/"
    done
    cp -n "$KD"/*.rsa.pub "$WORK/config_apk_keys/" 2>/dev/null || true
    touch "$_dst"/linux-postmarketos-stable-*.apk 2>/dev/null || true   # plus recent que la config (fichier du
    ↪  chroot, uid 12345 : pas grave)
    ok "noyau PPP pris dans $KD (raccourci : pas de compilation) -> $_dst"
fi
apk="$(find "$WORK/packages" -name 'linux-postmarketos-stable-[0-9]*.apk' -newer "$KCFG" 2>/dev/null | head -1)"
if [ "$REBUILD" = 0 ] && [ -n "$apk" ] && _a_ppp "$apk"; then
    ok "noyau deja construit avec PPP : $apk (--rebuild pour recompiler)"
else
    [ "$INIT_ONLY" = 1 ] && { warn "pas d apk PPP ; --init ne compile pas (relancer sans --init)"; exit 0; }
    sudo -v || die "sudo refuse"
    say "build du noyau (long : compter 15 a 40 min)..."
    pmb -y -j "$JOBS" build --force linux-postmarketos-stable \
        || die "build echoue - voir $WORK/log.txt (pmbootstrap log)"
    apk="$(find "$WORK/packages" -name 'linux-postmarketos-stable-[0-9]*.apk' -newer "$KCFG" 2>/dev/null | head
    ↪  -1)"
    [ -n "$apk" ] || die "aucun .apk du noyau dans $WORK/packages"
fi
_a_ppp "$apk" && ok "noyau : $apk contient ppp_async.ko" || die "le paquet $apk n a PAS ppp_async.ko - la config n
↪  a pas ete prise"
```

```
[ "$INIT_ONLY" = 1 ] && { ok "pmbootstrap pret (--init)"; exit 0; }
[ "$INSTALL" = 1 ] || { ok "fini (--no-install)"; exit 0; }

# -- 4. L IMAGE ----------------------------------------------------------
# On ecrit l image que la VM utilise : elle ne doit pas tourner dessus.
# Une image deja fournie (/opt/user_interface/pmos/image/qemu-amd64.img.zst,
# la VM de reference) est decompressee a la place d un install : c est le
# raccourci n° 2 - OSMO_PMOS_FRESH=1 pour installer quand meme.
if pgrep -f "qemu-system-x86_64.*$(basename "$IMG")" >/dev/null 2>&1; then
    die "la VM tourne encore sur $IMG : arrete-la (osmo-pmos-qemu stop) puis relance"
fi
sudo -v || die "sudo refuse"
# -- L IMAGE DE REFERENCE : LA, OU TELECHARGEE ---------------------------
# [2026-10-01] Si le .zst manque (ISO construite sans) ou est CORROMPU (zstd -t
# le dit : une copie coupee, un 404 enregistre tel quel, une cle retiree trop
# tot), on le reprend sur la release GitHub « pmos-image » - celle que le
# workflow embarque dans l ISO. 1,7 Go, reprise possible (curl -C -), teste
# avant d etre mis en place. OSMO_PMOS_IMAGE_URL pour une autre adresse, vide
# pour ne rien telecharger (on retombe alors sur pmbootstrap install).
ZST="$PM/image/qemu-amd64.img.zst"
: "${OSMO_PMOS_IMAGE_URL=https://github.com/bbaranoff/osmo-operator/releases/download/pmos-image/qemu-
↪  amd64.img.zst}"
pmos_zst_ok() { [ -f "$ZST" ] && zstd -t "$ZST" >/dev/null 2>&1; }
if [ "${OSMO_PMOS_FRESH:-0}" != 1 ] && [ ! -f "$IMG" ] && command -v zstd >/dev/null 2>&1 \
   && ! pmos_zst_ok && [ -n "$OSMO_PMOS_IMAGE_URL" ] && command -v curl >/dev/null 2>&1; then
    [ -f "$ZST" ] && warn "image de reference illisible ($ZST) - retelechargee"
    say "telechargement de l image de reference (1,7 Go, reprise possible) : $OSMO_PMOS_IMAGE_URL"
    sudo install -d "$(dirname "$ZST")"
    if sudo curl -fL -C - --retry 3 --retry-delay 5 -o "$ZST.part" "$OSMO_PMOS_IMAGE_URL" \
       && zstd -t "$ZST.part" >/dev/null 2>&1; then
        sudo mv -f "$ZST.part" "$ZST" && sudo chmod a+r "$ZST"
        ok "image de reference recuperee : $ZST ($(du -h "$ZST" | cut -f1))"
    else
        sudo rm -f "$ZST.part" "$ZST"
        warn "telechargement impossible ou archive invalide - install classique (pmbootstrap install)"
    fi
fi
if [ "${OSMO_PMOS_FRESH:-0}" != 1 ] && [ ! -f "$IMG" ] && pmos_zst_ok; then
    say "image de reference decompressee ($PM/image/qemu-amd64.img.zst -> $IMG)..."
    mkdir -p "$(dirname "$IMG")"
    zstd -d --sparse -o "$IMG" "$PM/image/qemu-amd64.img.zst" && ok "image prete : $IMG (VM de reference)" \
        || { rm -f "$IMG"; warn "decompression echouee - install classique"; }
fi
if [ ! -f "$IMG" ] || [ "${OSMO_PMOS_FRESH:-0}" = 1 ]; then
    say "installation de l image (nouvelle VM, mot de passe $PASS)..."
    pmb -y install --password "$PASS" || die "install echoue - voir $WORK/log.txt"
    ok "image prete : $IMG"
fi
[ "$LAUNCH" = 1 ] || { ok "fini (--no-launch) - « osmo-pmos-qemu $FORMAT » branche VM, modem et voix"; exit 0; }

# -- 5. LA VM, ET LE MODEM DES QU ELLE REPOND ----------------------------
# osmo-pmos-qemu garde la main (fenetre SDL) et son guetteur branche le modem
# et la voix tout seul (osmo-pmos-setup) des que le SSH de la VM repond.
QEMU="$(command -v osmo-pmos-qemu 2>/dev/null || echo "$PM/bin/osmo-pmos-qemu.sh")"
export OSMO_PMOS_FROM_BUILD=1     # osmo-pmos-qemu ne doit pas nous rappeler
exec "$QEMU" "$FORMAT"
```

3.426 /opt/GSM/osmo-operator/tools/osmo-pmos-install.sh

15545 octets, 277 lignes → 277 lignes

```
#!/bin/bash
# ============================================================================
# osmo-pmos-install.sh - L UI SMARTPHONE DU BANC, POSEE DANS /opt/user_interface
```

```
# UNE FOIS POUR LES TROIS CHEMINS (addition.sh, update.sh, le build ISO).
#
# [2026-09-08] Le telephone du banc (la VM postmarketOS/Phosh, son modem, son
# noyau PPP) vivait eparpille : les scripts dans le depot, pmbootstrap dans le
# home de nirvana (patche a la main), le noyau dans ~nirvana/test, le script
# de build dans ~nirvana. Tout cela vit maintenant a UN endroit :
#
#   /opt/user_interface/
#     pmos/                       L UI SMARTPHONE
#       bin/                      osmo-pmos.sh osmo-pmos-qemu.sh osmo-pmos-setup.sh
#                                 osmo-pmos-build.sh osmo-phonesim-banc.py at-cmd.py
#                                 (copies du depot ; /usr/local/bin/osmo-pmos* y pointent)
#       pmbootstrap/              pmbootstrap, PATCHE (patches/pmbootstrap-osmo-bench-qemu.patch)
#                                 -> /usr/local/bin/pmbootstrap
#       patches/                  les deux patches (pmbootstrap, pmaports PPP)
#       pmbootstrap_v3.cfg        le gabarit de config (Phosh, fr, paquets du modem)
#       image/qemu-amd64.img.zst  (optionnel) la VM de reference, compressee
#       README
#     kernel/                     LES .deb DU NOYAU pmOS PPP
#       osmo-build-pmos-kernel_*.deb
#       pmos/                     ce que le .deb pose : apk, APKINDEX, cle .pub,
#                                 pmaports.commit, patch (lu par osmo-pmos-build)
#
#   osmo-pmos-install               = --layout --launchers --kernel
#   osmo-pmos-install --layout      bin/, patches/, gabarit, pmbootstrap (clone + patch si absent)
#   osmo-pmos-install --launchers   /usr/local/bin/osmo-pmos*, pmbootstrap, .desktop
#   osmo-pmos-install --kernel      dpkg -i du .deb du noyau (kernel/ ou /var/cache/osmo-debs)
#   osmo-pmos-install --image <img>  range une image de VM (compressee en zstd) dans pmos/image/
#   osmo-pmos-install --image telecharger   la prend sur la release GitHub pmos-image (aussi
#                                 le repli automatique si <img> manque ou est corrompue)
#
# Idempotent ; utilisable en session ET dans le chroot de l ISO (aucun
# systemctl, aucun reseau requis si pmbootstrap est deja la).
# ============================================================================
set -u

REPO="${OSMO_REPO:-${DIR:-/opt/GSM/osmo-operator}}"
UI="${OSMO_UI_ROOT:-/opt/user_interface}"
PM="$UI/pmos"
KDIR="$UI/kernel"
DEB_CACHE="${OSMO_DEB_CACHE:-/var/cache/osmo-debs}"
PMB_GIT="${OSMO_PMB_GIT:-https://gitlab.postmarketos.org/postmarketOS/pmbootstrap.git}"
# [2026-09-08] Le .deb du noyau PPP est aussi publie sur GitHub (Git LFS) :
# https://github.com/bbaranoff/pmos_ppp_kernel - c est le repli quand ni
# /opt/user_interface/kernel ni /var/cache/osmo-debs ne l ont (une machine
# qui n est pas celle de reference, un build ISO sur une autre machine).
KERNEL_URL="${OSMO_PMOS_KERNEL_URL:-
 ↪  https://media.githubusercontent.com/media/bbaranoff/pmos_ppp_kernel/main/osmo-build-pmos-
 ↪  kernel_7.2.3.0+ppp~noble_amd64.deb}"

: "${GREEN:=}"; : "${YELLOW:=}"; : "${CYAN:=}"; : "${RED:=}"; : "${NC:=}"
_p_say()  { echo -e "  ${CYAN}→${NC} $*"; }
_p_ok()   { echo -e "      ${GREEN}✓${NC} $*"; }
_p_warn() { echo -e "      ${YELLOW}!${NC} $*"; }

pmos_layout() {
    _p_say "l UI smartphone dans $PM"
    install -d "$PM/bin" "$PM/patches" "$KDIR"
    local f
    for f in osmo-pmos.sh osmo-pmos-qemu.sh osmo-pmos-setup.sh osmo-pmos-build.sh osmo-phonesim-banc.py
     ↪  at-cmd.py; do
        [ -f "$REPO/tools/$f" ] && install -m755 "$REPO/tools/$f" "$PM/bin/$f"
    done
    for f in pmbootstrap-osmo-bench-qemu.patch pmaports-linux-postmarketos-stable-ppp.patch; do
        [ -f "$REPO/patches/$f" ] && install -m644 "$REPO/patches/$f" "$PM/patches/$f"
    done
```

```
    [ -f "$REPO/configs/pmos/pmbootstrap_v3.cfg" ] && install -m644 "$REPO/configs/pmos/pmbootstrap_v3.cfg"
     ↪  "$PM/pmbootstrap_v3.cfg"
    # pmbootstrap : celui deja la (le .deb osmo-build-pmbootstrap du Dockerfile,
    # ou une pose precedente), sinon un clone. Puis le patch du banc, s il
    # n est pas deja dedans (osmo_bench_args dans pmb/commands/qemu.py).
    if [ ! -f "$PM/pmbootstrap/pmbootstrap.py" ]; then
        if git clone -q "$PMB_GIT" "$PM/pmbootstrap" 2>/dev/null; then
            _p_ok "pmbootstrap clone ($PMB_GIT)"
        else
            _p_warn "pmbootstrap : pas de clone possible (hors ligne ?) - osmo-pmos-build le redemandera"
        fi
    fi
    if [ -f "$PM/pmbootstrap/pmbootstrap.py" ]; then
        # [2026-09-09] « deja patche » ne veut pas dire « a jour » : le .deb
        # osmo-build-pmbootstrap porte le patch d une date, le depot celui du
        # jour (bootindex=0 du disque, sinon OVMF passe par PXE). Sur un
        # clone git, on repart des fichiers d origine et on applique LE
        # patch courant - idempotent ; sans git, on se fie aux marqueurs.
        # Marqueur du patch le plus recent : OSMO_PMOS_HP (2026-09-09, la carte
        # « combine » nommee) ; avant lui bootindex=0 (2026-09-09 matin).
        if [ -d "$PM/pmbootstrap/.git" ] && ! grep -q 'OSMO_PMOS_HP' "$PM/pmbootstrap/pmb/commands/qemu.py"
         ↪  2>/dev/null \
           && git -C "$PM/pmbootstrap" checkout -q -- pmb 2>/dev/null \
           && git -C "$PM/pmbootstrap" apply "$PM/patches/pmbootstrap-osmo-bench-qemu.patch" 2>/dev/null; then
            _p_ok "pmbootstrap : patch du banc remis a jour (carte combine nommee, bootindex du disque, modem
             ↪  serie PCI, son du banc)"
        elif grep -q 'osmo_bench_args' "$PM/pmbootstrap/pmb/commands/qemu.py" 2>/dev/null; then
            _p_ok "pmbootstrap deja patche (osmo_bench_args$(grep -q 'OSMO_PMOS_HP'
             ↪  "$PM/pmbootstrap/pmb/commands/qemu.py" 2>/dev/null && echo ', carte combine nommee'))"
        elif git -C "$PM/pmbootstrap" apply --check "$PM/patches/pmbootstrap-osmo-bench-qemu.patch" 2>/dev/null;
         ↪  then
            git -C "$PM/pmbootstrap" apply "$PM/patches/pmbootstrap-osmo-bench-qemu.patch" && _p_ok "pmbootstrap
             ↪  patche (modem serie PCI, son du banc, ecran, port 5038)"
        else
            _p_warn "le patch pmbootstrap ne s applique pas sur cette version - a regarder (git -C $PM/pmbootstrap
             ↪  apply --check)"
        fi
        find "$PM/pmbootstrap" -name '__pycache__' -type d -exec rm -rf {} + 2>/dev/null || true
        find "$PM/pmbootstrap" -name '*.bak-*' -delete 2>/dev/null || true
    fi
    cat > "$PM/README" <<EOF
L UI SMARTPHONE DU BANC - une VM postmarketOS/Phosh branchee sur le modem
du banc (tools/osmo-phonesim-banc.py). Pose par osmo-pmos-install
(depot : $REPO, tools/osmo-pmos-install.sh).

  bin/                  les scripts (osmo-pmos, osmo-pmos-qemu, osmo-pmos-setup,
                        osmo-pmos-build) - /usr/local/bin/* y pointent
  pmbootstrap/          pmbootstrap patche pour le banc (patches/)
  pmbootstrap_v3.cfg    gabarit : ~/.config/pmbootstrap_v3.cfg du compte de session
  image/                (optionnel) qemu-amd64.img.zst, la VM de reference
  ../kernel/            le .deb du noyau PPP, et kernel/pmos/ son contenu

Premiere fois (l icone le fait toute seule ; root accepte : --as-root est passe) :
  osmo-pmos-build            noyau (raccourci : l apk de ../kernel/pmos), image, VM
  osmo-pmos-qemu smartphone  ensuite : la VM, le modem et la voix
Mot de passe de la VM : 147147.
EOF
    # Lisible et executable par le compte de session : pmbootstrap y est lance
    # sous lui, pas sous root.
    chmod -R a+rX "$PM" "$KDIR" 2>/dev/null || true
    _p_ok "$PM : bin/ ($(ls "$PM/bin" | wc -l) scripts), patches/, gabarit, pmbootstrap$( [ -f
     ↪  "$PM/pmbootstrap/pmbootstrap.py" ] && echo '' || echo ' (absent)')"
}

pmos_launchers() {
    local f n
```

```
    for f in osmo-pmos osmo-pmos-qemu osmo-pmos-setup osmo-pmos-build; do
        [ -f "$PM/bin/$f.sh" ] && ln -sfn "$PM/bin/$f.sh" "/usr/local/bin/$f"
    done
    [ -f "$PM/pmbootstrap/pmbootstrap.py" ] && ln -sfn "$PM/pmbootstrap/pmbootstrap.py" /usr/local/bin/pmbootstrap
    # [2026-10-01] Terminal=false partout : chaque Exec passe par
    # tools/osmo-gtk-run.py (fenetre GTK du banc, terminal VTE dedans quand la
    # commande parle ou pose des questions) - aucun favori n ouvre de terminal.
    # Les icones : la VM aux deux formats, l arret, le rebranchement du modem
    # (memes .desktop que update.sh / osmo-extras-install.sh, une source ici).
    install -d /usr/share/applications
    cat > /usr/share/applications/osmo-pmos.desktop <<'PMD'
[Desktop Entry]
Type=Application
Name=Le telephone du banc (postmarketOS)
Comment=Une VM postmarketOS/Phosh branchee sur le modem du banc - osmo-pmos up
Exec=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py --tty --root --title "Telephone (postmarketOS)" --
 ↪  /usr/local/bin/osmo-pmos up
Icon=phone
Terminal=false
Categories=Network;Telephony;
Keywords=postmarketos;pmos;qemu;modem;gsm;telephone;
Actions=status;stop;build;
[Desktop Action status]
Name=Etat du telephone (osmo-pmos status)
Exec=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py --root --keep-open --title "Etat du telephone" --
 ↪  /usr/local/bin/osmo-pmos status
[Desktop Action stop]
Name=Arreter le telephone (osmo-pmos stop)
Exec=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py --root --close-on-exit --title "Arret du telephone" --
 ↪  /usr/local/bin/osmo-pmos stop
[Desktop Action build]
Name=Construire l image (noyau PPP + pmbootstrap install)
Exec=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py --tty --title "Image postmarketOS (build)" --
 ↪  /usr/local/bin/osmo-pmos-build --no-launch
PMD
    for n in smartphone tablette; do
        local r=720x1440 ic=phone; [ "$n" = tablette ] && { r=1280x800; ic=tablet; }
        cat > "/usr/share/applications/osmo-pmos-$n.desktop" <<PMD
[Desktop Entry]
Type=Application
Name=Telephone du banc - $n
Comment=La VM postmarketOS au format $n ($r), le modem et la voix branches tout seuls
Exec=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py --tty --title "Telephone du banc (VM)" -- env OSMO_PMOS_RES=$r
 ↪  /usr/local/bin/osmo-pmos-qemu
Icon=$ic
Terminal=false
Categories=Network;Telephony;
Keywords=postmarketos;pmos;qemu;modem;gsm;telephone;$n;
Actions=stop;setup;nomodem;
[Desktop Action stop]
Name=Arreter le telephone (osmo-pmos-qemu stop)
Exec=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py --close-on-exit --title "Arret du telephone" --
 ↪  /usr/local/bin/osmo-pmos-qemu stop
[Desktop Action setup]
Name=Rebrancher le modem et la voix (osmo-pmos-setup)
Exec=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py --tty --title "Modem et voix du telephone" --
 ↪  /usr/local/bin/osmo-pmos-setup
[Desktop Action nomodem]
Name=VM seule, sans modem
Exec=/opt/GSM/osmo-operator/tools/osmo-gtk-run.py --tty --title "Telephone du banc (VM)" -- env OSMO_PMOS_RES=$r
 ↪  OSMO_PMOS_MODEM=0 /usr/local/bin/osmo-pmos-qemu
PMD
    done
    chmod 644 /usr/share/applications/osmo-pmos*.desktop
    update-desktop-database /usr/share/applications >/dev/null 2>&1 || true
```

```
    _p_ok "lanceurs : /usr/local/bin/osmo-pmos, osmo-pmos-qemu, osmo-pmos-setup, osmo-pmos-build, pmbootstrap ; 3
     ↪  .desktop"
}

pmos_kernel() {
    local deb
    if [ -f "$KDIR"/pmos/linux-postmarketos-stable-*.apk ]; then
        _p_ok "noyau PPP deja pose : $KDIR/pmos/$(basename "$(ls "$KDIR"/pmos/linux-postmarketos-stable-*.apk |
         ↪  head -1)")"
        return 0
    fi
    deb="$(ls -1 "$KDIR"/osmo-build-pmos-kernel_*.deb "$DEB_CACHE"/osmo-build-pmos-kernel_*.deb 2>/dev/null |
     ↪  head -1)"
    if [ -z "$deb" ] && [ -n "$KERNEL_URL" ] && command -v curl >/dev/null 2>&1; then
        install -d "$KDIR"
        deb="$KDIR/$(basename "$KERNEL_URL")"
        _p_say "telechargement du noyau PPP ($KERNEL_URL)..."
        if curl -fsSL --retry 3 -o "$deb.part" "$KERNEL_URL" && dpkg-deb --info "$deb.part" >/dev/null 2>&1; then
            mv -f "$deb.part" "$deb"; cp -f "$deb" "$DEB_CACHE/" 2>/dev/null || true
        else
            rm -f "$deb.part"; deb=""
        fi
    fi
    if [ -z "$deb" ]; then
       _p_warn "pas de .deb du noyau pmOS ($KDIR, $DEB_CACHE, GitHub) : osmo-pmos-build compilera le noyau (long)"
        return 0
    fi
    dpkg -i --force-overwrite "$deb" >/dev/null 2>&1 && _p_ok "noyau PPP pose depuis $(basename "$deb")" ||
     ↪  _p_warn "dpkg -i $(basename "$deb") a echoue"
    [ -f "$KDIR/$(basename "$deb")" ] || cp -f "$deb" "$KDIR/"
}

# [2026-10-01] L IMAGE PEUT VENIR DU RESEAU. --image sans fichier derriere
# (absent), ou un .zst corrompu (zstd -t), ou --image telecharger : on prend
# la release GitHub « pmos-image », la meme que l ISO embarque. Reprise
# possible, archive testee avant d etre mise en place.
PMOS_IMAGE_URL="${OSMO_PMOS_IMAGE_URL-https://github.com/bbaranoff/osmo-operator/releases/download/pmos-
 ↪  image/qemu-amd64.img.zst}"
pmos_image_telecharger() {
    local dst="$1"
    [ -n "$PMOS_IMAGE_URL" ] || { _p_warn "pas d URL d image (OSMO_PMOS_IMAGE_URL vide)"; return 1; }
    command -v curl >/dev/null 2>&1 || { _p_warn "curl absent - image non telechargee"; return 1; }
    command -v zstd >/dev/null 2>&1 || { _p_warn "zstd absent - image non verifiee, non telechargee"; return 1; }
    install -d "$(dirname "$dst")"
    _p_say "telechargement de l image de reference (1,7 Go, reprise possible) : $PMOS_IMAGE_URL"
    if curl -fL -C - --retry 3 --retry-delay 5 -o "$dst.part" "$PMOS_IMAGE_URL" \
       && zstd -t "$dst.part" >/dev/null 2>&1; then
        mv -f "$dst.part" "$dst"
        return 0
    fi
    rm -f "$dst.part"
    _p_warn "telechargement impossible ou archive invalide ($PMOS_IMAGE_URL)"
    return 1
}

pmos_image() {
    local src="$1" dst="$PM/image/qemu-amd64.img.zst"
    install -d "$PM/image"
    if [ "$src" = telecharger ] || [ ! -f "$src" ]; then
        [ "$src" = telecharger ] || _p_warn "image introuvable : $src - telechargement a la place"
        pmos_image_telecharger "$dst" || return 1
        chmod a+r "$dst"
        _p_ok "image de reference : $dst ($(du -h "$dst" | cut -f1))"
        return 0
    fi
    case "$src" in
```

```
        *.zst) if command -v zstd >/dev/null 2>&1 && ! zstd -t "$src" >/dev/null 2>&1; then
                   _p_warn "image corrompue : $src (zstd -t) - telechargement a la place"
                   pmos_image_telecharger "$dst" || return 1
               elif [ "$(readlink -f "$src")" != "$(readlink -f "$dst")" ]; then
                   cp -f "$src" "$dst"
               fi ;;
        *) command -v zstd >/dev/null 2>&1 || { _p_warn "zstd absent"; return 1; }
           _p_say "compression de $src (fichier creux : seuls les blocs ecrits comptent)..."
           zstd -T0 -3 --sparse -f -o "$dst" "$src" || return 1 ;;
    esac
    chmod a+r "$dst"
    _p_ok "image de reference : $dst ($(du -h "$dst" | cut -f1))"
}

osmo_pmos_install() {
    local layout=0 launchers=0 kernel=0 image="" a prev=""
    for a in "$@"; do
        if [ "$prev" = "--image" ]; then image="$a"; prev=""; continue; fi
        case "$a" in
            --layout) layout=1 ;; --launchers) launchers=1 ;; --kernel) kernel=1 ;; --image) prev="--image" ;;
            *) echo "osmo-pmos-install : option inconnue $a" >&2; return 2 ;;
        esac
    done
    if [ $((layout + launchers + kernel)) -eq 0 ] && [ -z "$image" ]; then layout=1; launchers=1; kernel=1; fi
    [ "$layout" = 1 ] && pmos_layout
    [ "$kernel" = 1 ] && pmos_kernel
    [ "$launchers" = 1 ] && pmos_launchers
    [ -n "$image" ] && pmos_image "$image"
    return 0
}

if [ "${BASH_SOURCE[0]}" = "$0" ]; then
    [ "$(id -u)" -eq 0 ] || { echo "osmo-pmos-install : root requis" >&2; exit 1; }
    osmo_pmos_install "$@"
fi
```

3.427 /opt/GSM/osmo-operator/tools/osmo-pmos-qemu.sh

24616 octets, 414 lignes → 414 lignes

```
#!/bin/bash
# osmo-pmos-qemu - postmarketOS en QEMU, avec le modem du banc et sa voix.
#
# [2026-09-07] ON NE RECONSTRUIT PLUS LA LIGNE DE COMMANDE QEMU. Ce script
# refaisait a la main ce que fait « pmbootstrap qemu », et c est de la que
# venait le bug graphique : la carte et l affichage choisis ici - virtio-vga
# avec SDL force en rendu logiciel - couvraient l ecran de paves de couleur.
# « pmbootstrap qemu » ne les a pas : il prend virtio-vga-gl et « gtk,gl=on ».
# Repasser par le rendu OpenGL de SDL ne servait a rien non plus, le chroot
# musl n a pas les pilotes de la NVIDIA de l hote (« libEGL warning: egl:
# failed to create dri2 screen » en boucle). On appelle donc pmbootstrap.
#
# Ce que pmbootstrap ne savait pas faire est desormais DEDANS, dans
# pmb/commands/qemu.py, fonction osmo_bench_args() :
#   - le modem du banc sur un port serie PCI (ni ISA, ni USB : le firmware
#     UEFI prend les reponses d un port ISA pour des touches et n amorce
#     jamais, et ce noyau n a pas ftdi_sio pour l USB). 8250_pci est integre :
#     le port apparait tout seul en /dev/ttyS1, ce que ModemManager sonde ;
#   - la carte son cablee sur le pont voix du banc plutot que sur les
#     peripheriques par defaut : gsm_audio porte le descendant (le RTP
#     d osmo-mgw), son monitor alimente donc la CAPTURE de la VM ; gsm_mic
#     porte le montant, il recoit donc ce que la VM JOUE.
#
# QEMU ne fait qu ECOUTER sur le port du modem : c est osmo-phonesim-banc.py
```

```
# qui vient s y brancher, et seulement une fois le firmware passe - sinon il
# n amorce pas. C est le role d osmo-pmos-setup, a relancer apres chaque
# redemarrage du banc : start-native.sh tue le modem de la VM au passage.
#
# L affichage reste SDL, mais celui que pmbootstrap monte : virtio-vga-gl et
# « sdl,gl=on ». Ce n est pas SDL qui donnait les paves, c est le virtio-vga
# sans GL pousse dans un rendu logiciel force. OSMO_PMOS_DISPLAY=gtk pour
# passer a GTK, =none pour une VM sans fenetre.
#
# Reglages : OSMO_PMOS_AT_PORT (12346), OSMO_PMOS_SINK (gsm_mic),
# OSMO_PMOS_SOURCE (gsm_audio.monitor), OSMO_PMOS_MEM (4096),
# OSMO_PMOS_DISPLAY (sdl), OSMO_PMOS_DISK (16G), OSMO_PMOS_MODEM (1 : le modem et la voix sont
# branches tout seuls des que la VM repond ; 0 : VM nue).
# [2026-09-07] LE MOT DE PASSE, DEMANDE D ABORD. pmbootstrap a besoin de root
# la ou l ancien script n en avait aucun besoin : il detache les boucles
# (« sudo losetup --json --list »), monte le chroot, y installe OVMF. Lance
# depuis l icone, personne n est la pour repondre a sudo : la commande echoue
# aussitot et il ne se passe RIEN. On demande donc le mot de passe en premier,
# avec un vrai terminal (d ou « Terminal=true » dans osmo-pmos.desktop), et on
# garde la fenetre ouverte si ca casse - sinon l erreur disparait avec elle.
# [2026-09-07] LE FORMAT, AU CHOIX ET AU LANCEMENT. Phosh se met en page
# d apres l ecran : haut et etroit, c est un smartphone ; large, une
# tablette. C est la carte virtio de QEMU qui porte cette taille (xres/yres,
# poses par le patch pmbootstrap depuis OSMO_PMOS_RES), et le systeme demarre
# dedans - on ne change pas de format sous une session ouverte. Le format
# se dit en premier argument (les icones du bureau passent par la variable) :
#   osmo-pmos-qemu smartphone      720x1440   (le defaut)
#   osmo-pmos-qemu tablette        1280x800
#   osmo-pmos-qemu 1024x768        n importe quelle taille <largeur>x<hauteur>
# [2026-09-08] STOP, ET UNE SEULE VM A LA FOIS. Il n y avait pas de moyen
# d arreter le telephone autrement qu en fermant sa fenetre, et relancer
# l icone pendant qu il tourne finissait en « pmbootstrap a echoue (1) » :
# QEMU refuse l image deja ouverte et les ports 2222 / 12346 deja pris, et
# son message part sur la console, pas dans le journal. Donc :
#   osmo-pmos-qemu stop     eteint proprement (poweroff par SSH, 30 s), sinon
#                           SIGTERM puis SIGKILL sur QEMU ; debranche le modem
#                           du banc (osmo-phonesim-banc.py --connect), que
#                           osmo-pmos-setup rebranche au prochain demarrage.
#   osmo-pmos-qemu status   dit si la VM tourne (pid) et si son SSH repond.
# Et au lancement, si une VM tourne deja, on le dit et on s arrete la.
PMOS_SSH_PORT="${OSMO_PMOS_SSH_PORT:-2222}"
PMOS_SSH="sshpass -p ${OSMO_PMOS_PASS:-147147} ssh -p $PMOS_SSH_PORT -T -o StrictHostKeyChecking=no -o
 ↪  UserKnownHostsFile=/dev/null -o LogLevel=ERROR -o ConnectTimeout=4 -o PreferredAuthentications=password
 ↪  ${OSMO_PMOS_USER:-user}@127.0.0.1"
# [2026-09-09] lib/audio.sh : l affectation des haut-parleurs et du micro (une
# seule source de verite, voir « L AFFECTATION DES HAUT-PARLEURS » la-bas).
# Ici : l annuleur d echo est charge AVANT QEMU, pour que la carte « combine »
# de la VM naisse directement sur lui (OSMO_PMOS_HP / OSMO_PMOS_MIC, lus par
# le patch pmbootstrap) ; les bouclages directs de l hote sont retires quand
# la VM part (la voix passe par elle) et REPOSES quand elle s arrete.
AUDIO_LIB="${OSMO_REPO:-/opt/GSM/osmo-operator}/lib/audio.sh"
if [ -f "$AUDIO_LIB" ]; then
    # shellcheck source=../lib/audio.sh
    . "$AUDIO_LIB"
else
    AUDIO_LIB=""
fi
# Le vrai QEMU (pas le « sh -c » qui l enveloppe) : celui qui tient l image.
pmos_qemu_pids() {
    pgrep -f '^[^ ]*ld-musl[^ ]* .*qemu-system-x86_64 .*rootfs/qemu-amd64\.img' 2>/dev/null
    pgrep -f '^[^ ]*qemu-system-x86_64 .*rootfs/qemu-amd64\.img' 2>/dev/null
}
pmos_ssh_ok() { $PMOS_SSH 'echo ok' 2>/dev/null | grep -q ok; }
pmos_status() {
    local pids; pids="$(pmos_qemu_pids | sort -u | tr '\n' ' ')"
    if [ -z "$pids" ]; then echo "osmo-pmos-qemu: aucune VM postmarketOS ne tourne"; return 1; fi
```

```
    echo "osmo-pmos-qemu: VM postmarketOS en marche (QEMU pid $pids)"
    pmos_ssh_ok && echo "osmo-pmos-qemu: SSH repond sur le port $PMOS_SSH_PORT" \
                || echo "osmo-pmos-qemu: SSH ne repond pas encore (port $PMOS_SSH_PORT)"
    pgrep -f '[o]smo-phonesim-banc.py --connect' >/dev/null && echo "osmo-pmos-qemu: modem du banc branche" \
                                                            || echo "osmo-pmos-qemu: modem du banc NON branche"
    return 0
}
pmos_stop() {
    local pids i
    pids="$(pmos_qemu_pids | sort -u | tr '\n' ' ')"
    if [ -z "$pids" ]; then
        echo "osmo-pmos-qemu: aucune VM postmarketOS a arreter"
    else
        if pmos_ssh_ok; then
            echo "osmo-pmos-qemu: extinction propre par SSH (poweroff)..."
            $PMOS_SSH "echo ${OSMO_PMOS_PASS:-147147} | sudo -S -p '' poweroff" >/dev/null 2>&1
            for i in $(seq 1 30); do
                sleep 1
                [ -z "$(pmos_qemu_pids)" ] && break
            done
        fi
        if [ -n "$(pmos_qemu_pids)" ]; then
            echo "osmo-pmos-qemu: QEMU encore la, SIGTERM (pid $pids)"
            # shellcheck disable=SC2086
            kill -TERM $pids 2>/dev/null
            for i in $(seq 1 5); do sleep 1; [ -z "$(pmos_qemu_pids)" ] && break; done
        fi
        if [ -n "$(pmos_qemu_pids)" ]; then
            echo "osmo-pmos-qemu: QEMU ne meurt pas, SIGKILL"
            # shellcheck disable=SC2086
            kill -KILL $(pmos_qemu_pids) 2>/dev/null
            sleep 1
        fi
        [ -z "$(pmos_qemu_pids)" ] && echo "osmo-pmos-qemu: VM arretee" \
                                   || { echo "osmo-pmos-qemu: la VM tourne toujours (droits ? relancer en root)";
                                    ↪  return 1; }
    fi
    # Le modem du banc etait branche sur le port serie de CETTE VM : on le
    # debranche, osmo-pmos-setup le rebranchera au prochain demarrage.
    # Le banc tourne en root depuis osmo-pmos-setup (le tun PPP l exige) :
    # sous le compte de session, pkill ne peut pas, sudo -n si le mot de
    # passe est encore en cache.
    if pkill -f '[o]smo-phonesim-banc.py --connect' 2>/dev/null \
       || sudo -n pkill -f '[o]smo-phonesim-banc.py --connect' 2>/dev/null; then
        echo "osmo-pmos-qemu: modem du banc debranche"
    fi
    # Un « pmbootstrap qemu » orphelin (VM tuee sous lui) ne sert plus a rien.
    # Motif ancre sur l interpreteur : un « pkill -f » large tuerait aussi le
    # terminal de quiconque a ces mots dans sa ligne de commande.
    pkill -f '^[^ ]*python[0-9.]* [^ ]*pmbootstrap qemu' 2>/dev/null || true
    # La VM partie, l hote reprend son ecoute directe (gsm_audio -> ses HP).
    [ -n "$AUDIO_LIB" ] && { ensure_local_loopback; ensure_local_mic; }
    return 0
}
# Depuis une icone (Terminal=true), la fenetre se fermerait avant qu on ait
# lu : on la garde quelques secondes quand on est sur un vrai terminal.
pmos_fin() { local rc=$1; [ -t 0 ] && read -r -t 8 -p "Entree pour fermer (8 s) " _; exit "$rc"; }
# [2026-09-08] BASCULE : un seul geste sur l icone. VM en marche -> on l eteint
# (pmos_stop) ; VM arretee -> on la lance, exactement comme sans argument.
# C est ce que fait l icone (osmo-pmos toggle -> ici) : plus besoin du clic
# droit, peu visible sous Phosh, pour arreter le telephone.
case "${1:-}" in
    stop|arret|arreter|off|down) pmos_stop; pmos_fin $? ;;
    status|etat)                 pmos_status; pmos_fin $? ;;
    toggle|bascule)
        if [ -n "$(pmos_qemu_pids)" ]; then
```

```
            echo "osmo-pmos-qemu: le telephone tourne - on l eteint"
            pmos_stop; pmos_fin $?
        fi
        echo "osmo-pmos-qemu: le telephone est arrete - on le lance"
        shift ;;
esac

case "${1:-}" in
    smartphone|telephone|phone) export OSMO_PMOS_RES=720x1440; shift ;;
    tablette|tablet)            export OSMO_PMOS_RES=1280x800; shift ;;
    [0-9]*x[0-9]*)              export OSMO_PMOS_RES="$1"; shift ;;
esac
export OSMO_PMOS_RES="${OSMO_PMOS_RES:-720x1440}"
echo "osmo-pmos-qemu: format ${OSMO_PMOS_RES} ($([ "$OSMO_PMOS_RES" = 1280x800 ] && echo tablette || echo
↪  smartphone))"
if [ -n "$(pmos_qemu_pids)" ]; then
    pmos_status
    echo "osmo-pmos-qemu: une seule VM a la fois - « osmo-pmos-qemu stop » pour l arreter, puis relancer"
    read -r -p "Entree pour fermer " _
    exit 1
fi

PMB="$(command -v pmbootstrap || echo "$HOME/.local/bin/pmbootstrap")"
if ! sudo -v; then
    echo "osmo-pmos-qemu: sudo refuse - pmbootstrap ne peut pas preparer l image" >&2
    read -r -p "Entree pour fermer " _
    exit 1
fi
# [2026-09-09] EN ROOT AUSSI. La session de l ISO s ouvre sur root (gdm3,
# iso_modules/84-comptes.sh) : l icone lancait « pmbootstrap qemu » en root,
# qui repond « Do not run pmbootstrap as root! » et s arrete - l ISO avait le
# telephone (pmbootstrap patche, noyau PPP, image de reference) et ne pouvait
# pas l allumer. pmbootstrap accepte --as-root : on le passe quand on est
# root, et le dossier de travail est alors celui de root.
PMB_OPTS=(); [ "$(id -u)" -eq 0 ] && PMB_OPTS=(--as-root)
# [2026-09-16] LA DATA DU TELEPHONE SORT VRAIMENT, A CHAQUE LANCEMENT. Le PPP
# montait (10.99.0.2), la passerelle repondait, et rien au-dela : docker pose
# la politique FORWARD a DROP sur l hote, et le paquet de l UE (ogstun, apres
# NAT) etait jete sans trace - NetworkManager restait « connecte (local) ».
# osmo-epc.sh (setup_forward) pose la regle au demarrage du coeur 4G ; on la
# repose ici, avant la VM, pour le cas ou le coeur est parti avant ce script
# ou ou docker a recree ses chaines entre-temps. DOCKER-USER : consultee
# avant les regles de docker, jamais reecrite par lui.
pmos_forward() {
    local su=(); [ "$(id -u)" -eq 0 ] || su=(sudo -n)
    "${su[@]}" iptables -N DOCKER-USER 2>/dev/null || true
    local spec
    for spec in "-i ogstun" "-o ogstun"; do
        # shellcheck disable=SC2086
        "${su[@]}" iptables -C DOCKER-USER $spec -j ACCEPT 2>/dev/null \
            || "${su[@]}" iptables -I DOCKER-USER 1 $spec -j ACCEPT 2>/dev/null \
            || { echo "osmo-pmos-qemu: FORWARD ($spec) refuse - la data du telephone s arretera a l hote" >&2;
             ↪  return 0; }
    done
    "${su[@]}" iptables -C FORWARD -j DOCKER-USER 2>/dev/null \
        || "${su[@]}" iptables -I FORWARD 1 -j DOCKER-USER 2>/dev/null || true
    echo "osmo-pmos-qemu: FORWARD ogstun autorise (DOCKER-USER) - la data du telephone sort vers Internet"
}
pmos_forward
# [2026-09-09] LE DOSSIER DE TRAVAIL EST CELUI DE LA CONFIG, plus ~/test en
# dur (celui de la machine de reference) : sur l ISO c est
# ~/.local/var/pmbootstrap (gabarit pose par 88-lte-pmos.sh dans /root et
# /home/osmocom, instancie par osmo-pmos-build). ~/test reste accepte.
PMB_WORK="${OSMO_PMB_WORK:-$(sed -n 's/^work *= *//p' "$HOME/.config/pmbootstrap_v3.cfg" 2>/dev/null | head -1)}"
[ -n "$PMB_WORK" ] || PMB_WORK="$HOME/.local/var/pmbootstrap"
[ -d "$PMB_WORK/chroot_native" ] || [ ! -d "$HOME/test/chroot_native" ] || PMB_WORK="$HOME/test"
```

```
IMG_PMB="$(ls -1 "$PMB_WORK"/chroot_native/home/pmos/rootfs/*.img 2>/dev/null | head -1)"
# [2026-09-09] PAS D IMAGE : ON LA FABRIQUE, ICI. Le premier clic sur l icone
# de l ISO tombait sur « pmbootstrap qemu » sans config, sans dossier de
# travail, avec l image de reference encore en .zst dans /opt/user_interface/
# pmos/image : echec, fenetre fermee. osmo-pmos-build fait tout cela (config
# depuis le gabarit, pmaports, noyau PPP, image decompressee) puis REVIENT ici
# (exec osmo-pmos-qemu, OSMO_PMOS_FROM_BUILD=1 pour ne pas boucler).
if [ -z "$IMG_PMB" ] && [ "${OSMO_PMOS_FROM_BUILD:-0}" != 1 ]; then
    BUILD="$(command -v osmo-pmos-build 2>/dev/null || echo /opt/user_interface/pmos/bin/osmo-pmos-build.sh)"
    if [ -x "$BUILD" ]; then
        echo "osmo-pmos-qemu: pas d image postmarketOS dans $PMB_WORK - premiere fois : osmo-pmos-build (long,
         ↪  reseau)"
        exec "$BUILD" "$([ "$OSMO_PMOS_RES" = 1280x800 ] && echo tablette || echo smartphone)"
    fi
    echo "osmo-pmos-qemu: pas d image postmarketOS dans $PMB_WORK, et pas d osmo-pmos-build : « pmbootstrap
     ↪  install » a la main" >&2
fi

# [2026-09-08] LE MODEM, PAR DEFAUT. Le telephone sans modem n est qu une VM :
# pas d appel, pas de SMS, pas de data - et le noyau a ete rebati EXPRES pour
# porter PPP sur ce modem (voir osmo-pmos-setup). On ne demande donc plus a
# l utilisateur de penser a « Brancher le modem et la voix » apres coup : un
# guetteur attend que le SSH de la VM reponde (donc que le firmware soit
# passe, cf. l entete : brancher plus tot bloque l amorcage) et lance
# osmo-pmos-setup - le modem du banc sur le port serie, puis la voix. Sa
# trace complete va dans un fichier ; seules les lignes utiles remontent
# ici, prefixees, au milieu de la console serie de la VM.
# OSMO_PMOS_MODEM=0 pour une VM nue (l ancien comportement).
SETUP="${OSMO_PMOS_SETUP:-/usr/local/bin/osmo-pmos-setup}"
[ -x "$SETUP" ] || SETUP="${OSMO_REPO:-/opt/GSM/osmo-operator}/tools/osmo-pmos-setup.sh"
SETUP_LOG="${XDG_RUNTIME_DIR:-/tmp}/osmo-pmos-setup.log"
# [2026-09-08] LA PLACE, VERIFIEE A CHAQUE LANCEMENT. Le guetteur tourne
# toujours (modem ou pas) : une fois la VM levee il regarde si la racine
# occupe bien tout le disque. Normalement l initramfs pmOS l a deja fait au
# boot (pmos.force-partition-resize sur la ligne du noyau) ; sinon - image
# refaite sans ce parametre - il etend lui-meme la partition 2 puis l ext4,
# a chaud, par SSH. Puis le modem si OSMO_PMOS_MODEM=1.
pmos_place() {
    # Le script cote VM voyage en base64 : trois niveaux de guillemets sinon.
    local b64; b64="$(base64 -w0 <<'PLACE'
set -e
d=$(blockdev --getsize64 /dev/vda); p=$(blockdev --getsize64 /dev/vda2)
s=$(sfdisk -d /dev/vda 2>/dev/null | sed -n 's/^.*vda2 : start= *\([0-9]*\).*/\1/p')
# de la place derriere la partition 2 (> 64 Mio) ? on l etend, a chaud
if [ $((d - s*512 - p)) -gt 67108864 ]; then
    echo "partition 2 : $((p/1048576)) Mio sur un disque de $((d/1048576)) Mio - extension"
    parted -f -s /dev/vda resizepart 2 100% >/dev/null 2>&1 \
        || echo ", +" | sfdisk --no-reread -N2 /dev/vda >/dev/null 2>&1 || true
    partprobe /dev/vda 2>/dev/null || true
fi
resize2fs /dev/vda2 2>&1 | grep -v '^resize2fs [0-9]' || true
df -h / | tail -1 | awk '{print "place : racine " $2 ", libre " $4 " (" $5 " pris)"}'
PLACE
)"
    $PMOS_SSH "echo ${OSMO_PMOS_PASS:-147147} | sudo -S -p '' sh -c \"\$(echo $b64 | base64 -d)\"" 2>/dev/null
}
GUETTEUR=""
(
    for _ in $(seq 1 120); do          # 10 min : l image peut etre a refaire
        sleep 5
        pmos_ssh_ok || continue
        echo "osmo-pmos-qemu: VM levee - verification de la place sur le disque"
        pmos_place | sed 's/^/osmo-pmos-qemu: /'
        if [ "${OSMO_PMOS_MODEM:-1}" = "1" ] && [ -x "$SETUP" ]; then
            echo "osmo-pmos-qemu: branchement du modem et de la voix ($SETUP)"
            "$SETUP" >"$SETUP_LOG" 2>&1
```

```
            grep -E 'modem|ATTENTION|descendant|montant|Modem|DNS' "$SETUP_LOG" \
                | sed 's/^/osmo-pmos-qemu: /'
            echo "osmo-pmos-qemu: trace complete dans $SETUP_LOG"
        elif [ -x "$SETUP" ]; then
            # [2026-09-09] SANS MODEM, LE SON QUAND MEME. La VM nue sortait
            # son audio sur la carte PONT (0x12 -> gsm_mic) : muette pour
            # l operateur. --voix ne fait que la partie son (combine par
            # defaut, bouclages, sourdines).
            echo "osmo-pmos-qemu: VM sans modem - le son seul ($SETUP --voix)"
            "$SETUP" --voix >"$SETUP_LOG" 2>&1
            grep -E 'ATTENTION|descendant|montant|sourdines' "$SETUP_LOG" | sed 's/^/osmo-pmos-qemu: /'
        fi
        exit 0
    done
    echo "osmo-pmos-qemu: la VM n a pas repondu en SSH en 10 min - place non verifiee, modem NON branche" \
         "(relancer osmo-pmos-setup a la main)"
) &
GUETTEUR=$!
if [ "${OSMO_PMOS_MODEM:-1}" = "1" ] && [ -x "$SETUP" ]; then
    echo "osmo-pmos-qemu: le modem du banc sera branche des que la VM repond (OSMO_PMOS_MODEM=0 pour l eviter)"
else
    echo "osmo-pmos-qemu: VM sans modem (OSMO_PMOS_MODEM=0 ou osmo-pmos-setup absent)"
fi
# La VM partie, le guetteur n a plus rien a attendre.
trap '[ -n "$GUETTEUR" ] && kill "$GUETTEUR" 2>/dev/null' EXIT

# [2026-09-08] LE DISQUE : 16 Go, pas 2. L image pmbootstrap fait 2 Go par
# defaut et etait PLEINE (2,2 Go a 100 %) apres apk add/upgrade : plus rien
# n ecrivait, ni dconf, ni NetworkManager, ni le journal. Un premier correctif
# a 4 Go n avait jamais ete deploye (le lanceur de /usr/local/bin datait
# d avant) : la VM tournait toujours sur 2,8 Go pleins. --image-size est
# passe A CHAQUE LANCEMENT : pmbootstrap ne fait qu un truncate (fichier
# creux, ne coute que ce qui est ecrit ; il refuse seulement de RETRECIR,
# d ou le calcul ci-dessous qui garde la taille courante si elle est plus
# grande), l initramfs pmOS etend la partition et l ext4 au demarrage, et le
# guetteur ci-dessus verifie le resultat. OSMO_PMOS_DISK pour une autre taille.
DISK="${OSMO_PMOS_DISK:-16G}"
# IMG_PMB : resolu plus haut, depuis le dossier de travail de la config.
if [ -n "$IMG_PMB" ]; then
    cur_m=$(( ($(stat -c %s "$IMG_PMB") + 1048575) / 1048576 ))
    case "$DISK" in *G) want_m=$(( ${DISK%G} * 1024 )) ;; *M) want_m=${DISK%M} ;; *) want_m=$cur_m ;; esac
    [ "$want_m" -lt "$cur_m" ] && DISK="${cur_m}M"
    echo "osmo-pmos-qemu: disque de la VM $DISK ($IMG_PMB, ${cur_m} Mio avant)"
fi
# [2026-09-08] ROBUSTE AUX HOTES RECALCITRANTS (AMD en tete). Le patch
# pmbootstrap coupe deja l OpenGL sur un GPU AMD et le KVM sans SVM ; mais
# une VM qui MEURT AUSSITOT (moins de 25 s, code non nul) sur un hote qu on
# n avait pas prevu, on ne la laisse pas mourir : on relance sans OpenGL,
# puis sans KVM, puis en fenetre GTK (SDL + Wayland pur). Chaque essai est
# annonce ; le dernier code d erreur est celui qu on garde. Une VM qui a
# tourne plus de 25 s puis s est arretee n est pas relancee : c est un arret.
# [2026-09-09] LES HAUT-PARLEURS DE LA VM, NOMMES AVANT SON DEPART. La carte
# « combine » (osmo_bench_args) prenait « les peripheriques par defaut » de
# l hote au moment ou QEMU se connectait - la carte brute, sans annuleur
# d echo, puisque celui-ci n arrivait qu avec osmo-pmos-setup, 30 a 90 s plus
# tard ; et le defaut memorise (osmo_hp_ec) n existait pas encore. Le flux
# tournait donc d abord sur les HP bruts, puis etait deplace. On charge
# l annuleur ICI, on retire les bouclages directs (la voix passe par la VM),
# et on donne les noms au patch pmbootstrap : la carte nait au bon endroit.
if [ -n "$AUDIO_LIB" ] && [ "${OSMO_PMOS_RELAI:-1}" = "1" ]; then
    ensure_echo_cancel 2>/dev/null | sed 's/^ *//; s/^/osmo-pmos-qemu: /'
    # [2026-09-09] ON NE COUPE PLUS L ECOUTE DE L HOTE AU LANCEMENT. La voix
    # ne traverse plus la VM par defaut (cf. lib/audio.sh, « QUI PORTE LA
    # VOIX ») : les bouclages directs SONT le trajet, et les retirer ici
    # rendait le banc muet pendant tout le demarrage du telephone.
    # OSMO_PMOS_VOIX_VM=1 : c est osmo-pmos-setup qui pose alors le marqueur
```

```
    # et retire ces bouclages, une fois la VM prete a les remplacer.
    if [ "${OSMO_PMOS_VOIX_VM:-0}" = "1" ]; then
        remove_direct_loopbacks 2>/dev/null | sed 's/^ *//; s/^/osmo-pmos-qemu: /'
    else
        { ensure_local_loopback; ensure_local_mic; } 2>/dev/null | sed 's/^ *//; s/^/osmo-pmos-qemu: /'
    fi
    _hp="$(audio_hp_sink)"; _mic="$(audio_mic_source)"
    [ -n "$_hp" ]  && export OSMO_PMOS_HP="${OSMO_PMOS_HP:-$_hp}"
    [ -n "$_mic" ] && export OSMO_PMOS_MIC="${OSMO_PMOS_MIC:-$_mic}"
    echo "osmo-pmos-qemu: carte « combine » de la VM -> HP ${OSMO_PMOS_HP:-defaut}, micro ${OSMO_PMOS_MIC:-defaut}"
fi
# [2026-10-15] LE QEMU DE L HOTE, PAS CELUI DU CHROOT. Le QEMU de pmbootstrap
# tourne dans un chroot musl qui n a pas les pilotes NVIDIA de la machine :
# meme avec virtio-vga-gl et gl=on, son EGL echoue et le rendu retombe en
# logiciel - l affichage casse. Celui de l hote (Ubuntu, glibc) charge
# libEGL_nvidia : verifie le 15/10 (SDL + gl=on, aucune erreur EGL). Il lui
# faut qemu-system-gui et qemu-system-modules-opengl (ui-sdl, ui-opengl,
# virtio-vga-gl). OSMO_PMOS_HOST_QEMU=0 revient au QEMU du chroot.
QEMU_MODDIR=/usr/lib/x86_64-linux-gnu/qemu
if [ -z "${OSMO_PMOS_HOST_QEMU:-}" ]; then
    if [ -e "$QEMU_MODDIR/ui-opengl.so" ] && [ -e "$QEMU_MODDIR/ui-sdl.so" ] && [ -e
     ↪  "$QEMU_MODDIR/hw-display-virtio-vga-gl.so" ]; then
        OSMO_PMOS_HOST_QEMU=1
    else
        OSMO_PMOS_HOST_QEMU=0
        echo "osmo-pmos-qemu: QEMU de l hote sans modules OpenGL (apt install qemu-system-gui
         ↪  qemu-system-modules-opengl) - QEMU du chroot"
    fi
fi
HOST_QEMU_OPT=(); [ "$OSMO_PMOS_HOST_QEMU" = 1 ] && HOST_QEMU_OPT=(--host-qemu)
_lance() {
    "$PMB" "${PMB_OPTS[@]}" qemu \
        "${HOST_QEMU_OPT[@]}" \
        --memory "${OSMO_PMOS_MEM:-4096}" \
        --image-size "$DISK" \
        --display "${OSMO_PMOS_DISPLAY:-sdl}" \
        "$@"
}
_mort_tot() { [ "$1" -ne 0 ] && [ $(( $(date +%s) - t0 )) -lt 25 ]; }
# [2026-10-15] L OPENGL D ABORD, ET JUSQU AU BOUT. Le premier repli etait
# « sans OpenGL » (virtio-vga en rendu logiciel) : c est exactement l affichage
# en paves, « tout moche », et il partait des que QEMU mourait dans les 25 s,
# quelle qu en soit la cause (sudo, image verrouillee, port pris). On essaie
# donc d abord GTK puis sans KVM, TOUJOURS avec OpenGL ; le rendu logiciel ne
# vient qu en dernier, et il le dit. OSMO_PMOS_GL=1 est pose d office (hote
# NVIDIA : pmbootstrap le couperait sinon dans certains cas).
export OSMO_PMOS_GL="${OSMO_PMOS_GL:-1}"
t0=$(date +%s); _lance "$@"; rc=$?
if _mort_tot "$rc" && [ "$OSMO_PMOS_HOST_QEMU" = 1 ]; then
    echo "osmo-pmos-qemu: le QEMU de l hote est mort aussitot ($rc) - relance avec celui du chroot
     ↪  (OSMO_PMOS_HOST_QEMU=0)"
    OSMO_PMOS_HOST_QEMU=0; HOST_QEMU_OPT=(); t0=$(date +%s); _lance "$@"; rc=$?
fi
if _mort_tot "$rc" && [ "${OSMO_PMOS_DISPLAY:-sdl}" = sdl ]; then
    echo "osmo-pmos-qemu: QEMU est mort aussitot ($rc) - relance en fenetre GTK, OpenGL garde
     ↪  (OSMO_PMOS_DISPLAY=gtk)"
    export OSMO_PMOS_DISPLAY=gtk; t0=$(date +%s); _lance "$@"; rc=$?
fi
if _mort_tot "$rc" && [ "${OSMO_PMOS_KVM:-}" != 0 ]; then
    echo "osmo-pmos-qemu: encore mort ($rc) - relance SANS KVM, OpenGL garde (OSMO_PMOS_KVM=0, lent)"
    export OSMO_PMOS_KVM=0; t0=$(date +%s); _lance "$@"; rc=$?
fi
if _mort_tot "$rc" && [ "$OSMO_PMOS_GL" != 0 ]; then
    echo "osmo-pmos-qemu: encore mort ($rc) - DERNIER RECOURS : sans OpenGL (OSMO_PMOS_GL=0) - l affichage sera en
     ↪  paves"
    export OSMO_PMOS_GL=0; t0=$(date +%s); _lance "$@"; rc=$?
```

```
fi
# La VM s est arretee (fenetre fermee, extinction depuis Phosh, echec) : le
# bouclage direct de l hote revient - sans lui, plus personne n ecoute gsm_audio.
[ -n "$AUDIO_LIB" ] && { ensure_local_loopback; ensure_local_mic; } 2>/dev/null | sed 's/^ *//;
 ↪  s/^/osmo-pmos-qemu: /'
[ "$rc" -eq 0 ] || read -r -p "pmbootstrap a echoue ($rc). Entree pour fermer " _
exit "$rc"
```

3.428 /opt/GSM/osmo-operator/tools/osmo-pmos-setup.sh

26446 octets, 450 lignes → 450 lignes

```
#!/bin/bash
# osmo-pmos-setup - prepare la VM postmarketOS pour le modem du banc.
#
# A lancer depuis l hote, la VM demarree (SSH sur 2222). Il applique, de facon
# PERSISTANTE :
#   - le DNS : postmarketOS passe par systemd-resolved, qui n a aucun serveur
#     en QEMU (d ou « pas de reseau » alors que l IP et la route sont bonnes) ;
#   - le pilote ftdi_sio : le modem du banc arrive en USB serie (QEMU emule un
#     FTDI 0403:6001) et sans ce module il n y a pas de /dev/ttyUSB0 ;
#   - la regle udev qui dit a ModemManager de sonder ce port, sinon il l ignore.
set -u
PORT="${OSMO_PMOS_SSH_PORT:-2222}"
PASS="${OSMO_PMOS_PASS:-147147}"
USER_VM="${OSMO_PMOS_USER:-user}"
ssh_vm() { sshpass -p "$PASS" ssh -p "$PORT" -T -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -o
 ↪  LogLevel=ERROR \
    -o ConnectTimeout=8 -o PreferredAuthentications=password "$USER_VM@127.0.0.1" "$@"; }
# [2026-09-09] lib/audio.sh : l affectation des haut-parleurs et du micro de
# l hote (annuleur d echo, bouclages directs) - une seule source de verite,
# partagee avec audio-chain.sh (boot), start-native et osmo-pmos-qemu.
AUDIO_LIB="${OSMO_REPO:-/opt/GSM/osmo-operator}/lib/audio.sh"
if [ -f "$AUDIO_LIB" ]; then
    # shellcheck source=../lib/audio.sh
    . "$AUDIO_LIB"
else
    AUDIO_LIB=""
fi
# [2026-09-09] LA VOIX EN FONCTION, ET SEULE SI ON VEUT (osmo-pmos-setup --voix).
# Sans modem (OSMO_PMOS_MODEM=0) rien ne basculait la VM sur sa carte
# « combine » : la sortie par defaut restait le PONT (0x12), dont la lecture
# part dans gsm_mic - le telephone semblait muet. osmo-pmos-qemu appelle
# --voix des que le SSH repond quand il ne branche pas le modem.
voix() {
# [2026-09-07] LA VOIX, ET POURQUOI IL FAUT DEUX CARTES SON. QEMU ne sait pas
# pousser du son dans le haut-parleur d un invite : d une carte emulee, la
# LECTURE part vers un sink de l hote et la CAPTURE vient d une source de
# l hote. Le descendant de l appel (gsm_audio) vit cote hote : il ne peut donc
# entrer dans la VM que par sa capture, jamais par son haut-parleur. Avec une
# seule carte la VM a le point de vue du RESEAU - elle entend le correspondant
# dans son micro - et Calls reste muet.
#
# La VM a donc deux cartes, posees par pmbootstrap (osmo_bench_args) :
#   0000:00:12.0  le PONT    : capture = gsm_audio.monitor (le descendant),
#                              lecture = gsm_mic (le montant) ;
#   0000:00:13.0  le COMBINE : les peripheriques par defaut de l hote, tes
#                              vrais haut-parleurs et ton vrai micro.
# On les croise DANS la VM : pont -> combine pour entendre, combine -> pont
# pour parler. La VM est alors reellement dans le trajet de la voix.
#
# Et on coupe les deux bouclages DIRECTS de l hote (gsm_audio.monitor vers les
# haut-parleurs, micro vers gsm_mic) : sans ca la voix arrive DEUX fois, une
# fois en direct et une fois par la VM avec sa latence en plus - un echo franc.
```

```
# OSMO_PMOS_RELAI=0 pour ne pas toucher a l audio.
if [ "${OSMO_PMOS_RELAI:-1}" = "1" ]; then
    echo "--- voix"
    # [2026-09-09] L AFFECTATION DES HP ET DU MICRO VIENT DE lib/audio.sh, seule
    # source de verite (voir « L AFFECTATION DES HAUT-PARLEURS » la-bas) : les
    # bouclages directs de l hote sont retires, l annuleur d echo pose (ou deja
    # la, osmo-pmos-qemu le charge avant QEMU) et devient le defaut, et les flux
    # QEMU « combine » vont sur lui - audio_hp_sink / audio_mic_source.
    HP_SINK=""; MIC_SRC=""
    if [ -n "$AUDIO_LIB" ]; then
        ensure_echo_cancel
        ensure_record_mix
        # [2026-09-09] LA VOIX RESTE SUR L HOTE (cf. lib/audio.sh, « QUI PORTE
        # LA VOIX »). OSMO_PMOS_VOIX_VM=1 la fait passer par la VM : on depose
        # alors le marqueur AVANT d appeler les fonctions, pour qu elles
        # retirent les bouclages directs au lieu de les reposer.
        if [ "${OSMO_PMOS_VOIX_VM:-0}" = "1" ]; then
            : > "$PMOS_VOIX_VM_MARQUEUR" 2>/dev/null
            remove_direct_loopbacks
            echo "  la voix passe PAR LA VM (OSMO_PMOS_VOIX_VM=1) - ~800 ms d aller-retour"
        else
            rm -f "$PMOS_VOIX_VM_MARQUEUR" 2>/dev/null
            ensure_local_loopback
            ensure_local_mic
            echo "  la voix reste sur l hote (40 ms) - OSMO_PMOS_VOIX_VM=1 pour la faire passer par la VM"
        fi
        HP_SINK="$(audio_hp_sink)"; MIC_SRC="$(audio_mic_source)"
        echo "  haut-parleurs de l operateur : ${HP_SINK:-aucun} ; micro : ${MIC_SRC:-aucun}"
    else
        echo "  ATTENTION : lib/audio.sh introuvable (OSMO_REPO ?) - l audio de l hote n est pas regle"
    fi

    # [2026-09-07] LES SOURDINES QUI REVIENNENT. Le descendant traverse quatre
    # potentiometres, et il suffit d UN sur muet pour que la voix disparaisse
    # sans qu aucun bouclage ne manque. Deux sont cote hote : le flux QEMU
    # « combine » (la sortie de la VM vers tes haut-parleurs) et le sink des
    # haut-parleurs. PulseAudio les MEMORISE par nom d application
    # (module-stream-restore) : une fois coupe, le flux revient coupe a chaque
    # lancement. On les rouvre donc a chaque mise en service.
    for i in $(pactl list sink-inputs short 2>/dev/null | cut -f1); do
        pactl set-sink-input-mute "$i" 0 2>/dev/null
        pactl set-sink-input-volume "$i" 100% 2>/dev/null
    done
    for o in $(pactl list source-outputs short 2>/dev/null | cut -f1); do
        pactl set-source-output-mute "$o" 0 2>/dev/null
    done
    # Les flux QEMU de la carte « combine » (media.name=combine) sur les HP et
    # le micro de l operateur - ils y sont deja quand osmo-pmos-qemu les a
    # nommes au lancement (OSMO_PMOS_HP / OSMO_PMOS_MIC), sinon on les y met.
    if [ -n "$MIC_SRC" ]; then
        for so in $(pactl list source-outputs 2>/dev/null | awk '/Source Output #/{id=$3} /media.name =
        ↪  "combine"/{print id}' | tr -d '#'); do
            pactl move-source-output "$so" "$MIC_SRC" 2>/dev/null
        done
    fi
    if [ -n "$HP_SINK" ]; then
        for si in $(pactl list sink-inputs 2>/dev/null | awk '/Sink Input #/{id=$3} /media.name = "combine"/{print
        ↪  id}' | tr -d '#'); do
            pactl move-sink-input "$si" "$HP_SINK" 2>/dev/null
        done
    fi
    pactl set-source-mute @DEFAULT_SOURCE@ 0 2>/dev/null
    pactl set-sink-mute @DEFAULT_SINK@ 0 2>/dev/null && echo "  sourdines de l hote levees (haut-parleur et micro)"
    # [2026-09-09] speech-dispatcher (module dummy) joue un message factice grave
    # et fort dans les haut-parleurs quand une appli demande une synthese vocale :
    # on l envoie dans le sink poubelle osmo_tts_off (cf. lib/audio.sh) ; stream-
```

```
    # restore s en souviendra pour les flux suivants.
    pactl list short sinks 2>/dev/null | grep -qw osmo_tts_off \
        || pactl load-module module-null-sink sink_name=osmo_tts_off format=s16le rate=8000 channels=1
         ↪  sink_properties=device.description=TTS_off >/dev/null 2>&1
    for si in $(pactl list sink-inputs 2>/dev/null | awk '/Sink Input #/{id=$3} /application.name =
     ↪  "speech-dispatcher/{print id}' | tr -d '#'); do
        pactl move-sink-input "$si" osmo_tts_off 2>/dev/null && echo "  flux speech-dispatcher detourne des
         ↪  haut-parleurs (osmo_tts_off)"
    done
    ssh_vm "VOIX_VM='${OSMO_PMOS_VOIX_VM:-0}'; "'
        pci() { pactl list "$1" short 2>/dev/null | grep "pci-0000_00_$2\.0" | grep -v monitor | head -n1 | cut
↪   -f2; }
        PONT_OUT=$(pci sinks 12);  PONT_IN=$(pci sources 12)
        COMB_OUT=$(pci sinks 13);  COMB_IN=$(pci sources 13)
        if [ -z "$PONT_IN" ] || [ -z "$COMB_OUT" ]; then
            echo "  ATTENTION : les deux cartes son ne sont pas la - VM lancee avant le patch ?"
            pactl list sinks short 2>/dev/null
            exit 0
        fi
        # Le combine devient le peripherique par defaut : sinon la sonnerie et
        # les notifications de Phosh partiraient dans le montant de l appel.
        pactl set-default-sink "$COMB_OUT"
        pactl set-default-source "$COMB_IN"
        # Les deux autres potentiometres du descendant sont ici : Phosh coupe
        # volontiers le combine (touche volume, mode silencieux) et la voix
        # meurt la, bouclages en place. On les rouvre.
        for c in "$PONT_OUT" "$COMB_OUT"; do
            pactl set-sink-mute "$c" 0 2>/dev/null
            pactl set-sink-volume "$c" 100% 2>/dev/null
        done
        for c in "$PONT_IN" "$COMB_IN"; do
            pactl set-source-mute "$c" 0 2>/dev/null
        done
        # [2026-09-09] IDEMPOTENT POUR DE VRAI. On rechargeait les deux
        # bouclages a chaque passage : relancer osmo-pmos-setup pendant un
        # appel coupait la voix une demi-seconde. Si les deux sont deja la,
        # epingles, avec les bons bouts, on n y touche pas.
        deja=$(pactl list short modules 2>/dev/null | grep -c "module-loopback.*sink_dont_move=true")
        if [ "$deja" = "2" ] \
           && pactl list short modules 2>/dev/null | grep -q "source=$PONT_IN.*sink=$COMB_OUT" \
           && pactl list short modules 2>/dev/null | grep -q "source=$COMB_IN.*sink=$PONT_OUT"; then
            echo "  bouclages deja en place et epingles - inchanges"
            exit 0
        fi
        # [2026-09-09] LE PULSE DE LA VM NE DOIT PLUS SUSPENDRE SES CARTES.
        # module-suspend-on-idle endort un sink des que plus personne ne joue :
        # QEMU FERME alors ses flux cote hote. Deux degats. Un, l hote ne voit
        # plus le telephone (il le reconnait desormais a son processus, cf.
        # lib/audio.sh) ; deux, chaque reveil renegocie la latence des deux
        # cartes emulees, et c est la que la voix casse - « ca marche les deux
        # premieres fois et apres non ». Nos bouclages tiennent les cartes
        # eveillees, mais pas avant qu ils soient poses : on retire le module.
        for m in $(pactl list modules short 2>/dev/null | grep module-suspend-on-idle | cut -f1); do
            pactl unload-module "$m" 2>/dev/null && echo "  mise en veille des cartes desactivee
↪   (module-suspend-on-idle)"
        done
        # [2026-09-09] LE CROISEMENT DES DEUX CARTES N EST PLUS LE DEFAUT.
        # Sans lui, la VM ne porte plus la voix : elle garde ses cartes pour la
        # sonnerie, les notifications et l application Appels, et l operateur
        # entend l appel par l hote (40 ms au lieu de ~800). On RETIRE donc les
        # bouclages qu une mise en service precedente aurait laisses, et on ne
        # repose que si l hote l a demande (OSMO_PMOS_VOIX_VM=1, passe ici par
        # la variable VOIX_VM).
        for m in $(pactl list modules short 2>/dev/null | grep module-loopback | cut -f1); do
            pactl unload-module "$m" 2>/dev/null
        done
```

```
        if [ "${VOIX_VM:-0}" != "1" ]; then
            echo "  cartes non croisees : la voix reste sur l hote"
            exit 0
        fi
        # Idempotent : on retire les bouclages precedents avant de reposer.
        # [2026-09-08] 200 ms et pas 40 : a 40 ms PulseAudio, dans la VM,
        # notait « Too many underruns, increasing latency » et la voix
        # craquait (echo du 600 brouille). Une VM n a pas la regularite d une
        # carte son ; 200 ms restent imperceptibles sur un appel.
        # [2026-09-09] adjust_time=0 ET 400 ms. Le mobile ecrit son audio par
        # trames de 20 ms au rythme du L1 : avec l ajustement de cadence par
        # defaut (adjust_time=10), module-loopback re-echantillonnait sans
        # cesse pour tenir ses 200 ms - mesure sur l hote : sauts de retard de
        # 46 ms en pleine parole, correlation de forme d onde 0,29 en appel
        # contre 0,87 en injection directe ; a l oreille « deux flux qui se
        # superposent », voix pourrie. Cadence fixe + 400 ms de tampon :
        # correlation 1,00 sur tout l appel, retard stable a 497 ms.
        for m in $(pactl list modules short 2>/dev/null | grep module-loopback | cut -f1); do
            pactl unload-module "$m" 2>/dev/null
        done
        # [2026-09-09] EPINGLES (dont_move) : callaudiod / Phosh changent le
        # peripherique par defaut A CHAQUE APPEL (journal : « card has no voice
        # profile and no usable sink ») et PulseAudio deplace alors tout flux
        # non epingle - les deux bouclages partaient ailleurs des le decroche.
        pactl load-module module-loopback source="$PONT_IN" sink="$COMB_OUT" latency_msec=400 adjust_time=0
↪   sink_dont_move=true source_dont_move=true >/dev/null \
            && echo "  descendant : pont -> combine (on entend)"
        pactl load-module module-loopback source="$COMB_IN" sink="$PONT_OUT" latency_msec=400 adjust_time=0
↪   sink_dont_move=true source_dont_move=true >/dev/null \
            && echo "  montant    : combine -> pont (on parle)"
    ' 2>&1 | tail -n 6
fi
}

# [2026-09-08] ON ATTEND LE SSH. Lance a la main pendant que la VM demarre
# encore, tout echouait (« n a pas acces ») : on patiente jusqu a 90 s.
for _ in $(seq 1 45); do
    ssh_vm 'echo ok' 2>/dev/null | grep -q ok && break
    [ "$_" -eq 1 ] && echo "  la VM ne repond pas encore en SSH (port $PORT), on attend..."
    sleep 2
done
ssh_vm 'echo ok' 2>/dev/null | grep -q ok || { echo "  ATTENTION : pas de SSH sur le port $PORT - la VM est-elle
↪   lancee ?"; exit 1; }
if [ "${1:-}" = "--voix" ] || [ "${1:-}" = "voix" ]; then voix; exit 0; fi

ssh_vm "echo $PASS | sudo -S -p '' sh -c '
    mkdir -p /etc/systemd/resolved.conf.d
    # DNSSEC=no : le DNS interne de QEMU (10.0.2.3) ne signe pas ses reponses,
    # et resolved refusait tout - « DNSSEC validation failed: no-signature ».
    printf \"[Resolve]\nDNS=10.0.2.3 1.1.1.1\nFallbackDNS=8.8.8.8\nDNSSEC=no\nDNSOverTLS=no\n\" \
        > /etc/systemd/resolved.conf.d/osmo.conf
    systemctl restart systemd-resolved 2>/dev/null

    # Le modem arrive sur un port serie PCI : 8250_pci est integre a ce noyau,
    # il n y a donc aucun module a charger (ftdi_sio, lui, n existe pas ici).

    mkdir -p /etc/udev/rules.d
    # [2026-09-08] DEUX PORTS, UN SEUL MODEM. hvc0 = commande, hvc1 = donnees
    # (ID_MM_PORT_TYPE_AT_PPP : c est la que ModemManager compose ATD*99), et
    # le meme ID_MM_PHYSDEV_UID pour que MM les regroupe en un modem au lieu
    # d en voir deux. ttyS[1-9] : le 16550 de OSMO_PMOS_SERIAL=pci.
    {
      printf \"ACTION==\\\"add|change\\\", SUBSYSTEM==\\\"tty\\\", KERNEL==\\\"hvc0\\\",
↪   ENV{ID_MM_DEVICE_PROCESS}=\\\"1\\\", ENV{ID_MM_PHYSDEV_UID}=\\\"osmo-banc\\\",
↪   ENV{ID_MM_PORT_TYPE_AT_PRIMARY}=\\\"1\\\"\n\"
```

```
      printf \"ACTION==\\\"add|change\\\", SUBSYSTEM==\\\"tty\\\", KERNEL==\\\"hvc1\\\",
↪  ENV{ID_MM_DEVICE_PROCESS}=\\\"1\\\", ENV{ID_MM_PHYSDEV_UID}=\\\"osmo-banc\\\",
↪  ENV{ID_MM_PORT_TYPE_AT_PPP}=\\\"1\\\"\n\"
      printf \"ACTION==\\\"add|change\\\", SUBSYSTEM==\\\"tty\\\", KERNEL==\\\"ttyS[1-9]\\\",
↪  ENV{ID_MM_DEVICE_PROCESS}=\\\"1\\\", ENV{ID_MM_TTY_BAUDRATE}=\\\"115200\\\"\n\"
    } > /etc/udev/rules.d/99-osmo-modem.rules
    udevadm control --reload 2>/dev/null
    udevadm trigger --subsystem-match=tty 2>/dev/null
    # [2026-09-08] Le port modem est un virtconsole (/dev/hvc0) : le
    # generateur getty de systemd ouvre une invite de login sur la premiere
    # console de virtualisation, et c est elle qui repondait au modem
    # (« pmos-gsm login: ATE0 »). On la masque, le port est au modem seul.
    systemctl mask --now serial-getty@hvc0.service getty@hvc0.service serial-getty@hvc1.service
↪  getty@hvc1.service >/dev/null 2>&1

    # [2026-09-07] LA DATA : pppd et le greffon PPP de NetworkManager. Sans
    # eux, ATD*99 aboutit a CONNECT puis NO CARRIER une seconde plus tard
    # (« PPP failed to start: libnm-ppp-plugin.so is not installed »). Le
    # noyau doit avoir PPP (build-pm.sh) ; les modules se chargent au boot.
    # [2026-09-08] Le filaire n a plus de route par defaut (voir le bloc
    # suivant) : pour apk, on la lui rend le temps de ce bloc, avec le DNS
    # de QEMU, et le service osmo-filaire la retire juste apres.
    ip route replace default via 10.0.2.2 dev eth0 2>/dev/null
    resolvectl dns eth0 10.0.2.3 2>/dev/null; resolvectl default-route eth0 yes 2>/dev/null
    apk info -e ppp networkmanager-ppp >/dev/null 2>&1 \
        || apk add ppp networkmanager-ppp 2>&1 | tail -n 1
    # networkmanager et networkmanager-ppp DOIVENT etre de la meme version :
    # NM cherche le greffon dans /usr/lib/NetworkManager/<sa version>/. apk add
    # du greffon avait pris 1.58 sur le depot alors que l image portait NM
    # 1.56 : « libnm-ppp-plugin.so is not installed », et pas de PPP.
    NMV=\$(apk info -v 2>/dev/null | grep -oE \"^networkmanager-[0-9.]+\" | head -n1 | sed
↪  \"s/^networkmanager-//\")
    NMP=\$(apk info -v 2>/dev/null | grep -oE \"^networkmanager-ppp-[0-9.]+\" | head -n1 | sed
↪  \"s/^networkmanager-ppp-//\")
    # TOUS les sous-paquets networkmanager-* (wwan, ppp, ...) : chacun pose son
    # greffon dans le repertoire de SA version ; un seul en retard, et NM ne
    # voit plus le modem (wwan) ou ne lance plus pppd (ppp).
    [ \"\$NMV\" = \"\$NMP\" ] || apk upgrade \$(apk info 2>/dev/null | grep -E \"^networkmanager\") ppp 2>&1 |
↪  tail -n 1
    # Le journal bornait a rien : 168 Mo sur un disque de 2 Go, plein.
    mkdir -p /etc/systemd/journald.conf.d
    printf \"[Journal]\nSystemMaxUse=48M\n\" > /etc/systemd/journald.conf.d/osmo.conf
    journalctl --vacuum-size=40M >/dev/null 2>&1; rm -rf /var/cache/apk/*
    printf \"ppp_generic\nppp_async\n\" > /etc/modules-load.d/osmo-ppp.conf
    modprobe ppp_async 2>/dev/null

    systemctl restart ModemManager 2>/dev/null
    sleep 3
    mmcli --scan-modems 2>/dev/null | tail -n 1
'" 2>&1 | tail -n 4

# [2026-09-08] LA DATA PAR NOTRE 4G SEULEMENT. La carte virtio de QEMU (le
# « filaire ») donnait au telephone une route par defaut et un DNS par le
# NAT de QEMU : Phosh affichait une prise reseau et tout passait par la, la
# 4G ne servait a rien. On ne peut pas retirer la carte de QEMU : c est par
# elle (hostfwd 2222) que ce script parle a la VM. On la rend donc INVISIBLE
# au telephone : NetworkManager ne la gere plus (driver virtio_net non gere),
# un petit service lui laisse juste son adresse de gestion 10.0.2.15 sans
# route par defaut ni DNS, resolved ne pointe plus sur le DNS de QEMU. Le
# telephone n a plus qu une sortie : la connexion mobile (APN srsapn, creee
# ici si elle manque, automatique), donc le PPP du banc, donc la radio.
# OSMO_PMOS_FILAIRE=1 remet le filaire (gere par NM, DNS de QEMU).
# Le basculement coupe une seconde l adresse par laquelle on est connecte :
# il tourne detache dans la VM, et on attend le retour du SSH.
FILAIRE="${OSMO_PMOS_FILAIRE:-0}"
{ echo "$PASS"; echo "FILAIRE=$FILAIRE"; cat <<'GUEST'
```

```
set -u
CONF=/etc/NetworkManager/conf.d/90-osmo-filaire.conf
SVC=/etc/systemd/system/osmo-filaire.service
BIN=/usr/local/bin/osmo-filaire
mkdir -p /etc/NetworkManager/conf.d /etc/systemd/resolved.conf.d
if [ "$FILAIRE" = 0 ]; then
    printf '[keyfile]\nunmanaged-devices=driver:virtio_net\n' > "$CONF"
    cat > "$BIN" <<'EOB'
#!/bin/sh
# osmo-filaire - la carte virtio de QEMU garde son adresse de gestion (SSH
# depuis l hote par hostfwd), sans route par defaut ni DNS : la data du
# telephone passe par la connexion mobile, donc par le PPP du banc.
for d in /sys/class/net/*; do
    n=${d##*/}
    drv=$(basename "$(readlink "$d/device/driver" 2>/dev/null)" 2>/dev/null)
    [ "$drv" = virtio_net ] || continue
    ip link set "$n" up
    ip addr replace 10.0.2.15/24 dev "$n"
    ip route del default dev "$n" 2>/dev/null
    resolvectl revert "$n" 2>/dev/null     # plus de DNS de QEMU par ce lien
done
exit 0
EOB
    chmod 755 "$BIN"
    cat > "$SVC" <<'EOS'
[Unit]
Description=osmo-filaire - carte virtio en adresse de gestion seule (data par la 4G)
After=NetworkManager.service
[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/usr/local/bin/osmo-filaire
[Install]
WantedBy=multi-user.target
EOS
    systemctl daemon-reload; systemctl enable osmo-filaire >/dev/null 2>&1
    printf '[Resolve]\nDNSSEC=no\nDNSOverTLS=no\n' > /etc/systemd/resolved.conf.d/osmo.conf
    # Un seul profil mobile : celui que l utilisateur a deja cree (srsapn)
    # sert tel quel, sinon on en cree un.
    GSM=$(nmcli -t -f NAME,TYPE con show 2>/dev/null | awk -F: '$2 == "gsm" {print $1; exit}')
    if [ -z "$GSM" ]; then
        nmcli con add type gsm ifname '*' con-name 'Osmocom 4G' apn srsapn \
            connection.autoconnect yes ipv4.route-metric 50 >/dev/null 2>&1 && GSM='Osmocom 4G'
    fi
    # NetworkManager est REDEMARRE : apk vient parfois de le mettre a jour avec
    # le greffon PPP, et le daemon encore en memoire cherche le greffon de SON
    # numero de version (« libnm-ppp-plugin.so is not installed » alors qu il
    # est la, dans le repertoire de la version neuve). Le fichier de
    # configuration etant pose, il repart avec le filaire non gere.
    # Detache par systemd-run : un simple « setsid ... & » meurt avec la
    # session SSH (logind de la VM tue les processus de l utilisateur a la
    # deconnexion), et rien ne se passait.
    systemd-run --quiet --collect --unit "osmo-filaire-bascule-$$" sh -c "
        sleep 1
        systemctl restart NetworkManager
        sleep 2
        /usr/local/bin/osmo-filaire
        systemctl restart systemd-resolved
        [ -n '$GSM' ] && nmcli con up '$GSM' >/dev/null 2>&1
        exit 0
    "
    echo "  filaire cache au telephone : adresse de gestion seule, data par la connexion mobile (APN srsapn)"
else
    rm -f "$CONF" "$BIN" "$SVC"; systemctl daemon-reload
    printf '[Resolve]\nDNS=10.0.2.3 1.1.1.1\nFallbackDNS=8.8.8.8\nDNSSEC=no\nDNSOverTLS=no\n' >
↪   /etc/systemd/resolved.conf.d/osmo.conf
```

```
    systemd-run --quiet --collect --unit "osmo-filaire-bascule-$$" sh -c '
        sleep 1
        systemctl restart NetworkManager
        systemctl restart systemd-resolved
    '
    echo "  filaire rendu au telephone (OSMO_PMOS_FILAIRE=1)"
fi
GUEST
} | ssh_vm "sudo -S -p '' sh -s" 2>/dev/null | tail -n 2
for _ in $(seq 1 20); do
    sleep 1
    ssh_vm 'echo ok' 2>/dev/null | grep -q ok && break
done
ssh_vm 'echo ok' 2>/dev/null | grep -q ok || echo "  ATTENTION : la VM ne repond plus en SSH apres le basculement
↪  reseau"

# Le modem du banc vient maintenant se brancher sur le port que QEMU tient
# ouvert. On ne le fait qu ICI, systeme demarre : branche des le debut, le
# firmware UEFI prend ses reponses pour des touches et n amorce jamais (voir
# l entete de osmo-pmos-qemu).
PORT_AT="${OSMO_PMOS_AT_PORT:-12346}"
REPO="${OSMO_REPO:-/opt/GSM/osmo-operator}"
# [2026-09-08] LE MODEM EN ROOT : la data du telephone est un lien PPP dont le
# banc fait un tun (/dev/net/tun + TUNSETIFF), qu il pose dans l espace reseau
# de srsUE avec un NAT vers tun_srsue. Tout cela exige root ; lance sous le
# compte de session, le banc repondait CONNECT mais « la data n a pas de
# sortie radio ». osmo-pmos-qemu a deja fait sudo -v dans ce terminal ; sinon
# on demande si on a un terminal, et on se rabat sur l utilisateur en le disant.
BANC_LOG="/tmp/osmo-phonesim-vm-$(id -un).log"
BANC_SUDO=""
if [ "$(id -u)" -ne 0 ]; then
    if sudo -n true 2>/dev/null || { [ -t 0 ] && sudo -v; }; then
        BANC_SUDO="sudo -n"
    else
        echo "  ATTENTION : pas de root pour le modem - la data PPP n aura pas de sortie radio"
    fi
fi
if pgrep -f "[o]smo-phonesim-banc.py --connect" >/dev/null 2>&1; then
    echo "  modem deja branche sur la VM"
else
    # La trace AT de CE modem dans son propre fichier : en root, les deux
    # instances du banc ecrivaient dans /var/log/osmo-at-ofono.log et leurs
    # dialogues se melangeaient (le sondage +CLCC de l hote au milieu des RING
    # de la VM).
    # sudo AVANT setsid : le cache du mot de passe est attache au terminal,
    # et setsid s en detache - « sudo » lance sous setsid redemandait le mot
    # de passe (« a terminal is required ») et le modem ne partait pas.
    # --data : le port de donnees (hvc1, chardev QEMU sur PORT_AT+1), voir le
    # modem a deux ports dans osmo-phonesim-banc.py.
    # [2026-10-24] LA SORTIE RADIO, LE BANC LA CHOISIT (_radio_choix de
    # osmo-phonesim-banc.py) : la 4G d abord (tun_srsue dans le netns ue1),
    # sinon le PDP GPRS du modem osmocom-bb (tun_modem dans msgprs). On
    # forcait ici OSMO_LTE_TUN=tun_modem OSMO_LTE_NETNS= : la branche 4G
    # etait coupee, et tun_modem cherche sur l hote (il vit dans msgprs) -
    # 4G attachee, telephone sans data. On ne redit plus rien.
    $BANC_SUDO setsid nohup env OSMO_AT_LOG_OFONO=/tmp/osmo-at-vm.log \
        OSMO_PPP_DNS="${OSMO_PPP_DNS:-192.168.1.254,1.1.1.1}" \
        "$REPO/tools/osmo-phonesim-banc.py" --connect "127.0.0.1:$PORT_AT" --data "127.0.0.1:$((PORT_AT + 1))" \
        >>"$BANC_LOG" 2>&1 </dev/null &
    sleep 3
    pgrep -f "[o]smo-phonesim-banc.py --connect" >/dev/null 2>&1 \
        && echo "  modem du banc branche sur la VM (port $PORT_AT)" \
        || echo "  ATTENTION : le modem ne s est pas branche (cf. $BANC_LOG)"
fi
# [2026-09-08] MODEMMANAGER SONDE APRES LE BRANCHEMENT. Le redemarrage de MM
# plus haut a lieu AVANT que le banc soit sur le port : la sonde de hvc0 tombe
```

```
# dans le vide, MM raye le port et ne le ressonde plus (« No modems were
# found » pour de bon). Un second redemarrage, le modem branche, et il le voit.
# Et NetworkManager APRES ModemManager : relance (par la bascule reseau plus
# haut) pendant que MM etait absent, NM notait « error creating ModemManager
# client » et ne voyait jamais le modem ensuite - pas de device gsm, pas de
# PPP, modem « disabled ». Detache par systemd-run : la relance de NM coupe
# la session SSH qui la demande.
ssh_vm "echo $PASS | sudo -S -p '' systemd-run --quiet --collect --unit osmo-mm-puis-nm sh -c 'systemctl restart
 ↪  ModemManager; sleep 6; systemctl restart NetworkManager'" >/dev/null 2>&1
for _ in $(seq 1 20); do
    sleep 1
    ssh_vm 'echo ok' 2>/dev/null | grep -q ok && break
done
sleep 8

voix

echo "--- etat"
# ModemManager vient d etre relance : il sonde le port pendant quelques
# secondes, et « No modems were found » lu trop tot faisait croire a une
# panne. On lui laisse jusqu a 40 s.
ssh_vm 'for i in $(seq 1 20); do mmcli -L 2>/dev/null | grep -q Modem/ && break; sleep 2; done
ls /dev/hvc0 /dev/hvc1 /dev/ttyS[1-9] 2>/dev/null | head -n 4
getent hosts alpinelinux.org >/dev/null 2>&1 && echo "DNS OK" || echo "DNS KO"
mmcli -L 2>/dev/null | tail -n 1
mmcli -m any 2>/dev/null | grep -E "state:|access tech|primary port|ports" | sed "s/^ *|//"' 2>&1 | tail -n 8
```

3.429 /opt/GSM/osmo-operator/tools/osmo-pmos.sh

21456 octets, 437 lignes → 437 lignes

```
#!/bin/bash
# ============================================================================
# osmo-pmos.sh - LE TELEPHONE DU BANC : UNE VM postmarketOS / Phosh.
#
# CE QUI A CHANGE, ET POURQUOI. Le telephone du banc etait un Android en
# conteneur (Waydroid). Android n a pas de pile radio dans ce cadre-la : il
# aurait fallu un rild et un RIL proprietaire pour esperer une VRAIE telephonie,
# et sans eux tout passait par un pont qui posait des NOTIFICATIONS - une
# imitation d appel, pas un appel. postmarketOS, lui, arrive avec ModemManager,
# Calls et Chatty : c est un vrai telephone Linux, et un vrai telephone se
# branche sur un MODEM. Le banc en a un - tools/osmo-phonesim-banc.py, qui parle
# 27.007 d un cote et pilote le banc de l autre (VTY d osmo-bsc, SMPP d osmo-msc,
# AMI d Asterisk). Il ne restait donc qu a poser le fil entre les deux.
#
#   Phosh / Calls / Chatty
#        ↕  D-Bus
#   ModemManager (dans la VM)
#        ↕  /dev/ttyS1  (port serie de la VM)
#   QEMU  -serial tcp:127.0.0.1:12346,server=on,wait=off
#        ↕  TCP
#   osmo-phonesim-banc.py --connect 127.0.0.1:12346   (le modem du banc)
#        ↕  VTY / SMPP / AMI
#   osmo-bsc · osmo-msc · Asterisk · le mobile osmocom-bb
#
# POURQUOI C EST NOUS QUI ALLONS VERS QEMU, et pas l inverse : le firmware UEFI
# de la VM lit tout port serie comme une console. Un modem deja bavard pendant
# le demarrage lui fait croire qu on interrompt le boot, et il s arrete sur son
# menu. QEMU ECOUTE donc (server=on,wait=off) et le modem vient s y connecter
# une fois le systeme leve - c est le mode --connect du modem, qui attend que le
# port SSH de la VM reponde avant de se brancher.
#
#   osmo-pmos up         le geste de l icone : la VM pmbootstrap (osmo-pmos-qemu),
#                        format smartphone|tablette en argument ; stop et status y vont aussi
```

```
#   osmo-pmos install    paquets qemu + l image postmarketOS + la preparation
#   osmo-pmos fetch      telecharge (ou reprend) l image et verifie son sha256
#   osmo-pmos provision  mot de passe, sshd, et la regle ModemManager du port AT
#   osmo-pmos start      demarre la VM ET le modem du banc qui va avec
#   osmo-pmos toggle     bascule : eteint la VM si elle tourne, la lance sinon (l icone)
#   osmo-pmos stop       arrete les deux
#   osmo-pmos status     ou en est chaque morceau
#   osmo-pmos shell      un shell dans la VM (ssh)
#   osmo-pmos mm         ce que ModemManager voit du modem, DANS la VM
#   osmo-pmos at         parler a la main au modem du banc (tools/at-cmd.py)
#   osmo-pmos log        la trace AT de la VM, en direct
#
# LE MOT DE PASSE EST 147147. C est celui des images officielles
# postmarketOS, et on le REPOSE a chaque provision pour que la cle live, le
# systeme installe et une image re-telechargee repondent tous pareil - un banc
# ou le mot de passe depend de l historique de la machine n est pas un banc.
# Il ouvre la session Phosh (l ecran de verrouillage), le compte « user » et
# le ssh. Ce n est pas un secret : c est un banc de demonstration.
# ============================================================================
set -u

REPO="${OSMO_REPO:-/opt/GSM/osmo-operator}"
VAR="${OSMO_PMOS_DIR:-/var/lib/osmo-pmos}"
IMG="$VAR/pmos.img"
RUNDIR="${OSMO_PMOS_RUN:-/run/osmo-pmos}"
PIDF="$RUNDIR/qemu.pid"
MDMPID="$RUNDIR/modem.pid"

# Le mot de passe du telephone. 147147 partout : ecran de verrouillage, compte
# « user », ssh. Modifiable pour une demonstration particuliere, mais ce n est
# pas le cas d usage - voir l entete.
PMOS_PASS="${OSMO_PMOS_PASSWORD:-147147}"
PMOS_USER="${OSMO_PMOS_USER:-user}"

# Le port serie de la VM (le modem), et le port SSH. Le second sert AUSSI de
# temoin : le modem ne se branche qu une fois qu il repond (cf. l entete).
AT_PORT="${OSMO_PMOS_AT_PORT:-12346}"
SSH_PORT="${OSMO_PMOS_SSH_PORT:-2222}"

# L image. Les images officielles vivent dans des repertoires dates ; on
# resout donc « le plus recent » a la volee, et tout est surchargeable :
#   OSMO_PMOS_IMG_URL   l URL exacte d une image .img.xz
#   OSMO_PMOS_IMG_FILE  une image DEJA telechargee (rien n est telecharge)
PMOS_BRANCHE="${OSMO_PMOS_BRANCHE:-v25.12}"
PMOS_APPAREIL="${OSMO_PMOS_APPAREIL:-generic-x86_64}"
PMOS_IHM="${OSMO_PMOS_IHM:-phosh}"
PMOS_BASE="https://images.postmarketos.org/bpo/$PMOS_BRANCHE/$PMOS_APPAREIL/$PMOS_IHM/"

RED='\033[0;31m'; GREEN='\033[0;32m'; YELLOW='\033[1;33m'
CYAN='\033[0;36m'; BOLD='\033[1m'; NC='\033[0m'
_say()  { echo -e "  ${CYAN}→${NC} $*"; }
_ok()   { echo -e "  ${GREEN}✓${NC} $*"; }
_warn() { echo -e "  ${YELLOW}!${NC} $*"; }
_err()  { echo -e "  ${RED}✗${NC} $*" >&2; }

_root() {
    [ "$(id -u)" -eq 0 ] && return 0
    _err "cette commande demande root : sudo osmo-pmos $*"
    return 1
}

# ── LES PAQUETS ──────────────────────────────────────────────────────────
# qemu-system-x86 (la VM), ovmf (le firmware UEFI : l image postmarketOS
# generic-x86_64 ne demarre pas en BIOS), xz-utils (l image est compressee),
# et sshpass pour « osmo-pmos shell » sans taper le mot de passe a chaque fois.
_pmos_paquets() {
```

```
    export DEBIAN_FRONTEND=noninteractive
    local p manque=()
    for p in qemu-system-x86 qemu-system-gui qemu-system-modules-opengl ovmf xz-utils curl sshpass; do
        dpkg -s "$p" >/dev/null 2>&1 || manque+=("$p")
    done
    [ "${#manque[@]}" -eq 0 ] && { _ok "paquets deja la (qemu, ovmf, xz, curl, sshpass)"; return 0; }
    _say "installation : ${manque[*]}"
    apt-get update -y >/dev/null 2>&1 || true
    for p in "${manque[@]}"; do
        apt-get install -y --no-install-recommends "$p" >/dev/null 2>&1 \
            && _ok "$p installe" || _warn "$p indisponible (apt) - ignore"
    done
}

_ovmf() {
    local f
    for f in /usr/share/OVMF/OVMF_CODE_4M.fd /usr/share/OVMF/OVMF_CODE.fd \
             /usr/share/ovmf/OVMF.fd /usr/share/qemu/OVMF.fd; do
        [ -f "$f" ] && { echo "$f"; return 0; }
    done
    return 1
}

# ── L IMAGE ──────────────────────────────────────────────────────────
# On resout le repertoire date le plus recent, on telecharge en reprenant
# (curl -C -), et on VERIFIE le sha256 publie a cote. Une image tronquee par
# une coupure de reseau donne une VM qui ne demarre pas, sans dire pourquoi.
_pmos_url() {
    [ -n "${OSMO_PMOS_IMG_URL:-}" ] && { echo "$OSMO_PMOS_IMG_URL"; return 0; }
    local index date fichier
    index="$(curl -fsSL "$PMOS_BASE" 2>/dev/null)" || return 1
    date="$(printf '%s' "$index" | grep -o 'href="[0-9]\{8\}-[0-9]\{4\}/"' \
            | sed 's/href="//; s|/"||' | sort -r | head -1)"
    [ -n "$date" ] || return 1
    index="$(curl -fsSL "$PMOS_BASE$date/" 2>/dev/null)" || return 1
    fichier="$(printf '%s' "$index" | grep -o 'href="[^"]*\.img\.xz"' \
               | sed 's/href="//; s/"//' | head -1)"
    [ -n "$fichier" ] || return 1
    echo "$PMOS_BASE$date/$fichier"
}

pmos_fetch() {
    _root fetch || return 1
    install -d "$VAR" || return 1
    if [ -n "${OSMO_PMOS_IMG_FILE:-}" ]; then
        [ -f "$OSMO_PMOS_IMG_FILE" ] || { _err "OSMO_PMOS_IMG_FILE : $OSMO_PMOS_IMG_FILE introuvable"; return 1; }
        _say "image fournie : $OSMO_PMOS_IMG_FILE"
        case "$OSMO_PMOS_IMG_FILE" in
            *.xz) xz -dc "$OSMO_PMOS_IMG_FILE" > "$IMG" ;;
            *)    cp -f "$OSMO_PMOS_IMG_FILE" "$IMG" ;;
        esac
        _ok "image posee : $IMG"
        return 0
    fi
    if [ -f "$IMG" ] && [ "${OSMO_PMOS_REFETCH:-0}" != "1" ]; then
        _ok "image deja la : $IMG ($(du -h "$IMG" | cut -f1)) - OSMO_PMOS_REFETCH=1 pour la reprendre"
        return 0
    fi
    local url; url="$(_pmos_url)" || { _err "index postmarketOS injoignable ($PMOS_BASE) - donnez
    ↪  OSMO_PMOS_IMG_URL ou OSMO_PMOS_IMG_FILE"; return 1; }
    local xzf="$VAR/$(basename "$url")"
    _say "telechargement : $url"
    _say "1,5 Go environ - la reprise est possible (relancez la commande)"
    curl -fL -C - -o "$xzf" "$url" || { _err "telechargement interrompu - relancez, il reprend"; return 1; }
    if curl -fsSL -o "$xzf.sha256" "$url.sha256" 2>/dev/null; then
        _say "verification sha256..."
```

```
        ( cd "$VAR" && sha256sum -c "$(basename "$xzf").sha256" >/dev/null 2>&1 ) \
            && _ok "sha256 conforme" \
            || { _err "sha256 NON conforme : image corrompue, on ne la deballe pas"; return 1; }
    else
        _warn "pas de sha256 publie - on continue sans verifier"
    fi
    _say "decompression (compter quelques minutes)..."
    xz -dc "$xzf" > "$IMG" || { _err "decompression impossible"; return 1; }
    # L image sort a la taille juste ; un telephone a besoin d un peu de place
    # pour ses journaux et ses messages. Le fichier est creux : ces 8 Gio ne
    # coutent rien tant qu ils ne servent pas.
    qemu-img resize -f raw "$IMG" 8G >/dev/null 2>&1 || true
    _ok "image prete : $IMG"
    [ "${OSMO_PMOS_GARDER_XZ:-0}" = "1" ] || rm -f "$xzf" "$xzf.sha256"
}

# ── LA PREPARATION DE L IMAGE ──────────────────────────────────────
# On monte la partition racine de l image sur l hote (losetup -P) et on y pose
# trois choses. C est plus sur qu un premier demarrage interactif : la VM est
# prete AVANT d etre allumee, et une image re-telechargee repasse par ici.
pmos_provision() {
    _root provision || return 1
    [ -f "$IMG" ] || { _err "pas d image : osmo-pmos fetch d abord"; return 1; }
    local loop mnt="$VAR/mnt" part
    loop="$(losetup -f --show -P "$IMG")" || { _err "losetup impossible"; return 1; }
    # La racine est la plus grosse partition de l image (la premiere est l ESP).
    part="$(lsblk -brno NAME,SIZE "$loop" | tail -n +2 | sort -k2 -n | tail -1 | cut -d' ' -f1)"
    [ -n "$part" ] || { losetup -d "$loop"; _err "aucune partition dans l image"; return 1; }
    install -d "$mnt"
    if ! mount "/dev/$part" "$mnt" 2>/dev/null; then
        losetup -d "$loop"
        _err "/dev/$part non montable (image chiffree ? partition inattendue)"
        return 1
    fi
    local rc=0
    if [ ! -d "$mnt/etc" ]; then
        _err "/dev/$part ne ressemble pas a la racine du systeme"
        rc=1
    else
        # 1. LE MOT DE PASSE, pose dans /etc/shadow. On ne chroote pas : l image
        #    est en musl/Alpine et son /bin/sh n a pas les memes bibliotheques
        #    que l hote. openssl fabrique le condensat SHA-512 que attend shadow.
        local hash
        hash="$(openssl passwd -6 "$PMOS_PASS" 2>/dev/null)"
        if [ -n "$hash" ] && [ -f "$mnt/etc/shadow" ]; then
            local u
            for u in root "$PMOS_USER"; do
                grep -q "^$u:" "$mnt/etc/shadow" || continue
                # Le champ 2 est le condensat ; « | » comme separateur de sed
                # parce qu un hash contient des « / ».
                awk -F: -v u="$u" -v h="$hash" 'BEGIN{OFS=":"} $1==u{$2=h} {print}' \
                    "$mnt/etc/shadow" > "$mnt/etc/shadow.osmo" \
                    && mv "$mnt/etc/shadow.osmo" "$mnt/etc/shadow"
            done
            chmod 640 "$mnt/etc/shadow" 2>/dev/null || true
            _ok "mot de passe ${BOLD}$PMOS_PASS${NC} pose pour root et $PMOS_USER"
        else
            _warn "mot de passe non pose (openssl absent ou /etc/shadow introuvable)"
        fi

        # 2. LE PORT AT VU PAR ModemManager. Sans cette regle, rien ne se
        #    passe et c est SILENCIEUX : ModemManager ne sonde pas un ttyS au
        #    hasard (sa politique par defaut est « strict » - il lui faut un
        #    marqueur udev), et l ecran de Phosh reste sur « pas de modem ».
        #    Le port serie qu ajoute QEMU est un pci-serial (vendeur 0x1b36,
        #    « Red Hat, Inc. ») : les ttyS0-3 de la carte mere, eux, n ont pas
```

```
        #    de parent PCI et ne sont donc pas touches par cette regle.
        install -d "$mnt/etc/udev/rules.d"
        cat > "$mnt/etc/udev/rules.d/77-osmo-modem-at.rules" <<'UDEV'
# Le modem du banc, presente a la VM comme un port serie PCI par QEMU.
# ID_MM_DEVICE_PROCESS : « sonde ce port » (ModemManager ignore sinon).
# ID_MM_PORT_TYPE_AT   : « c est un port de commandes », pas un port de donnees.
SUBSYSTEM=="tty", KERNEL=="ttyS*", ATTRS{vendor}=="0x1b36", \
    ENV{ID_MM_DEVICE_PROCESS}="1", ENV{ID_MM_PORT_TYPE_AT}="1", \
    ENV{ID_MM_TTY_BAUDRATE}="115200"
UDEV
        _ok "regle udev posee : ModemManager sondera le port AT du banc"

        # 3. LE SSH. Il sert de temoin (« la VM est levee »), de shell de
        #    depannage, et c est par lui que le modem sait quand se brancher.
        if [ -d "$mnt/etc/init.d" ] && [ -x "$mnt/etc/init.d/sshd" ]; then
            install -d "$mnt/etc/runlevels/default"
            ln -sf /etc/init.d/sshd "$mnt/etc/runlevels/default/sshd" 2>/dev/null || true
            _ok "sshd active au demarrage de la VM"
        else
            _warn "sshd absent de l image - « osmo-pmos shell » ne marchera pas"
        fi
    fi
    umount "$mnt" 2>/dev/null || true
    losetup -d "$loop" 2>/dev/null || true
    return $rc
}

# ── DEMARRER ──────────────────────────────────────────────────────────
pmos_start() {
    echo "[2/3] machine virtuelle"
    _root start || return 1
    [ -f "$IMG" ] || { _err "pas d image : osmo-pmos install d abord"; return 1; }
    install -d "$RUNDIR"
    if pmos_tourne; then _ok "la VM tourne deja (pid $(cat "$PIDF"))"; else
        local bios; bios="$(_ovmf)" || { _err "firmware UEFI introuvable (paquet ovmf) - l image generic-x86_64 ne
         ↪  demarre qu en UEFI"; return 1; }
        local accel=tcg
        [ -w /dev/kvm ] && accel=kvm
        [ "$accel" = tcg ] && _warn "pas de /dev/kvm : la VM tournera en emulation (lent mais fonctionnel)"
        # -serial tcp:...,server=on,wait=off : QEMU ECOUTE et n attend
        # personne. C est le modem qui viendra, une fois le systeme leve (voir
        # l entete). wait=on bloquerait le demarrage de la VM elle-meme.
        # Le format telephone (720x1440) n existe pas sur tous les QEMU : on
        # retombe sur la definition par defaut si le reglage est refuse.
        local gpu="virtio-vga"
        local res="${OSMO_PMOS_RES:-720x1440}"
        local xr="${res%x*}" yr="${res#*x}"
        local base=(
            qemu-system-x86_64
            -name osmo-pmos
            -machine q35 -accel "$accel" -m "${OSMO_PMOS_RAM:-2048}" -smp "${OSMO_PMOS_CPU:-2}"
            -bios "$bios"
            -drive "if=virtio,format=raw,file=$IMG"
            -device virtio-tablet-pci -device virtio-keyboard-pci
            -netdev "user,id=n0,hostfwd=tcp:127.0.0.1:$SSH_PORT-:22"
            -device virtio-net-pci,netdev=n0
            -device pci-serial,chardev=at
            -chardev "socket,id=at,host=127.0.0.1,port=$AT_PORT,server=on,wait=off"
            -display "${OSMO_PMOS_DISPLAY:-gtk}"
            -pidfile "$PIDF"
            -daemonize
        )
        if ! "${base[@]}" -device "$gpu,xres=$xr,yres=$yr" 2>"$RUNDIR/qemu.err"; then
            _warn "definition $res refusee par ce QEMU - on garde la sienne"
            "${base[@]}" -device "$gpu" 2>"$RUNDIR/qemu.err" || {
                _err "QEMU n a pas demarre :"; sed 's/^/      /' "$RUNDIR/qemu.err" >&2; return 1; }
```

```
        fi
        _ok "VM postmarketOS demarree (ssh 127.0.0.1:$SSH_PORT, modem 127.0.0.1:$AT_PORT)"
        _say "mot de passe : ${BOLD}$PMOS_PASS${NC} (ecran de verrouillage, compte $PMOS_USER, ssh)"
    fi
    pmos_modem_start
}

# Le modem du banc, cote hote. Mode --connect : il attend que le SSH de la VM
# reponde, puis se branche sur le port serie que QEMU tient ouvert.
pmos_modem_start() {
    echo "[3/3] modem du banc"
    if pgrep -f "osmo-phonesim-banc.py --connect 127.0.0.1:$AT_PORT" >/dev/null 2>&1; then
        _ok "le modem du banc est deja branche sur le port $AT_PORT"
        return 0
    fi
    # /usr/bin/python3 et non le shebang : lance depuis le banc, le PATH place
    # le venv en tete et l interprete qu on y trouve n a pas les liaisons dont
    # ces outils ont besoin (meme piege que dans start-native.sh).
    OSMO_VM_READY_PORT="$SSH_PORT" setsid /usr/bin/python3 \
        "$REPO/tools/osmo-phonesim-banc.py" --connect "127.0.0.1:$AT_PORT" \
        >>/tmp/osmo-phonesim-vm.log 2>&1 &
    echo $! > "$MDMPID"
    _ok "modem du banc lance (il se branchera des que la VM repondra en ssh)"
    _say "journal : /tmp/osmo-phonesim-vm.log ; trace AT : /var/log/osmo-at-ofono.log"
}

pmos_tourne() {
    [ -f "$PIDF" ] || return 1
    kill -0 "$(cat "$PIDF" 2>/dev/null)" 2>/dev/null
}

pmos_stop() {
    _root stop || return 1
    pkill -f "osmo-phonesim-banc.py --connect 127.0.0.1:$AT_PORT" 2>/dev/null \
        && _ok "modem du banc arrete" || true
    if pmos_tourne; then
        kill "$(cat "$PIDF")" 2>/dev/null
        sleep 2
        pmos_tourne && kill -9 "$(cat "$PIDF")" 2>/dev/null
        _ok "VM arretee"
    else
        _say "la VM ne tournait pas"
    fi
    rm -f "$PIDF" "$MDMPID"
}

pmos_status() {
    echo -e "${BOLD}postmarketOS / Phosh - le telephone du banc${NC}"
    if pmos_tourne; then _ok "VM : en marche (pid $(cat "$PIDF"))"
    else _warn "VM : arretee"; fi
    [ -f "$IMG" ] && _ok "image : $IMG ($(du -h "$IMG" 2>/dev/null | cut -f1))" \
                  || _warn "image : absente (osmo-pmos install)"
    if pgrep -f "osmo-phonesim-banc.py --connect 127.0.0.1:$AT_PORT" >/dev/null 2>&1; then
        _ok "modem du banc : lance"
    else
        _warn "modem du banc : arrete"
    fi
    if timeout 2 bash -c "</dev/tcp/127.0.0.1/$SSH_PORT" 2>/dev/null; then
        _ok "ssh de la VM : ouvert sur 127.0.0.1:$SSH_PORT (mot de passe $PMOS_PASS)"
    else
        _warn "ssh de la VM : ferme (la VM demarre encore ?)"
    fi
    if grep -q . /var/log/osmo-at-ofono.log 2>/dev/null; then
        echo -e "  ${CYAN}dernieres lignes du dialogue AT :${NC}"
        tail -6 /var/log/osmo-at-ofono.log | sed 's/^/      /'
    fi
```

```
}

pmos_shell() {
    command -v sshpass >/dev/null 2>&1 \
        && exec sshpass -p "$PMOS_PASS" ssh -o StrictHostKeyChecking=no \
                -o UserKnownHostsFile=/dev/null -p "$SSH_PORT" "$PMOS_USER@127.0.0.1" "$@"
    exec ssh -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null \
             -p "$SSH_PORT" "$PMOS_USER@127.0.0.1" "$@"
}

# Ce que ModemManager voit, vu de la VM : le seul endroit ou l on sait si le
# modem du banc a ete reconnu, et sous quel numero.
pmos_mm() {
    pmos_shell "sudo mmcli -L; echo; sudo mmcli -m 0 2>/dev/null || true"
}

pmos_install() {
    echo "[1/3] image postmarketOS"
    _root install || return 1
    _pmos_paquets
    pmos_fetch     || return 1
    pmos_provision || return 1
    echo
    _ok "${BOLD}pret${NC} - « osmo-pmos start » allume le telephone"
    _say "mot de passe : ${BOLD}$PMOS_PASS${NC}"
}

# [2026-09-08] LE GESTE DE L ICONE VA A pmbootstrap. L icone du dock
# (osmo-pmos.desktop) et osmo-launcher.py appellent « osmo-pmos up » ; or la
# VM du banc est desormais celle de pmbootstrap (tools/osmo-pmos-qemu.sh :
# modem sur port serie PCI, cartes son du pont voix, format d ecran), pas
# l image telechargee ici dans /var/lib/osmo-pmos. « up » finissait donc sur
# « il faut d abord poser l image... cette commande demande root », fenetre
# fermee aussitot. up/start, stop/down et status vont au lanceur pmbootstrap
# des qu il est la ; le reste (fetch, provision, shell, mm, at, log) garde
# son sens ici.
QEMU_LANCEUR="${OSMO_PMOS_QEMU:-/usr/local/bin/osmo-pmos-qemu}"
[ -x "$QEMU_LANCEUR" ] || QEMU_LANCEUR="$REPO/tools/osmo-pmos-qemu.sh"
if [ -x "$QEMU_LANCEUR" ]; then
    case "${1:-status}" in
        up|start)  shift; exec "$QEMU_LANCEUR" "$@" ;;
        toggle|bascule) shift; exec "$QEMU_LANCEUR" toggle "$@" ;;
        stop|down) exec "$QEMU_LANCEUR" stop ;;
        status)    exec "$QEMU_LANCEUR" status ;;
    esac
fi
case "${1:-status}" in
    install)   pmos_install ;;
    # « up » = le geste de l icone : on installe si l image manque, puis on
    # allume. L image (1,5 Go a telecharger) n est PAS dans l ISO - elle
    # tiendrait mal dans un squashfs qui doit vivre en RAM - et le premier
    # lancement est donc long, une fois. Les paquets, eux, sont deja dans
    # l image (tools/osmo-extras-install.sh les pose au build).
    up)
        # Les reperes « [n/3] » imprimes par les fonctions font la barre de
        # progression de la fenetre de lancement (osmo-gtk-run.py --progress).
        if [ ! -f "$IMG" ]; then
            _say "premier lancement : il faut d abord poser l image postmarketOS"
            pmos_install || exit 1
        fi
        pmos_start
        ;;
    fetch)     pmos_fetch ;;
    provision) pmos_provision ;;
    start)     pmos_start ;;
    stop|down) pmos_stop ;;
```

```
    status)    pmos_status ;;
    shell)     shift; pmos_shell "$@" ;;
    mm)        pmos_mm ;;
    at)        shift; exec /usr/bin/python3 "$REPO/tools/at-cmd.py" "$@" ;;
    log)       exec tail -f /var/log/osmo-at-ofono.log ;;
    *)
        sed -n '/^#   osmo-pmos/,/^#   osmo-pmos log/p' "$0" | sed 's/^# \{0,2\}//'
        exit 2
        ;;
esac
```

3.430 /opt/GSM/osmo-operator/tools/osmo-reglages.py

20937 octets, 504 lignes → 504 lignes

```
#!/usr/bin/env python3
# [2026-10-21] Page web de reglage de la pile Osmocom du banc.
#
#   python3 tools/osmo-reglages.py [--port 8090] [--bind 127.0.0.1]
#   puis http://127.0.0.1:8090/
#
# La page est dans web/ (index.html, style.css, app.js) ; ce script la sert et fournit l'API /api/...
# Quatre onglets :
#   globals.conf : formulaire sur le modele de globals.conf.example (tools/globals-complet.py : reglages
#                  reseau + toutes les variables du lanceur), valeurs lues et reecrites dans globals.conf
#   Demons       : running-config de chaque demon lue par la VTY, chaque ligne modifiable
#                  en direct (configure terminal + chemin du noeud + ligne), « no », ajout,
#                  write memory, console VTY libre
#   Fichiers     : edition brute de configs/*.cfg et /etc/osmocom/*.cfg (sauvegarde)
#   Etat         : processus osmo-*, ports, compteurs utiles
#
# Rien n'est relance : les changements VTY sont immediats, globals.conf et les fichiers
# servent au prochain run. Ecoute sur 127.0.0.1 seulement : la console VTY execute tout.
# Bibliotheque standard uniquement.

import argparse, glob, html, json, os, re, shutil, socket, subprocess, time
from http.server import BaseHTTPRequestHandler, ThreadingHTTPServer
from urllib.parse import urlparse, parse_qs

RACINE = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
GLOBALS = os.path.join(RACINE, "globals.conf")
EXEMPLE = GLOBALS + ".example"
WEB = os.path.join(RACINE, "web")
TYPES = {".html": "text/html", ".css": "text/css", ".js": "application/javascript", ".svg": "image/svg+xml",
 ↪  ".png": "image/png"}
DOSSIERS_CFG = [os.path.join(RACINE, "configs"), "/etc/osmocom", "/root/.osmocom/bb"]
PROMPT = re.compile(rb"[\r\n][^\r\n]*[>#] ?$")


# ── VTY ──────────────────────────────────────────────────────────

class Vty:
    def __init__(self, hote, port, delai=4.0):
        self.s = socket.create_connection((hote, port), 2)
        self.s.settimeout(delai)
        self.lire()
        self.cmd("enable")

    def lire(self):
        b = b""
        while True:
            try:
                d = self.s.recv(65536)
            except socket.timeout:
```

```
                break
            if not d:
                break
            b += d
            b = re.sub(rb"\xff[\xfb-\xfe].", b"", b)  # negociation telnet (IAC WILL/WONT/DO/DONT)
            if PROMPT.search(b):
                break
        return b.decode("utf-8", "replace")

    def cmd(self, ligne):
        self.s.sendall(ligne.encode() + b"\r\n")
        r = self.lire().replace("\r", "")
        # retire l'echo et le prompt final
        l = r.split("\n")
        if l and l[0].strip() == ligne.strip():
            l = l[1:]
        if l and re.search(r"[>#] ?$", l[-1]):
            l = l[:-1]
        return "\n".join(l)

    def fermer(self):
        try:
            self.s.sendall(b"exit\r\n")
            self.s.close()
        except OSError:
            pass


_cache_vty = {"t": 0, "liste": []}


def demons_vty():
    """Ports d'ecoute des osmo-* et de mobile, sondes : la VTY repond par une banniere."""
    if time.time() - _cache_vty["t"] < 15:
        return _cache_vty["liste"]
    out = subprocess.run(["ss", "-ltnpH"], capture_output=True, text=True).stdout
    vus, liste = set(), []
    for l in out.splitlines():
        m = re.search(r"\s(\d+\.\d+\.\d+\.\d+):(\d+)\s.*users:\(\("([^"]+)",pid=(\d+)", l)
        if not m:
            continue
        hote, port, nom, pid = m.group(1), int(m.group(2)), m.group(3), m.group(4)
        if not (nom.startswith("osmo") or nom == "mobile") or (hote, port) in vus:
            continue
        vus.add((hote, port))
        try:
            s = socket.create_connection((hote, port), 0.5)
            s.settimeout(0.6)
            b = b""
            try:
                while len(b) < 4096:
                    d = s.recv(4096)
                    if not d:
                        break
                    b += d
                    if PROMPT.search(b):
                        break
            except socket.timeout:
                pass
            s.close()
        except OSError:
            continue
        t = b.decode("latin-1", "replace")
        if "VTY" not in t and not re.search(r"[>#] ?$", t):
            continue  # port CTRL / GSUP / autre
        invite = re.findall(r"([\w.-]+)[>#] ?$", t)
```

```
        liste.append({"hote": hote, "port": port, "nom": nom, "pid": pid,
                      "invite": invite[-1] if invite else nom})
    liste.sort(key=lambda d: (d["nom"], d["hote"], d["port"]))
    _cache_vty.update(t=time.time(), liste=liste)
    return liste


def arbre_config(texte):
    """running-config -> liste de lignes {txt, prof, chemin[], noeud}."""
    lignes = []
    for brut in texte.split("\n"):
        if not brut.strip() or brut.strip() == "!" or brut.startswith("Current configuration"):
            continue
        prof = len(brut) - len(brut.lstrip(" "))
        lignes.append([prof, brut.strip()])
    sortie, pile = [], []
    for i, (prof, txt) in enumerate(lignes):
        while pile and pile[-1][0] >= prof:
            pile.pop()
        noeud = i + 1 < len(lignes) and lignes[i + 1][0] > prof
        sortie.append({"txt": txt, "prof": prof, "chemin": [p[1] for p in pile], "noeud": noeud})
        if noeud:
            pile.append((prof, txt))
    return sortie


def appliquer(hote, port, chemin, lignes):
    v = Vty(hote, port)
    journal = []
    try:
        journal.append(("configure terminal", v.cmd("configure terminal")))
        for c in chemin:
            journal.append((c, v.cmd(c)))
        for l in lignes:
            journal.append((l, v.cmd(l)))
        journal.append(("end", v.cmd("end")))
    finally:
        v.fermer()
    erreur = any(r.lstrip().startswith("%") for _, r in journal)
    return {"ok": not erreur,
            "journal": "\n".join("> " + c + ("\n" + r if r.strip() else "") for c, r in journal)}


def console(hote, port, lignes):
    v = Vty(hote, port, delai=6.0)
    try:
        out = []
        for l in lignes:
            if l.strip():
                out.append("> " + l + "\n" + v.cmd(l))
        return "\n".join(out)
    finally:
        v.fermer()


# ── globals.conf ──────────────────────────────────────────

LIGNE_VAR = re.compile(r"^([A-Z_][A-Z0-9_]*)=(.*)$")


def _valeur(brut):
    v = brut.strip()
    if len(v) >= 2 and v[0] == v[-1] and v[0] in "\"'":
        v = v[1:-1]
    return v
```

```
def _valeurs(chemin):
    r = {}
    if os.path.exists(chemin):
        for l in open(chemin, encoding="utf-8"):
            m = LIGNE_VAR.match(l.rstrip("\n"))
            if m:
                r[m.group(1)] = _valeur(m.group(2))
    return r


def _patron():
    return EXEMPLE if os.path.exists(EXEMPLE) else GLOBALS


def lire_globals(patron=None, actuelles=None):
    """Les champs du patron (ordre, section, commentaire, defaut, choix) avec les valeurs actuelles."""
    if actuelles is None:
        actuelles = _valeurs(GLOBALS)
    champs, section, doc = [], "", []
    for l in open(patron or _patron(), encoding="utf-8"):
        l = l.rstrip("\n")
        m = re.match(r"#\s*──\s*(.+?)\s*─", l)
        if m:
            section, doc = m.group(1), []
            continue
        if l.startswith("#"):
            if section:
                doc.append(l.lstrip("# ").rstrip())
            continue
        m = LIGNE_VAR.match(l)
        if m:
            cle = m.group(1)
            defaut, choix = "", []
            for d in doc:
                md = re.match(r"defaut : (.*?)\s+\(\S+:\d+\)$", d)
                if md:
                    defaut = "" if md.group(1) == "(vide)" else md.group(1)
                mc = re.match(r"choix : (.*)$", d)
                if mc:
                    choix = [c.strip() for c in mc.group(1).split("|") if c.strip()]
            texte = " ".join(d for d in doc if not d.startswith("defaut :") and not d.startswith("choix :"))
            val = actuelles.get(cle, _valeur(m.group(2)))
            champs.append({"cle": cle, "val": val, "section": section, "doc": texte, "defaut": defaut, "choix":
↪   choix})
            doc = []
    # variables du fichier absentes du patron : gardees, en fin de liste
    connues = {c["cle"] for c in champs}
    for cle, val in actuelles.items():
        if cle not in connues:
            champs.append({"cle": cle, "val": val, "section": "Hors patron", "doc": "", "defaut": "", "choix": []})
    return champs


# ── instances (c54x_exe/run.sh, un fichier par Calypso) ──────

INSTANCES = os.path.join(RACINE, "instances")
PATRON_INST = os.path.join(INSTANCES, "instance.env.example")
ROLES = {1: "Instance 1 — BTS : TRX Calypso (--bts) ou mobile sur le pont (--bts-classique)",
         2: "Instance 2 — MS QEMU par l air (--bts --ms / --ms-magnetite)",
         3: "Instance 3 — 2e TRX Calypso, TS1 (--ts 2)"}
LIGNE_INST = re.compile(r'^: "\$\{([A-Z_][A-Z0-9_]*):=(.*)\}"$')


def _fichier_instance(n):
    return os.path.join(INSTANCES, "instance-%d.env" % n)
```

```
def _valeurs_instance(n):
    r = {}
    chemin = _fichier_instance(n)
    if os.path.exists(chemin):
        for l in open(chemin, encoding="utf-8"):
            m = LIGNE_INST.match(l.rstrip("\n"))
            if m:
                r[m.group(1)] = m.group(2)
    return r


def lire_instance(n):
    if not os.path.exists(PATRON_INST):
        return []
    return lire_globals(PATRON_INST, _valeurs_instance(n))


def ecrire_instance(n, nouvelles):
    """instance-N.env sur le modele du patron : `: "${VAR:=valeur}"` (l environnement garde la priorite)."""
    valeurs = _valeurs_instance(n)
    valeurs.update({k: str(v).replace('"', "").replace("\n", " ") for k, v in nouvelles.items()})
    sortie = ["# instances/instance-%d.env — %s\n" % (n, ROLES.get(n, "")),
              "# ecrit par la page web (tools/osmo-reglages.py) ; lu par c54x_exe/run.sh (INSTANCE=%d)\n" % n]
    vues = set()
    for l in open(PATRON_INST, encoding="utf-8"):
        m = LIGNE_VAR.match(l.rstrip("\n"))
        if m:
            cle = m.group(1)
            val = valeurs.get(cle, "")
            vues.add(cle)
            sortie.append(('#%s\n' % cle) if val == "" else ': "${%s:=%s}"\n' % (cle, val))
        elif not l.startswith("# instances/instance.env.example") and not l.startswith("# Genere par"):
            sortie.append(l if l.endswith("\n") else l + "\n")
    for k in valeurs:
        if k not in vues and valeurs[k] != "":
            sortie.append(': "${%s:=%s}"\n' % (k, valeurs[k]))
    os.makedirs(INSTANCES, exist_ok=True)
    sauvegarde(_fichier_instance(n))
    with open(_fichier_instance(n), "w", encoding="utf-8") as f:
        f.writelines(sortie)


def sauvegarde(chemin):
    if os.path.exists(chemin):
        shutil.copy2(chemin, chemin + ".bak-" + time.strftime("%Y%m%d-%H%M%S"))


def ecrire_globals(nouvelles):
    """Reecrit globals.conf sur le modele du patron : memes lignes et commentaires, valeurs actuelles +
     ↪  nouvelles."""
    valeurs = _valeurs(GLOBALS)
    valeurs.update({k: str(v).replace('"', "").replace("\n", " ") for k, v in nouvelles.items()})
    # [22:20] dans le bloc du lanceur (marqueurs de tools/globals-complet.py), une variable VIDE est ecrite
    # commentee : non definie, comme avant. Exportees vides par `set -a`, les 159 variables cassaient le lancement
    # (« Logs and working area (LOG_DIR non defini) », 21/10 22:11). Seules les valeurs saisies deviennent actives.
    sortie, vues, bloc = [], set(), False
    for l in open(_patron(), encoding="utf-8"):
        if l.startswith("# >>> VARIABLES DU LANCEUR"):
            bloc = True
        elif l.startswith("# <<< FIN DES VARIABLES DU LANCEUR"):
            bloc = False
        m = LIGNE_VAR.match(l.rstrip("\n"))
        if m:
            cle = m.group(1)
```

```
            val = valeurs.get(cle, "")
            vues.add(cle)
            sortie.append(('#%s=""\n' % cle) if (bloc and val == "") else '%s="%s"\n' % (cle, val))
        else:
            sortie.append(l if l.endswith("\n") else l + "\n")
    reste = [k for k in valeurs if k not in vues]
    if reste:
        sortie.append("\n# ── Hors patron (gardees de l ancien globals.conf) ──────────\n")
        sortie += ['%s="%s"\n' % (k, valeurs[k]) for k in reste]
    sauvegarde(GLOBALS)
    with open(GLOBALS, "w", encoding="utf-8") as f:
        f.writelines(sortie)


# ── Fichiers ──────────────────────────────────────────────

def fichiers_cfg():
    r = []
    for d in DOSSIERS_CFG:
        for p in sorted(glob.glob(os.path.join(d, "*.cfg")) + glob.glob(os.path.join(d, "*.conf"))):
            r.append(p)
    return r


def fichier_autorise(p):
    p = os.path.realpath(p)
    return p in [os.path.realpath(x) for x in fichiers_cfg()]


# [2026-10-22] Onglet « Configs osmo » : chaque /etc/osmocom/osmo-*.cfg, son gabarit configs/ (start.sh
# regenere /etc/osmocom depuis configs/ a chaque run : une retouche de /etc ne vit que jusque-la) et le
# demon qui le lit en ce moment (-c <fichier> dans sa ligne de commande).
PROTEGES = "/etc/osmocom/.proteges"   # lib/gabarits.sh install_configs_native n'ecrase pas ces fichiers


def proteges():
    try:
        with open(PROTEGES, encoding="utf-8") as f:
            return {l.split("#")[0].strip() for l in f} - {""}
    except FileNotFoundError:
        return set()


def proteger(nom, on):
    if "/" in nom or not nom:
        raise ValueError("nom invalide")
    if (nom in proteges()) == on:
        return
    lignes = []
    if os.path.exists(PROTEGES):
        with open(PROTEGES, encoding="utf-8") as f:
            lignes = [l for l in f if l.split("#")[0].strip() != nom]
    else:
        lignes = ["# Fichiers de /etc/osmocom que start.sh ne regenere pas depuis configs/ (un par ligne).\n"]
    if on:
        lignes.append(nom + "\n")
    with open(PROTEGES, "w", encoding="utf-8") as f:
        f.writelines(lignes)


def configs_osmo():
    cmds = subprocess.run(["ps", "-eo", "pid,args"], capture_output=True, text=True).stdout.splitlines()[1:]
    prot = proteges()
    r = []
    for p in sorted(glob.glob("/etc/osmocom/osmo-*.cfg")):
        nom = os.path.basename(p)
```

```
        gab = os.path.join(RACINE, "configs", nom)
        st = os.stat(p)
        pid = next((c.split(None, 1)[0] for c in cmds
                    if re.search(r"(^|\s)-c\s+" + re.escape(p) + r"(\s|$)", c.split(None, 1)[-1])), None)
        r.append({"nom": nom[:-4], "etc": p, "gabarit": gab if os.path.exists(gab) else None,
                  "pid": pid, "taille": st.st_size, "mtime": int(st.st_mtime), "protege": nom in prot})
    return r


# ── Etat ──────────────────────────────────────────────────────

def etat():
    ps = subprocess.run(["ps", "-eo", "pid,etimes,rss,pcpu,args", "--sort=args"],
                        capture_output=True, text=True).stdout.splitlines()
    procs = []
    for l in ps[1:]:
        c = l.split(None, 4)
        if len(c) == 5 and re.search(r"(^|/)(osmo-[\w-]+|mobile|qemu-system-arm|c54x_exe)\b", c[4]) \
                and "osmo-reglages" not in c[4]:
            procs.append({"pid": c[0], "age": int(c[1]), "rss": int(c[2]) // 1024,
                          "cpu": c[3], "cmd": c[4][:160]})
    return {"procs": procs, "vty": demons_vty()}


# ── HTTP ──────────────────────────────────────────────────────

class H(BaseHTTPRequestHandler):
    def log_message(self, *a):
        pass

    def envoyer(self, code, corps, type_="application/json"):
        if not isinstance(corps, bytes):
            corps = (json.dumps(corps) if type_ == "application/json" else corps).encode()
        self.send_response(code)
        self.send_header("Content-Type", type_ + "; charset=utf-8")
        self.send_header("Cache-Control", "no-store")
        self.send_header("Content-Length", str(len(corps)))
        self.end_headers()
        self.wfile.write(corps)

    def corps_json(self):
        n = int(self.headers.get("Content-Length") or 0)
        return json.loads(self.rfile.read(n) or b"{}")

    def do_GET(self):
        u = urlparse(self.path)
        q = {k: v[0] for k, v in parse_qs(u.query).items()}
        try:
            if u.path == "/" or not u.path.startswith("/api/"):
                nom = "index.html" if u.path == "/" else u.path.lstrip("/")
                chemin = os.path.realpath(os.path.join(WEB, nom))
                if not chemin.startswith(os.path.realpath(WEB) + os.sep) or not os.path.isfile(chemin):
                    return self.envoyer(404, {"err": "introuvable"})
                with open(chemin, "rb") as f:
                    return self.envoyer(200, f.read(), TYPES.get(os.path.splitext(chemin)[1],
                        ↪  "application/octet-stream"))
            if u.path == "/api/globals":
                return self.envoyer(200, lire_globals())
            if u.path == "/api/instances":
                return self.envoyer(200, [{"n": n, "role": r, "fichier": _fichier_instance(n),
                                           "existe": os.path.exists(_fichier_instance(n))} for n, r in ROLES.items()])
            if u.path == "/api/instance":
                return self.envoyer(200, lire_instance(int(q.get("n", "1"))))
            if u.path == "/api/demons":
                if q.get("rafraichir"):
                    _cache_vty["t"] = 0
```

```
                return self.envoyer(200, demons_vty())
            if u.path == "/api/config":
                v = Vty(q["hote"], int(q["port"]), delai=6.0)
                try:
                    t = v.cmd("show running-config")
                finally:
                    v.fermer()
                return self.envoyer(200, {"lignes": arbre_config(t), "brut": t})
            if u.path == "/api/fichiers":
                return self.envoyer(200, fichiers_cfg())
            if u.path == "/api/configs-osmo":
                return self.envoyer(200, configs_osmo())
            if u.path == "/api/fichier":
                if not fichier_autorise(q["p"]):
                    return self.envoyer(403, {"err": "fichier hors liste"})
                with open(q["p"], encoding="utf-8", errors="replace") as f:
                    return self.envoyer(200, {"texte": f.read()})
            if u.path == "/api/etat":
                return self.envoyer(200, etat())
            self.envoyer(404, {"err": "?"})
        except Exception as e:
            self.envoyer(500, {"err": "%s: %s" % (type(e).__name__, e)})

    def do_POST(self):
        u = urlparse(self.path)
        try:
            j = self.corps_json()
            if u.path == "/api/globals":
                ecrire_globals(j)
                return self.envoyer(200, {"ok": True})
            if u.path == "/api/instance":
                ecrire_instance(int(j["n"]), j["valeurs"])
                return self.envoyer(200, {"ok": True, "fichier": _fichier_instance(int(j["n"]))})
            if u.path == "/api/appliquer":
                return self.envoyer(200, appliquer(j["hote"], int(j["port"]), j["chemin"], j["lignes"]))
            if u.path == "/api/console":
                return self.envoyer(200, {"sortie": console(j["hote"], int(j["port"]), j["lignes"])})
            if u.path == "/api/fichier":
                if not fichier_autorise(j["p"]):
                    return self.envoyer(403, {"err": "fichier hors liste"})
                sauvegarde(j["p"])
                with open(j["p"], "w", encoding="utf-8") as f:
                    f.write(j["texte"])
                # un fichier de /etc/osmocom retouche ici n'est plus ecrase par la regeneration
                protege = os.path.dirname(os.path.realpath(j["p"])) == "/etc/osmocom"
                if protege:
                    proteger(os.path.basename(j["p"]), True)
                return self.envoyer(200, {"ok": True, "protege": protege})
            if u.path == "/api/protege":
                proteger(os.path.basename(j["nom"]), bool(j["on"]))
                return self.envoyer(200, {"ok": True})
            self.envoyer(404, {"err": "?"})
        except Exception as e:
            self.envoyer(500, {"err": "%s: %s" % (type(e).__name__, e)})


if __name__ == "__main__":
    a = argparse.ArgumentParser(description="Page web de reglage Osmocom")
    a.add_argument("--port", type=int, default=8090)
    a.add_argument("--bind", default="127.0.0.1")
    o = a.parse_args()
    print("Reglages Osmocom : http://%s:%d/" % (o.bind, o.port))
    ThreadingHTTPServer((o.bind, o.port), H).serve_forever()
```

3.431 /opt/GSM/osmo-operator/tools/osmo-ril-atmodem.py

19801 octets, 490 lignes → 490 lignes

```
#!/usr/bin/env python3
# osmo-ril-atmodem.py - LE MODEM AT VIRTUEL QUI BRANCHE oFono SUR LE RIL.
#
# LA CHAINE, ET LE TROU QU ON BOUCHE ICI.
#
#   Android → rild → libreference-ril.so → port serie AT → CE PROGRAMME → oFono → banc
#
# UN RIL Android (libreference-ril.so) n ouvre qu un port serie et y envoie du
# 3GPP 27.007 (AT+CFUN, AT+CREG?, ATD..., AT+CMGS...). oFono, lui, n est pas un
# modem : c est un consommateur de modem, avec une API D-Bus. Les deux ne se
# rencontrent donc jamais - sauf si quelqu un TIENT LE ROLE DU MODEM. C est ce
# que fait ce programme : il ouvre un pseudo-terminal, se comporte comme un
# modem 27.007 vu du port, et execute chaque commande sur oFono.
#
#   AT+CREG?        <- l etat d enregistrement lu sur org.ofono.NetworkRegistration
#   ATD100102;      <- org.ofono.VoiceCallManager.Dial
#   ATA / ATH       <- Answer / HangupAll
#   AT+CLCC         <- la liste des appels en cours (GetCalls)
#   AT+CMGS         <- org.ofono.MessageManager.SendMessage
#   +CMTI / RING    <- pousses vers le port quand oFono signale un SMS ou un appel
#
# A QUOI IL SERT AUJOURD HUI. [2026-09-07] Il a ete ecrit pour donner un modem
# au RIL d un Android en conteneur (Waydroid) ; Android est abandonne, et le
# telephone du banc - la VM postmarketOS - se branche, lui, DIRECTEMENT sur le
# modem du banc (tools/osmo-phonesim-banc.py --connect). Ce programme-ci reste
# le modem AT adosse a oFono : c est par lui qu on verifie a la main, avec
# tools/at-cmd.py sur /run/osmo-ril/at-pty, ce qu oFono repond vraiment - et
# c est le port qu on donnerait a n importe quel logiciel attendant un modem.
#
# Reglages :
#   OSMO_RIL_DIR      ou publier le lien vers le pty (defaut /run/osmo-ril)
#   OSMO_OFONO_MODEM  le modem oFono a piloter (defaut : le premier trouve)
#   OSMO_AT_ECHO      1 pour repondre en echo des le depart (ATE1)
import os
import pty
import re
import sys
import termios
import time
import tty

from gi.repository import Gio, GLib

RUN = os.environ.get("OSMO_RIL_DIR", "/run/osmo-ril")
LINK = os.path.join(RUN, "at-pty")
MODEM = os.environ.get("OSMO_OFONO_MODEM", "")
# [2026-09-06] LE JOURNAL DU PORT AT. Tout ce qui passe entre le RIL d Android
# et ce modem, dans les deux sens, horodate a la milliseconde. C est l outil
# pour voir OU une sequence s arrete - une commande a laquelle on repond mal
# bloque tout le reste en silence. « tail -f /var/log/osmo-at.log » pendant qu on
# manipule Android.
AT_LOG = os.environ.get("OSMO_AT_LOG", "/var/log/osmo-at.log")

OFONO = "org.ofono"
IF_MANAGER = "org.ofono.Manager"
IF_MODEM = "org.ofono.Modem"
IF_SMS = "org.ofono.MessageManager"
IF_VOICE = "org.ofono.VoiceCallManager"
IF_CALL = "org.ofono.VoiceCall"
IF_NETREG = "org.ofono.NetworkRegistration"
IF_SIM = "org.ofono.SimManager"
```

```
def log(*a):
    print("[at]", *a, flush=True)


_atlog = None


def atlog(sens, texte):
    """sens : '<' ce que le RIL envoie, '>' ce que le modem repond."""
    global _atlog
    if _atlog is None:
        try:
            _atlog = open(AT_LOG, "a", buffering=1)   # ligne par ligne : direct
            os.chmod(AT_LOG, 0o644)                   # lisible sans etre root
        except OSError as e:
            print("[at] journal %s impossible : %s" % (AT_LOG, e), file=sys.stderr)
            _atlog = False
    if not _atlog:
        return
    stamp = time.strftime("%H:%M:%S") + ".%03d" % (int(time.time() * 1000) % 1000)
    for ligne in texte.replace("\r", "\n").split("\n"):
        if ligne.strip():
            try:
                _atlog.write("%s %s %s\n" % (stamp, sens, ligne.strip()))
            except OSError:
                return


# ── oFONO ──────────────────────────────────────────────────────────
class Ofono:
    """Le strict necessaire d oFono, en Gio (pas de dbus-python sur l image)."""

    def __init__(self):
        self.c = Gio.bus_get_sync(Gio.BusType.SYSTEM, None)
        self.modem = MODEM or self._first_modem()

    def _first_modem(self):
        try:
            modems = self.call("/", IF_MANAGER, "GetModems").unpack()[0]
        except GLib.Error:
            return ""
        return modems[0][0] if modems else ""

    def call(self, path, iface, method, params=None, timeout=20000):
        return self.c.call_sync(OFONO, path, iface, method, params, None,
                                Gio.DBusCallFlags.NONE, timeout, None)

    def props(self, iface, path=None):
        try:
            return self.call(path or self.modem, iface, "GetProperties").unpack()[0]
        except GLib.Error:
            return {}

    def calls(self):
        try:
            return self.call(self.modem, IF_VOICE, "GetCalls").unpack()[0]
        except GLib.Error:
            return []

    def subscribe(self, iface, signal, cb, path=None):
        return self.c.signal_subscribe(OFONO, iface, signal, path, None,
                                       Gio.DBusSignalFlags.NONE, cb)


# ── LE MODEM VU DU PORT SERIE ──────────────────────────────────────
```

```
# Les etats 27.007 attendus par le RIL. On traduit ce que dit oFono.
_CREG_STATE = {"unregistered": 0, "registered": 1, "searching": 2,
               "denied": 3, "unknown": 4, "roaming": 5}


class AtModem:
    def __init__(self, ofono, fd):
        self.o = ofono
        self.fd = fd
        self.buf = b""
        self.echo = os.environ.get("OSMO_AT_ECHO") == "1"
        self.cmee = 0
        self.creg_mode = 0          # 0 = pas d URC, 1/2 = URC d enregistrement
        self.cmgf = 0               # 0 = PDU (ce que veut le RIL), 1 = texte
        self.sms_target = None      # numero en attente entre AT+CMGS et le ^Z

    # -- ecriture vers le RIL
    def send(self, text):
        atlog(">", text)
        try:
            os.write(self.fd, ("\r\n%s\r\n" % text).encode())
        except OSError as e:
            log("ecriture impossible :", e)

    def ok(self):
        self.send("OK")

    def err(self, code=100):
        self.send("+CME ERROR: %d" % code if self.cmee else "ERROR")

    # -- lecture depuis le RIL
    def feed(self, data):
        if self.echo:
            try:
                os.write(self.fd, data)
            except OSError:
                pass
        self.buf += data
        while True:
            m = re.search(rb"[\r\n\x1a]", self.buf)
            if not m:
                return
            line, self.buf = self.buf[:m.start()], self.buf[m.end():]
            end = self.buf[m.start():m.end()] if False else m.group()
            text = line.decode(errors="replace").strip()
            if text:
                atlog("<", text)
            if end == b"\x1a" or self.sms_target is not None:
                self.on_sms_body(text, submitted=(end == b"\x1a"))
            elif text:
                self.on_line(text)

    # -- le corps d un SMS, entre AT+CMGS=... et Ctrl-Z
    def on_sms_body(self, text, submitted):
        if not submitted:
            return                      # on attend encore le Ctrl-Z
        num, self.sms_target = self.sms_target, None
        if not num:
            self.err()
            return
        try:
            self.o.call(self.o.modem, IF_SMS, "SendMessage",
                        GLib.Variant("(ss)", (num, text)))
            self.send("+CMGS: 1")
            self.ok()
            log("SMS envoye a %s" % num)
```

```
    except GLib.Error as e:
        log("envoi refuse :", e.message.split(":")[-1].strip())
        self.err()

# -- une commande AT
def on_line(self, line):
    cmd = line.upper()
    if not cmd.startswith("AT"):
        return
    body = line[2:]
    ub = cmd[2:]

    # les basiques que le RIL envoie au demarrage
    if ub in ("", "&F", "Z", "E0", "E1", "+CMEE=1", "+CMEE=2", "V1", "Q0",
              "S0=0", "+CSCS=\"UCS2\"", "+CSCS=\"IRA\"", "+CMGF=0",
              "+CNMI=1,2,0,1,0", "+CGEREP=1,0", "+CLIP=1", "+COLP=0",
              "+CSSN=0,0", "+CSNS=0", "+CMOD=0"):
        if ub == "E0":
            self.echo = False
        elif ub == "E1":
            self.echo = True
        elif ub.startswith("+CMEE"):
            self.cmee = 1
        elif ub == "+CMGF=0":
            self.cmgf = 0
        self.ok()
        return

    # identite du modem : ce que le RIL affiche dans « a propos »
    if ub in ("+CGMI", "+GMI"):
        self.send("Osmocom"); self.ok(); return
    if ub in ("+CGMM", "+GMM"):
        self.send("osmo-ril-atmodem"); self.ok(); return
    if ub in ("+CGMR", "+GMR"):
        self.send("1.0"); self.ok(); return
    if ub in ("+CGSN", "+GSN"):
        self.send(self.o.props(IF_MODEM).get("Serial", "000000000000000"))
        self.ok(); return

    # la SIM
    if ub == "+CPIN?":
        sim = self.o.props(IF_SIM)
        self.send("+CPIN: READY" if sim.get("Present", True) else "+CME ERROR: 10")
        self.ok(); return
    if ub == "+CIMI":
        self.send(self.o.props(IF_SIM).get("SubscriberIdentity", "001010000000000"))
        self.ok(); return
    if ub.startswith("+CRSM") or ub.startswith("+CCID") or ub.startswith("+CSIM"):
        self.err(); return       # le RIL sait s en passer

    # la radio
    if ub.startswith("+CFUN"):
        if ub.endswith("?"):
            on = self.o.props(IF_MODEM).get("Online", False)
            self.send("+CFUN: %d" % (1 if on else 4))
            self.ok(); return
        want = ub.rstrip().endswith("1")
        try:
            self.o.call(self.o.modem, IF_MODEM, "SetProperty",
                        GLib.Variant("(sv)", ("Online", GLib.Variant("b", want))))
        except GLib.Error as e:
            log("CFUN refuse :", e.message.split(":")[-1].strip())
        self.ok(); return

    # l enregistrement reseau
    if ub.startswith("+CREG") or ub.startswith("+CGREG") or ub.startswith("+CEREG"):
```

```
    tag = "+CREG" if ub.startswith("+CREG") else ("+CGREG" if ub.startswith("+CGREG") else "+CEREG")
    if "=" in ub:
        try:
            self.creg_mode = int(ub.split("=")[1][0])
        except (ValueError, IndexError):
            self.creg_mode = 0
        self.ok(); return
    net = self.o.props(IF_NETREG)
    st = _CREG_STATE.get(str(net.get("Status", "unknown")), 4)
    lac = int(net.get("LocationAreaCode", 0) or 0)
    cid = int(net.get("CellId", 0) or 0)
    if self.creg_mode >= 2:
        self.send("%s: %d,%d,\"%04X\",\"%08X\"" % (tag, self.creg_mode, st, lac, cid))
    else:
        self.send("%s: %d,%d" % (tag, self.creg_mode, st))
    self.ok(); return

if ub.startswith("+COPS"):
    net = self.o.props(IF_NETREG)
    if ub.endswith("?"):
        self.send("+COPS: 0,0,\"%s\",0" % net.get("Name", "Osmocom"))
    self.ok(); return

if ub.startswith("+CSQ"):
    # oFono donne une force en %, le 27.007 attend 0..31
    pct = int(net_strength(self.o))
    self.send("+CSQ: %d,99" % min(31, round(pct * 31 / 100)))
    self.ok(); return

# les appels
if cmd.startswith("ATD") and (cmd.endswith(";") or ";" in cmd):
    num = re.sub(r"[^0-9+*#]", "", body.split(";")[0][1:])
    try:
        self.o.call(self.o.modem, IF_VOICE, "Dial",
                    GLib.Variant("(ss)", (num, "default")))
        self.ok()
        log("appel vers %s" % num)
    except GLib.Error as e:
        log("appel refuse :", e.message.split(":")[-1].strip())
        self.err()
    return
if ub == "A":
    for path, props in self.o.calls():
        if props.get("State") == "incoming":
            try:
                self.o.call(path, IF_CALL, "Answer")
            except GLib.Error:
                pass
    self.ok(); return
if ub in ("H", "+CHUP") or ub.startswith("+CHLD"):
    try:
        self.o.call(self.o.modem, IF_VOICE, "HangupAll")
    except GLib.Error:
        pass
    self.ok(); return
if ub == "+CLCC":
    for i, (path, props) in enumerate(self.o.calls(), start=1):
        st = {"active": 0, "held": 1, "dialing": 2, "alerting": 3,
              "incoming": 4, "waiting": 5}.get(props.get("State", ""), 6)
        direction = 1 if props.get("State") in ("incoming", "waiting") else 0
        num = props.get("LineIdentification", "")
        self.send("+CLCC: %d,%d,%d,0,0,\"%s\",129" % (i, direction, st, num))
    self.ok(); return

# les SMS
if ub.startswith("+CMGS="):
```

```
            # En mode texte le numero est entre guillemets ; en PDU le RIL
            # envoie une longueur, et le numero est dans le PDU - non gere ici
            # (le pont oFono s en charge tant qu on n a pas de rild).
            m = re.search(r'"([^"]+)"', body)
            if m:
                self.sms_target = m.group(1)
                try:
                    os.write(self.fd, b"\r\n> ")
                except OSError:
                    pass
                return
            self.err(); return
        if ub.startswith("+CMGF"):
            self.cmgf = 1 if ub.rstrip().endswith("1") else 0
            self.ok(); return
        if ub.startswith("+CSCA"):
            self.send("+CSCA: \"\",129"); self.ok(); return

        # —— CE QUE LE RIL D ANDROID DEMANDE A L INITIALISATION ——————
        # [2026-09-06] Releve sur le banc, la premiere fois que osmo-rild a
        # ouvert ce port : ATE0Q0V1, AT+CTEC?, AT+WNAM, AT+CCWA=1, AT+CMUT=0,
        # AT+CSSN=0,1, AT+CSCS="HEX", AT+CUSD=1... Tant qu on refusait tout ca,
        # libreference-ril restait bloquee au tout debut de sa sequence et ne
        # publiait jamais son interface - le framework ne voyait donc aucun
        # service radio.
        if ub in ("E0Q0V1", "E1Q0V1", "Q0V1", "E0V1"):
            self.echo = ub.startswith("E1")
            self.ok(); return
        if ub.startswith("+CTEC"):
            # « current technology » du reference-ril : 0 = GSM, ce que le banc
            # fait effectivement.
            self.send("+CTEC: 0,1" if ub.endswith("?") else "+CTEC: 0")
            self.ok(); return
        if ub.startswith("+WNAM"):
            self.send('+WNAM: "%s"' % (self.o.props(IF_NETREG).get("Name") or "Osmocom"))
            self.ok(); return
        if ub.startswith("+CSCS"):
            if ub.endswith("=?"):
                self.send('+CSCS: ("IRA","GSM","UCS2","HEX")')
            elif ub.endswith("?"):
                self.send('+CSCS: "HEX"')
            self.ok(); return

        # Meme regle que le modem cote oFono (tools/osmo-phonesim-banc.py) :
        # une AFFECTATION est acceptee - un modem qui accepte un reglage sans
        # l appliquer ne raconte rien de faux ; une LECTURE inconnue est
        # refusee, parce qu une reponse inventee ferait croire a une capacite
        # qui n existe pas.
        if "=" in ub and not ub.endswith("=?"):
            log("reglage accepte sans effet : %s" % line)
            self.ok(); return

        log("lecture non geree (refusee) : %s" % line)
        self.err()

# —— les evenements d oFono pousses vers le RIL ——————————————————
def on_incoming_sms(self, _c, _s, _p, _i, _sig, params):
    info = params[1]
    log("SMS entrant de %s" % info.get("Sender", "?"))
    self.send("+CMTI: \"SM\",1")

def on_call_added(self, _c, _s, _p, _i, _sig, params):
    props = params[1]
    if props.get("State") == "incoming":
        num = props.get("LineIdentification", "")
        self.send("+CLIP: \"%s\",129" % num)
```

```
            self.send("RING")

    def on_netreg_changed(self, _c, _s, _p, _i, _sig, params):
        if self.creg_mode and params[0] in ("Status", "CellId", "LocationAreaCode"):
            net = self.o.props(IF_NETREG)
            st = _CREG_STATE.get(str(net.get("Status", "unknown")), 4)
            self.send("+CREG: %d" % st)


# -- LE MODEM D ESSAI ----------------------------------------------------
# Sans modem oFono (banc arrete, EC25 debranche), le programme n aurait rien a
# piloter et sortirait aussitot - impossible de mettre au point le dialogue AT
# ni de verifier ce que le RIL recevra. --essai remplace donc oFono par des
# reponses par defaut : le modem repond comme un modem eteint mais poli
# (CFUN 0, CREG 0, aucun appel), et TOUTE commande d action est journalisee au
# lieu d etre executee. C est un banc d essai pour la mecanique, jamais un
# substitut au vrai chemin.
class OfonoEssai:
    modem = "/essai/modem0"

    def props(self, iface, path=None):
        if iface == IF_SIM:
            return {"Present": True, "SubscriberIdentity": "001010000000001"}
        if iface == IF_NETREG:
            return {"Status": "unregistered", "Name": "Osmocom", "Strength": 0}
        if iface == IF_MODEM:
            return {"Online": False, "Serial": "000000000000001"}
        return {}

    def calls(self):
        return []

    def call(self, path, iface, method, params=None, timeout=20000):
        log("[essai] %s.%s ignore" % (iface.rsplit(".", 1)[-1], method))
        return GLib.Variant("()", ())

    def subscribe(self, *a, **kw):
        return 0


def net_strength(o):
    return o.props(IF_NETREG).get("Strength", 0) or 0


def main():
    essai = "--essai" in sys.argv or os.environ.get("OSMO_AT_ESSAI") == "1"
    o = None
    if not essai:
        try:
            o = Ofono()
        except GLib.Error as e:
            print("[at] bus systeme injoignable : %s" % e, file=sys.stderr)
            return 1
        if not o.modem:
            print("[at] aucun modem oFono (ofonod demarre ? modem branche ?)\n"
                  "[at] --essai pour ouvrir quand meme un modem de mise au point",
                  file=sys.stderr)
            return 1
        log("modem oFono", o.modem)
    else:
        o = OfonoEssai()
        log("MODE ESSAI : aucun oFono derriere, reponses par defaut")

    master, slave = pty.openpty()
    name = os.ttyname(slave)
    tty.setraw(master)
```

```
    # Le RIL ouvrira le lien stable, pas le /dev/pts/N du jour.
    os.makedirs(RUN, exist_ok=True)
    try:
        if os.path.islink(LINK) or os.path.exists(LINK):
            os.remove(LINK)
        os.symlink(name, LINK)
    except OSError as e:
        log("lien %s non pose : %s" % (LINK, e))
    log("modem AT virtuel sur %s (lien : %s)" % (name, LINK))
    log("a donner a rild :  rild -l /vendor/lib64/libreference-ril.so -d %s" % LINK)

    m = AtModem(o, master)
    o.subscribe(IF_SMS, "IncomingMessage", m.on_incoming_sms, path=o.modem)
    o.subscribe(IF_VOICE, "CallAdded", m.on_call_added, path=o.modem)
    o.subscribe(IF_NETREG, "PropertyChanged", m.on_netreg_changed, path=o.modem)

    def readable(_src, _cond):
        try:
            data = os.read(master, 4096)
        except OSError:
            return True
        if data:
            m.feed(data)
        return True

    GLib.io_add_watch(master, GLib.IO_IN, readable)
    GLib.MainLoop().run()
    return 0


if __name__ == "__main__":
    sys.exit(main())
```

3.432 /opt/GSM/osmo-operator/tools/osmo-sim-install.sh

8179 octets, 163 lignes → 163 lignes

```
#!/bin/bash
# ============================================================================
# osmo-sim-install.sh - LA SIM DU BANC, posee une seule fois pour DEUX chemins.
#
# Meme logique que tools/osmo-extras-install.sh : ce qui doit exister a la fois
# dans le SUPPLEMENT (addition.sh --sim, sur une machine deja installee) et dans
# le NATIF (l ISO, iso_modules/80-chroot.sh, dans le chroot) est ecrit UNE fois
# ici ; les deux chemins sourcent ce script et appellent osmo_sim_install.
# Rien n est fatal : un paquet absent ne doit pas arreter le reste.
#
# Ce qu il pose :
#   - pcscd + libccid : le demon PC/SC et le pilote CCID du lecteur HID Omnikey
#     (6121 verifie le 2026-10-14, ATR de la sysmoISIM-SJA5 lu) ; pcsc-tools
#     (pcsc_scan) et opensc (opensc-tool -l/-a) pour voir le lecteur et l ATR ;
#   - Ruby + ruby-libxml + le gem smartcard : softSIM (/opt/GSM/softsim), le
#     serveur SAP qui sert soit une SIM logicielle (sim.xml, type sim), soit la
#     carte du lecteur (type pcsc) au mobile osmocom-bb (« sim sap ») ;
#   - pySim (Osmocom, depuis son git : le « pySim » de PyPI est un autre projet)
#     dans son venv /opt/GSM/pysim-venv : pySim-read, pySim-prog, pySim-shell,
#     pour lire et programmer la sysmoISIM-SJA5 (clé ADM1 requise pour ecrire) ;
#   - sysmo-usim-tool (sysmocom) dans /opt/GSM/sysmo-usim-tool : l outil qui ecrit
#     IMSI, Ki, algorithme (COMP128v1) et OPc sur la sysmoISIM-SJA5 ;
#   - softSIM lui-meme si absent (clone + patches tools/softsim/*.patch + shim
#     Ruby 3.2), comme iso_modules/51-softsim.sh le fait pour l ISO ;
#   - pcscd.socket active (sauf en chroot : systemd absent, l unite s active au
#     premier boot par son preset).
#
```

```
# Idempotent : relancable sans degat. Variables :
#   OSMO_SIM_PYSIM=0      ne pas poser pySim
#   OSMO_SIM_SOFTSIM=0    ne pas poser softSIM
#   PYSIM_VENV            le venv de pySim (defaut /opt/GSM/pysim-venv)
#   SOFTSIM_DIR           l arbre softSIM (defaut /opt/GSM/softsim)
#   REPO / DIR            le depot osmo-operator (patches softSIM, shim Ruby)
# ============================================================================

: "${GREEN:=}"; : "${YELLOW:=}"; : "${CYAN:=}"; : "${BOLD:=}"; : "${NC:=}"

_sim_repo() {
    local d
    for d in "${REPO:-}" "${DIR:-}" /opt/GSM/osmo-operator "$(cd "$(dirname "${BASH_SOURCE[0]}")/.." 2>/dev/null
     ↪  && pwd)"; do
        [ -n "$d" ] && [ -f "$d/tools/softsim/ruby_compat.rb" ] && { printf '%s' "$d"; return 0; }
    done
    return 1
}

_sim_say()  { echo -e "  ${GREEN}✓${NC} $*"; }
_sim_warn() { echo -e "  ${YELLOW}!${NC} $*" >&2; }

# En chroot (ISO) : pas de systemd qui tourne, pas de lecteur a tester.
_sim_in_chroot() { [ "$(stat -c %d:%i / 2>/dev/null)" != "$(stat -c %d:%i /proc/1/root/. 2>/dev/null)" ]; }

osmo_sim_install_apt() {
    local pkgs="pcscd libccid pcsc-tools opensc libpcsclite1 libpcsclite-dev
                python3 python3-venv python3-pip python3-pyscard swig
                ruby ruby-libxml ruby-dev build-essential git"
    export DEBIAN_FRONTEND=noninteractive
    local apt=apt-get; command -v apt-fast >/dev/null 2>&1 && apt=apt-fast
    # shellcheck disable=SC2086
    if $apt install -y --no-install-recommends $pkgs >/dev/null 2>&1; then
        _sim_say "paquets SIM : pcscd, libccid, pcsc-tools, opensc, pyscard, ruby"
    else
        # un par un : un paquet absent du miroir ne doit pas bloquer les autres
        local p; for p in $pkgs; do
            $apt install -y --no-install-recommends "$p" >/dev/null 2>&1 || _sim_warn "paquet absent : $p"
        done
    fi
    if ! _sim_in_chroot && command -v systemctl >/dev/null 2>&1; then
        systemctl enable --now pcscd.socket >/dev/null 2>&1 && _sim_say "pcscd.socket actif" \
            || _sim_warn "pcscd.socket non active"
    fi
}

osmo_sim_install_gem() {
    command -v gem >/dev/null 2>&1 || { _sim_warn "gem absent : pas de softSIM pcsc"; return 0; }
    if gem list -i smartcard >/dev/null 2>&1; then
        _sim_say "gem smartcard deja present"
    elif gem install smartcard --no-document >/dev/null 2>&1; then
        _sim_say "gem smartcard installe (softSIM en lecteur PC/SC)"
    else
        _sim_warn "gem smartcard non installe (softSIM pcsc indisponible)"
    fi
}

osmo_sim_install_pysim() {
    [ "${OSMO_SIM_PYSIM:-1}" = 1 ] || return 0
    local venv="${PYSIM_VENV:-/opt/GSM/pysim-venv}"
    local repo="${OSMO_PYSIM_REPO:-https://gitea.osmocom.org/sim-card/pysim.git}"
    if [ -x "$venv/bin/pySim-shell.py" ] && [ "${OSMO_SIM_PYSIM_UPDATE:-0}" != 1 ]; then
        _sim_say "pySim deja present : $venv"
        return 0
    fi
    python3 -m venv "$venv" >/dev/null 2>&1 || { _sim_warn "venv pySim impossible ($venv)"; return 0; }
```

```
    # pyscard a besoin de swig + libpcsclite-dev pour se construire (PKGS ci-dessus).
    if "$venv/bin/pip" install -q --disable-pip-version-check --no-cache-dir "git+$repo" >/dev/null 2>&1; then
        _sim_say "pySim (Osmocom) : $venv/bin/pySim-{read,prog,shell}.py"
    else
        _sim_warn "pySim non installe (reseau ? swig/libpcsclite-dev ?)"
    fi
}

osmo_sim_install_sysmo_tool() {
    local d="${SYSMO_TOOL:-/opt/GSM/sysmo-usim-tool}"
    if [ -f "$d/sysmo-isim-tool.sja5.py" ]; then _sim_say "sysmo-usim-tool deja present : $d"; return 0; fi
    if GIT_TERMINAL_PROMPT=0 git clone -q --depth 1
     ↪  "${OSMO_SYSMO_TOOL_REPO:-https://gitea.sysmocom.de/sysmocom/sysmo-usim-tool.git}" "$d" 2>/dev/null; then
        _sim_say "sysmo-usim-tool (sysmo-isim-tool.sja5.py) : $d"
    else
        rm -rf "$d"; _sim_warn "sysmo-usim-tool : clone impossible (reseau ?) - pySim-prog servira"
    fi
}

osmo_sim_install_softsim() {
    [ "${OSMO_SIM_SOFTSIM:-1}" = 1 ] || return 0
    local sdir="${SOFTSIM_DIR:-/opt/GSM/softsim}" repo p
    repo="$(_sim_repo)" || { _sim_warn "depot osmo-operator introuvable : softSIM non pose"; return 0; }
    if [ ! -f "$sdir/src/demo_server.rb" ]; then
        rm -rf "$sdir"
        if GIT_TERMINAL_PROMPT=0 git clone -q --depth 1
         ↪  "${OSMO_SOFTSIM_REPO:-https://gitea.osmocom.org/sim-card/softsim.git}" "$sdir" 2>/dev/null; then
            _sim_say "softSIM clone : $sdir"
        else
            rm -rf "$sdir"; _sim_warn "softSIM : clone impossible (reseau ?)"; return 0
        fi
    fi
    install -Dm644 "$repo/tools/softsim/comp128.rb" "$sdir/src/lib/comp128.rb"
    for p in "$repo"/tools/softsim/*.patch; do
        [ -f "$p" ] || continue
        if git -C "$sdir" apply --check -R "$p" >/dev/null 2>&1; then
            :  # deja applique
        elif git -C "$sdir" apply "$p" >/dev/null 2>&1; then
            _sim_say "softSIM : patch $(basename "$p") applique"
        else
            _sim_warn "softSIM : patch $(basename "$p") ne s applique pas"
        fi
    done
    install -Dm644 "$repo/tools/softsim/ruby_compat.rb" /usr/local/share/softsim/ruby_compat.rb
    if [ -f "$sdir/src/sim.xml" ]; then
        python3 "$repo/tools/softsim/make_sim_op1.py" "$sdir/src/sim.xml" "$sdir/src/sim-op1.xml" \
            --imsi "${OSMO_SIM_IMSI:-001010001000001}" >/dev/null 2>&1 \
            && _sim_say "softSIM : sim-op1.xml (IMSI ${OSMO_SIM_IMSI:-001010001000001})"
    fi
    _sim_say "softSIM pret : $sdir (shim /usr/local/share/softsim/ruby_compat.rb)"
}

osmo_sim_install() {
    echo -e "${CYAN}SIM du banc : lecteur PC/SC (Omnikey), softSIM (SAP), pySim${NC}"
    osmo_sim_install_apt
    osmo_sim_install_gem
    osmo_sim_install_pysim
    osmo_sim_install_sysmo_tool
    osmo_sim_install_softsim
    if ! _sim_in_chroot && command -v opensc-tool >/dev/null 2>&1; then
        local r; r="$(timeout 10 opensc-tool -l 2>/dev/null | sed -n '3,$p')"
        [ -n "$r" ] && echo -e "  ${BOLD}lecteurs :${NC}\n$(printf '%s\n' "$r" | sed 's/^/    /')" \
                    || echo -e "  ${YELLOW}aucun lecteur PC/SC branche${NC}"
    fi
    echo -e "  ${BOLD}suite :${NC} tools/osmo-sim.sh lecteur | lire | programmer | sap  (voir wiki/SIM.md)"
    return 0
```

```
}

# Lance directement : fait le travail (sinon, source + osmo_sim_install).
if [ "${BASH_SOURCE[0]}" = "$0" ]; then
    [ "$(id -u)" -eq 0 ] || { echo "osmo-sim-install.sh : a lancer en root (sudo)" >&2; exit 1; }
    osmo_sim_install
fi
```

3.433 /opt/GSM/osmo-operator/tools/osmo-sim.sh

20356 octets, 371 lignes → 371 lignes

```
#!/bin/bash
# ============================================================================
# osmo-sim.sh - LA SIM DU BANC DANS UN LECTEUR PC/SC (HID Omnikey).
#
# La carte (sysmoISIM-SJA5 de sysmocom, dans un OMNIKEY 6121) est lue et
# programmee avec pySim (Osmocom, venv /opt/GSM/pysim-venv), puis SERVIE au
# mobile osmocom-bb par le serveur SAP de softSIM en mode pcsc : c est ce que
# fait start-native.sh --sap quand une carte est dans le lecteur.
#
#   tools/osmo-sim.sh lecteur                les lecteurs PC/SC, l ATR, le type de carte
#   tools/osmo-sim.sh lire                   tout ce que la carte dit (pySim-read)
#   tools/osmo-sim.sh programmer [options]   ecrit l identite du banc sur la carte
#   tools/osmo-sim.sh verifier  [options]    la carte porte-t-elle l identite du banc,
#                                            le HLR la connait-il : prete pour --sap ?
#   tools/osmo-sim.sh hlr [--creer] [options] l abonne de la carte dans le HLR
#   tools/osmo-sim.sh sap [--socket S]       sert la carte par SAP (premier plan),
#                                            ce que start-native.sh --sap fait tout seul
#   tools/osmo-sim.sh shell                  pySim-shell sur la carte (ADM1 si connue)
#
# L IDENTITE DU BANC (meme plan que qosmo/run_modules/21-abonnes-hlr.sh et
# start-native.sh, section "ms_imsi / ms_ki") :
#   IMSI   = MCC MNC <operateur sur 4> <MS sur 6>        001 01 0001 000001
#   Ki     = 00112233445566778899aabbccdd <MS> <operateur>   ...dd 01 01
#   MSISDN = noeud*100000 + operateur*100 + MS                 100101
#   algo   = COMP128v1 (osmo-hlr : aud2g comp128v1)
# Options : --ms N  --op N  --mcc 001  --mnc 01  --imsi I  --ki K  --msisdn M
#           --lecteur N (index PC/SC, defaut 0)  --dry-run  --oui (pas de question)
#
# L ECRITURE passe par l outil de sysmocom, sysmo-usim-tool (sysmo-isim-tool.sja5.py,
# SYSMO_TOOL=/opt/GSM/sysmo-usim-tool) : IMSI, Ki, algorithme COMP128v1, OPc. Sans
# lui (ou OSMO_SIM_PYSIM_PROG=1) : pySim-prog puis pySim-shell pour l algorithme.
#
# LA CLE ADM1 : ecrire sur une sysmoISIM demande la cle ADM1 de LA carte (8
# chiffres, fournie par sysmocom avec la carte : fichier CSV / courriel de la
# commande). Trois essais faux BLOQUENT la carte pour de bon : le script ne
# devine jamais. Sources, dans l ordre : --adm1 K, OSMO_SIM_ADM1, le fichier
# /etc/osmocom/sim-adm1 (une ligne, ou "ICCID clé" par ligne pour plusieurs
# cartes), ~/.config/osmo/sim-adm1.
# ============================================================================
set -u

HERE="$(cd "$(dirname "${BASH_SOURCE[0]}")/.." && pwd)"
PYSIM_VENV="${PYSIM_VENV:-/opt/GSM/pysim-venv}"
SOFTSIM_DIR="${SOFTSIM_DIR:-/opt/GSM/softsim}"
SAP_SOCK="${SAP_SOCK:-/tmp/osmocom_sap}"
HLR_VTY_PORT="${HLR_VTY_PORT:-4258}"
HLR_DB="${HLR_DB:-/var/lib/osmocom/hlr.db}"
OSMOCOM_CFG="${OSMOCOM_CFG:-/etc/osmocom}"
# Le python qui a pyscard : celui du venv pySim (le profil de root met /root/.env,
# le venv du pont, en tete du PATH, et il n a pas pyscard), sinon celui du systeme.
PY="$PYSIM_VENV/bin/python3"; [ -x "$PY" ] || PY=/usr/bin/python3
```

```
if [ -t 1 ]; then
    C_OK=$'\033[32m'; C_KO=$'\033[31m'; C_W=$'\033[33m'; C_B=$'\033[1m'; C_D=$'\033[2m'; C_Z=$'\033[0m'
else
    C_OK=; C_KO=; C_W=; C_B=; C_D=; C_Z=
fi
ok()   { printf '  %s✓%s %s\n' "$C_OK" "$C_Z" "$*"; }
ko()   { printf '  %s✗%s %s\n' "$C_KO" "$C_Z" "$*" >&2; }
warn() { printf '  %s!%s %s\n' "$C_W" "$C_Z" "$*" >&2; }
die()  { ko "$*"; exit 1; }

usage() { sed -n '2,/^# ====*$/p' "$0" | sed -n '2,$p' | sed 's/^# \{0,1\}//' | sed '$d'; }

# ── le plan d identite ──────────────────────────────────────────────
MS=1 OP=1 MCC="" MNC="" IMSI="" KI="" MSISDN="" READER="${OSMO_SIM_READER:-0}"
DRY=0 OUI=0 ADM1="${OSMO_SIM_ADM1:-}" CREER=0 SOCK="$SAP_SOCK" VERB="${SAP_VERBOSITY:-3}"

plan_mcc() {
    [ -n "$MCC" ] && { printf '%s' "$MCC"; return; }
    sed -n 's/^[[:space:]]*network country code[[:space:]]\{1,\}\([0-9]*\).*/\1/p' \
        "$OSMOCOM_CFG/osmo-msc.cfg" 2>/dev/null | head -1 | grep . || printf '001'
}
plan_mnc() {
    [ -n "$MNC" ] && { printf '%s' "$MNC"; return; }
    sed -n 's/^[[:space:]]*mobile network code[[:space:]]\{1,\}\([0-9]*\).*/\1/p' \
        "$OSMOCOM_CFG/osmo-msc.cfg" 2>/dev/null | head -1 | grep . || printf '01'
}
plan_node() {
    local n="${OSMO_WAN_NODE:-${WAN_NODE_ID:-}}"
    [ -n "$n" ] || n="$(sed -n 's/^PLAN_NODE=//p' "$OSMOCOM_CFG/radio-plan.env" 2>/dev/null | tail -1)"
    case "$n" in [1-9]) ;; *) n=1 ;; esac
    printf '%s' "$n"
}
plan_imsi()   { [ -n "$IMSI" ] && printf '%s' "$IMSI" || printf '%s%s%04d%06d' "$(plan_mcc)" "$(plan_mnc)" "$OP"
 ↪  "$MS"; }
plan_ki()     { [ -n "$KI" ] && printf '%s' "$KI" || printf '00112233445566778899aabbccdd%02x%02x' "$MS" "$OP"; }
plan_msisdn() { [ -n "$MSISDN" ] && printf '%s' "$MSISDN" || printf '%d' $(( $(plan_node) * 100000 + OP * 100 + MS
 ↪  )); }

# ── les outils ──────────────────────────────────────────────────────
pysim() {  # pysim read|prog|shell args...
    local t="$1"; shift
    local bin="$PYSIM_VENV/bin/pySim-$t.py"
    [ -x "$bin" ] || die "pySim absent ($bin) : sudo $HERE/tools/osmo-sim-install.sh"
    "$bin" -p "$READER" "$@"
}
pcscd_pret() {
    if command -v systemctl >/dev/null 2>&1 && ! systemctl is-active --quiet pcscd pcscd.socket 2>/dev/null; then
        systemctl start pcscd.socket >/dev/null 2>&1 || true
    fi
}
# Les lecteurs, et s ils portent une carte : "N Yes|No nom"
lecteurs() {
    "$PY" - <<'PYSC' 2>/dev/null
from smartcard.System import readers
for i, r in enumerate(readers()):
    try:
        c = r.createConnection(); c.connect(); atr = bytes(c.getATR()).hex(); c.disconnect()
        print(i, "Yes", atr, str(r))
    except Exception:
        print(i, "No", "-", str(r))
PYSC
}
carte_presente() { lecteurs | awk -v r="$READER" '$1==r && $2=="Yes"{f=1} END{exit !f}'; }
exiger_carte() {
    pcscd_pret
    "$PY" -c 'import smartcard' 2>/dev/null \
```

```
        || die "pyscard absent ($PY) : sudo $HERE/tools/osmo-sim-install.sh"
    [ -n "$(lecteurs)" ] || die "aucun lecteur PC/SC (pcscd actif ? Omnikey branche ? libccid ?)"
    carte_presente || die "pas de carte dans le lecteur $READER :$(lecteurs | sed 's/^/\n    /')"
}
# L ICCID et l IMSI de la carte (pySim-read), "iccid imsi"
carte_identite() {
    local out
    out="$(timeout 90 "$PYSIM_VENV/bin/pySim-read.py" -p "$READER" 2>/dev/null)" || return 1
    printf '%s %s %s' \
        "$(printf '%s\n' "$out" | sed -n 's/^ICCID: *//p' | head -1)" \
        "$(printf '%s\n' "$out" | sed -n 's/^IMSI: *//p' | head -1)" \
        "$(printf '%s\n' "$out" | sed -n 's/^Autodetected card type: *//p' | head -1 | tr ' ' '_')"
}

# La cle ADM1 : --adm1, OSMO_SIM_ADM1, /etc/osmocom/sim-adm1, ~/.config/osmo/sim-adm1
adm1_trouver() {  # $1 = ICCID de la carte
    local f k
    [ -n "$ADM1" ] && { printf '%s' "$ADM1"; return 0; }
    for f in "$OSMOCOM_CFG/sim-adm1" "$HOME/.config/osmo/sim-adm1"; do
        [ -r "$f" ] || continue
        k="$(awk -v ic="$1" 'NF==2 && $1==ic {print $2; exit} NF==1 && $1 !~ /^#/ {one=$1} END{if(!k && one) print
        ↪  one}' "$f")"
        [ -n "$k" ] && { printf '%s' "$k"; return 0; }
    done
    return 1
}

vty() {  # vty cmd... -> sortie du VTY HLR
    local c; { echo enable; for c in "$@"; do echo "$c"; done; echo exit; } \
        | timeout 10 socat -T 5 - "TCP:127.0.0.1:$HLR_VTY_PORT,crlf" 2>/dev/null
}
hlr_connait() {  # $1 = IMSI
    if [ -r "$HLR_DB" ] && command -v sqlite3 >/dev/null 2>&1; then
        [ "$(sqlite3 -readonly "$HLR_DB" "select count(*) from subscriber where imsi='$1';" 2>/dev/null)" -ge 1 ]
        ↪  2>/dev/null && return 0
    fi
    command -v socat >/dev/null 2>&1 && vty "subscriber imsi $1 show" | grep -q "IMSI: $1"
}

# ── les commandes ──────────────────────────────────────────────────────
cmd_lecteur() {
    pcscd_pret
    local l; l="$(lecteurs)"
    [ -n "$l" ] || die "aucun lecteur PC/SC (pcscd actif ? Omnikey branche ? libccid ?)"
    printf '%slecteurs PC/SC :%s\n' "$C_B" "$C_Z"
    printf '%s\n' "$l" | while read -r i yes atr name; do
        if [ "$yes" = Yes ]; then
            printf '  %s%s%s  %s\n      carte : ATR %s\n' "$C_B" "$i" "$C_Z" "$name" "$atr"
        else
            printf '  %s%s%s  %s\n      %spas de carte%s\n' "$C_B" "$i" "$C_Z" "$name" "$C_D" "$C_Z"
        fi
    done
    if carte_presente && [ -x "$PYSIM_VENV/bin/pySim-read.py" ]; then
        read -r iccid imsi type <<<"$(carte_identite)"
       printf '  lecteur %s : %s  ICCID %s  IMSI %s\n' "$READER" "${type:-type_inconnu}" "${iccid:-?}" "${imsi:-?}"
    fi
}

cmd_lire() { exiger_carte; pysim read; }

cmd_shell() {
    exiger_carte
    local k; k="$(adm1_trouver "$(carte_identite | cut -d' ' -f1)")" || true
    if [ -n "$k" ]; then pysim shell -a "$k" "$@"; else pysim shell "$@"; fi
}
```

```
# pySim-prog (identite, PLMN, SMSP, Ki) puis pySim-shell : l algorithme 2G de la
# sysmoISIM. pySim-prog n ecrit que le Ki (octet 1..16 de EF.SIM_AUTH_KEY), pas
# l octet de configuration : la carte resterait en Milenage et le HLR (COMP128v1)
# repondrait « authentication failed ». Octet 0 : bits 7-6 libres, bit 5 SRES
# derivation 1, bit 4 OPc, bits 3-0 algorithme (1 = COMP128v1, 4 = Milenage) :
# 0x11. Ecrit dans DF.SYSTEM/EF.SIM_AUTH_KEY (6f20) et ADF.USIM/EF.USIM_AUTH_KEY_2G
# (af22), les deux fichiers 2G de la carte. OP/OPc : 16 octets a zero.
cmd_programmer() {
    exiger_carte
    local imsi ki msisdn mcc mnc iccid cur_imsi type
    imsi="$(plan_imsi)"; ki="$(plan_ki)"; msisdn="$(plan_msisdn)"; mcc="$(plan_mcc)"; mnc="$(plan_mnc)"
    case "$imsi" in [0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]) ;; *) die "IMSI
     ↪  a 15 chiffres attendu : $imsi" ;; esac
    case "$ki" in [0-9a-fA-F][0-9a-fA-F][0-9a-fA-F][0-9a-fA-F][0-9a-fA-F][0-9a-fA-F][0-9a-fA-F][0-9a-fA-F][0-9a-
     ↪  fA-F][0-9a-fA-F][0-9a-fA-F][0-9a-fA-F][0-9a-fA-F][0-9a-fA-F][0-9a-fA-F][0-9a-fA-F][0-9a-fA-F][0-9a-fA-
     ↪  F][0-9a-fA-F][0-9a-fA-F][0-9a-fA-F][0-9a-fA-F][0-9a-fA-F][0-9a-fA-F][0-9a-fA-F][0-9a-fA-F][0-9a-fA-F][0-
     ↪  9a-fA-F][0-9a-fA-F][0-9a-fA-F][0-9a-fA-F][0-9a-fA-F]) ;; *) die "Ki : 32 caracteres hexa attendus : $ki"
     ↪  ;; esac
    read -r iccid cur_imsi type <<<"$(carte_identite)"
    [ -n "$iccid" ] || die "la carte ne repond pas a pySim-read (lecteur $READER)"
    printf '%sla carte%s   : %s  ICCID %s  IMSI actuel %s\n' "$C_B" "$C_Z" "${type:-?}" "$iccid" "${cur_imsi:-?}"
    printf '%sle banc%s    : IMSI %s  PLMN %s-%s  Ki %s  MSISDN %s  COMP128v1\n' "$C_B" "$C_Z" "$imsi" "$mcc"
     ↪  "$mnc" "$ki" "$msisdn"
    local adm=""
    if [ "$DRY" = 0 ]; then
        adm="$(adm1_trouver "$iccid")" || die "cle ADM1 inconnue pour l ICCID $iccid : --adm1 K, OSMO_SIM_ADM1, ou
         ↪  $OSMOCOM_CFG/sim-adm1 (voir l en-tete). Elle vient de sysmocom avec la carte. --dry-run montre sans
         ↪  ecrire."
        case "$adm" in [0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]) ;; *) die "ADM1 : 8 chiffres attendus (une
         ↪  sysmoISIM), lu « ${#adm} caracteres »" ;; esac
        if [ "$OUI" = 0 ]; then
            printf '%sEcrire ces valeurs sur la carte %s ? Trois ADM1 faux bloquent la carte. [o/N] %s' "$C_W"
             ↪  "$iccid" "$C_Z"
            read -r rep; case "$rep" in o|O|oui|y|Y) ;; *) echo "abandon"; exit 1 ;; esac
        fi
    fi
    local sysmo="${SYSMO_TOOL:-/opt/GSM/sysmo-usim-tool}/sysmo-isim-tool.sja5.py"
    if [ "$type" = sysmoISIM-SJA5 ] && [ -f "$sysmo" ] && [ "${OSMO_SIM_PYSIM_PROG:-0}" != 1 ]; then
        # 1. L outil de sysmocom (sysmo-usim-tool, sysmo-isim-tool.sja5.py) : IMSI,
        # longueur du MNC, Ki (2G/3G/4G5G), algorithme 2G COMP128v1 (3G Milenage),
        # OPc. C est lui qui sait poser l algorithme ; pySim-prog ne le fait pas.
        # [2026-10-14] Verifie sur la carte : « 2g: 1=COMP128v1 », Ki relu.
        printf '%s1. sysmo-isim-tool.sja5 (sysmocom)%s\n' "$C_B" "$C_Z"
        if [ "$DRY" = 1 ]; then
            printf '    %s -a <ADM1> -J %s -N %s -K %s -T COMP128v1:MILENAGE -C %s\n' "$(basename "$sysmo")"
             ↪  "$imsi" "${#mnc}" "$ki" "00000000000000000000000000000000"
            ok "dry-run : rien n a ete ecrit"; return 0
        fi
        local out
        if ! out="$("$PY" "$sysmo" -a "$adm" -J "$imsi" -N "${#mnc}" -K "$ki" -T COMP128v1:MILENAGE \
                    -C 00000000000000000000000000000000 2>&1)"; then
            printf '%s\n' "$out" | tail -12 | sed 's/^/    /' >&2
            die "sysmo-isim-tool a echoue (ADM1 refusee ? carte retiree ?)"
        fi
        printf '%s\n' "$out" | grep -E 'programmed|algorithm|2g:|3g:|IMSI|Error|rror' | sed 's/^/    /'
        # 2. relire l algorithme et le Ki avec le meme outil
        printf '%s2. relecture (algorithme, Ki)%s\n' "$C_B" "$C_Z"
        out="$("$PY" "$sysmo" -a "$adm" -t -k 2>&1)"
       printf '%s\n' "$out" | grep -q '2g: 1=COMP128v1' || { printf '%s\n' "$out" | tail -8 | sed 's/^/    /' >&2;
         ↪  die "COMP128v1 non relu sur la carte"; }
        printf '%s\n' "$out" | grep -q "2g: Key: $ki" || { printf '%s\n' "$out" | tail -8 | sed 's/^/    /' >&2;
         ↪  die "Ki non relu sur la carte"; }
        ok "COMP128v1 et Ki relus sur la carte (2g)"
    else
        # Sans l outil sysmocom (ou OSMO_SIM_PYSIM_PROG=1) : pySim-prog puis pySim-shell.
        # 1. pySim-prog : ICCID conserve, IMSI, PLMN (HPLMN, PLMNsel), SPN, SMSP, MSISDN, Ki
```

```
        local args=(-t sysmoISIM-SJA5 -s "$iccid" -i "$imsi" -k "$ki" -o 00000000000000000000000000000000
                    -x "$mcc" -y "$mnc" --mnclen "${#mnc}" -n "Osmo $OP" --msisdn "$msisdn" --acc 0001)
        [ "$type" = sysmoISIM-SJA5 ] || args=("${args[@]:2}")   # autre carte : laisser pySim la reconnaitre
        [ "$DRY" = 1 ] && args+=(--dry-run)
        [ -n "$adm" ] && args+=(-a "$adm")
        printf '%s1. pySim-prog%s\n' "$C_B" "$C_Z"
        local out
        if ! out="$(pysim prog "${args[@]}" 2>&1)"; then
            printf '%s\n' "$out" | tail -15 >&2
            die "pySim-prog a echoue (ADM1 refusee ? carte retiree ?)"
        fi
        printf '%s\n' "$out" | grep -E '^( > |Programming|Dry Run|Autodetected)' | sed 's/^/    /'
        [ "$DRY" = 1 ] && { ok "dry-run : rien n a ete ecrit"; return 0; }
        # 2. pySim-shell : l algorithme 2G = COMP128v1 dans les deux fichiers d authentification.
        # verify_adm recoit la cle : sans argument il la cherche dans un CSV par ICCID
        # (« Unable to find card key data »), -a ne suffit pas.
        printf '%s2. pySim-shell : algorithme COMP128v1%s\n' "$C_B" "$C_Z"
        local script; script="$(mktemp)"
        cat >"$script" <<EOF
verify_adm $adm
select MF
select DF.SYSTEM
select EF.SIM_AUTH_KEY
update_binary 11${ki}00000000000000000000000000000000
read_binary_decoded
select MF
select ADF.USIM
select EF.USIM_AUTH_KEY_2G
update_binary 11${ki}00000000000000000000000000000000
read_binary_decoded
EOF
        out="$(pysim shell -a "$adm" --noprompt --script "$script" 2>&1)"; rm -f "$script"
        if printf '%s\n' "$out" | grep -q -i -e 'SW Match Error' -e 'Exception' -e 'EXCEPTION' -e 'Error'; then
            printf '%s\n' "$out" | grep -i -e error -e exception | head -5 | sed 's/^/    /' >&2
            die "l ecriture de l algorithme a echoue (voir ci-dessus) ; l identite (etape 1) est ecrite"
        fi
        local n; n="$(printf '%s\n' "$out" | grep -c '"algorithm": "comp128v1"')"
        [ "$n" -ge 1 ] || { printf '%s\n' "$out" | tail -8 | sed 's/^/    /' >&2; die "COMP128v1 non relu sur la
        ↪  carte"; }
        ok "COMP128v1 relu dans $n fichier(s) d authentification"
    fi
    # 3. relire
    read -r iccid cur_imsi type <<<"$(carte_identite)"
    [ "$cur_imsi" = "$imsi" ] || die "relecture : IMSI $cur_imsi au lieu de $imsi"
    ok "carte programmee : IMSI $imsi, Ki ${ki:0:8}..., COMP128v1, MSISDN $msisdn"
    cmd_hlr
}

cmd_hlr() {
    local imsi ki iccid type
    if [ -n "$IMSI" ]; then imsi="$IMSI"; else
        exiger_carte; read -r iccid imsi type <<<"$(carte_identite)"
        [ -n "$imsi" ] || die "IMSI illisible sur la carte"
    fi
    if hlr_connait "$imsi"; then
        ok "HLR : l abonne $imsi existe ($HLR_DB / VTY $HLR_VTY_PORT)"
        return 0
    fi
    if [ "$CREER" = 1 ]; then
        ki="$(plan_ki)"
        command -v socat >/dev/null 2>&1 || die "socat absent : impossible d ecrire dans le VTY HLR"
        vty "subscriber imsi $imsi create" "subscriber imsi $imsi update msisdn $(plan_msisdn)" \
            "subscriber imsi $imsi update aud2g comp128v1 ki $ki" >/dev/null
        hlr_connait "$imsi" && ok "HLR : abonne $imsi cree (COMP128v1, Ki ${ki:0:8}..., MSISDN $(plan_msisdn))" \
                            || die "HLR : creation de $imsi echouee (osmo-hlr lance ? VTY $HLR_VTY_PORT ?)"
        return 0
```

```
    fi
    if [ "$imsi" = "$(plan_imsi)" ]; then
        warn "HLR : $imsi inconnu pour l instant - c est l IMSI du MS#$MS, start-native.sh le cree au lancement
         ↪  (21-abonnes-hlr)"
    else
        warn "HLR : $imsi inconnu - tools/osmo-sim.sh hlr --creer --ki <Ki de la carte> (ou programmer la carte au
         ↪  plan du banc)"
    fi
    return 1
}

cmd_verifier() {
    exiger_carte
    local iccid imsi type rc=0
    read -r iccid imsi type <<<"$(carte_identite)"
    [ -n "$imsi" ] || die "IMSI illisible sur la carte (lecteur $READER)"
    ok "carte : ${type:-?}  ICCID $iccid  IMSI $imsi"
    if [ "$imsi" = "$(plan_imsi)" ]; then ok "IMSI = celui du MS#$MS de l operateur $OP (plan du banc)"
    else warn "IMSI $imsi ≠ plan du banc $(plan_imsi) : tools/osmo-sim.sh programmer, ou hlr --creer"; rc=1; fi
    cmd_hlr || rc=1
    if [ -f "$SOFTSIM_DIR/src/demo_server.rb" ] && ruby -e 'require "smartcard"' >/dev/null 2>&1; then
        ok "softSIM pcsc : $SOFTSIM_DIR, gem smartcard present"
    else warn "softSIM pcsc indisponible : sudo $HERE/tools/osmo-sim-install.sh"; rc=1; fi
    if [ $rc -eq 0 ]; then ok "prete : ./start-native.sh --sap (la carte est servie par SAP au mobile)"
    else warn "pas encore prete pour --sap"; fi
    return $rc
}

cmd_sap() {
    exiger_carte
    [ -f "$SOFTSIM_DIR/src/demo_server.rb" ] || die "softSIM absent ($SOFTSIM_DIR) : sudo
     ↪  $HERE/tools/osmo-sim-install.sh"
    ruby -e 'require "smartcard"' >/dev/null 2>&1 || die "gem smartcard absent : sudo
     ↪  $HERE/tools/osmo-sim-install.sh"
    local shim=/usr/local/share/softsim/ruby_compat.rb
    [ -f "$shim" ] || shim="$HERE/tools/softsim/ruby_compat.rb"
    for p in $(ps -eo pid,args | awk '/[d]emo_server\.rb/ {print $1}'); do kill "$p" 2>/dev/null; done
    rm -f "$SOCK"
    ok "serveur SAP (softSIM, type pcsc) sur $SOCK - Ctrl-C pour arreter ; le mobile : sim sap + sap-socket $SOCK"
    cd "$SOFTSIM_DIR/src" && exec ruby -r"$shim" demo_server.rb --type pcsc --socket unix --unix "$SOCK"
     ↪  --verbosity "$VERB"
}

# ── ligne de commande ──────────────────────────────────────────────────
CMD="${1:-}"; [ $# -gt 0 ] && shift
REST=()
while [ $# -gt 0 ]; do
    case "$1" in
        --ms)       MS="${2:?}"; shift ;;
        --op)       OP="${2:?}"; shift ;;
        --mcc)      MCC="${2:?}"; shift ;;
        --mnc)      MNC="${2:?}"; shift ;;
        --imsi)     IMSI="${2:?}"; shift ;;
        --ki)       KI="${2:?}"; shift ;;
        --msisdn)   MSISDN="${2:?}"; shift ;;
        --adm1)     ADM1="${2:?}"; shift ;;
        --lecteur|-p) READER="${2:?}"; shift ;;
        --socket)   SOCK="${2:?}"; shift ;;
        --verbosity) VERB="${2:?}"; shift ;;
        --dry-run)  DRY=1 ;;
        --oui|-y)   OUI=1 ;;
        --creer)    CREER=1 ;;
        -h|--help)  usage; exit 0 ;;
        *)          REST+=("$1") ;;
    esac
    shift
```

```
done
case "$CMD" in
    lecteur|lecteurs) cmd_lecteur ;;
    lire)             cmd_lire ;;
    programmer)       cmd_programmer ;;
    verifier)         cmd_verifier ;;
    hlr)              cmd_hlr ;;
    sap)              cmd_sap ;;
    shell)            cmd_shell "${REST[@]}" ;;
    ''|-h|--help)     usage; exit 0 ;;
    *)                die "commande inconnue : $CMD (lecteur, lire, programmer, verifier, hlr, sap, shell)" ;;
esac
```

3.434 /opt/GSM/osmo-operator/tools/osmo-supplements-firstboot.sh

2334 octets, 55 lignes → 55 lignes

```
#!/bin/bash
# osmo-supplements-firstboot.sh - les supplements choisis dans l installeur, au premier demarrage.
#
# [2026-10-24] La page « Supplements » de Calamares (installer/calamares/modules/
# packagechooser-supplements.conf) ne fait qu ecrire /etc/osmo-supplements :
#   OSMO_SUPPLEMENTS="--multi --opencl ..."   les drapeaux d addition.sh
#   HUB_IMAGE="..."                           l origine de l image operateur (multi)
# osmo-supplements.service lance ce script une fois le reseau monte. Reussi : le fichier
# devient /etc/osmo-supplements.fait et l unite se desactive. En echec, on reessaie au
# demarrage suivant, trois fois au plus (/etc/osmo-supplements.essais), puis
# /etc/osmo-supplements.echec : « sudo addition.sh » reste la, a la main.
# Journal : /var/log/osmo-supplements.log (et journalctl -u osmo-supplements).
set -u
CONF=/etc/osmo-supplements
ESSAIS=/etc/osmo-supplements.essais
LOG=/var/log/osmo-supplements.log
REPO="${OSMO_REPO:-/opt/GSM/osmo-operator}"
MAX="${OSMO_SUPPLEMENTS_ESSAIS:-3}"

_fin() {  # _fin <suffixe> : range la config et desactive l unite
    mv -f "$CONF" "$CONF.$1"
    rm -f "$ESSAIS"
    systemctl disable osmo-supplements.service >/dev/null 2>&1 || true
}

[ -f "$CONF" ] || exit 0
# shellcheck disable=SC1090
. "$CONF"
[ -n "${OSMO_SUPPLEMENTS:-}" ] || { _fin fait; exit 0; }

n=$(( $(cat "$ESSAIS" 2>/dev/null || echo 0) + 1 ))
echo "$n" > "$ESSAIS"

# network-online.target ne garantit pas Internet (DHCP lent, Wi-Fi pas encore associe) :
# on attend que GitHub reponde, cinq minutes au plus.
for _ in $(seq 1 60); do
    curl -fsS -m 5 -o /dev/null https://github.com && break
    sleep 5
done

echo "=== $(date '+%F %T') essai $n/$MAX : addition.sh $OSMO_SUPPLEMENTS ===" >>"$LOG"
# shellcheck disable=SC2086
# OSMO_COMPTE : le compte de l installeur (groupe docker, icones), personne n etant
# encore connecte ; a defaut, le premier compte humain (uid 1000 et plus).
[ -n "${OSMO_COMPTE:-}" ] || OSMO_COMPTE="$(getent passwd | awk -F: '$3>=1000 && $3<60000 {print $1; exit}')"
env HUB_IMAGE="${HUB_IMAGE:-}" OSMO_COMPTE="${OSMO_COMPTE:-}" bash "$REPO/addition.sh" $OSMO_SUPPLEMENTS >>"$LOG"
 ↪  2>&1 </dev/null
```

```
rc=$?
echo "=== $(date '+%F %T') fin : statut $rc ===" >>"$LOG"

if [ "$rc" -eq 0 ]; then
    _fin fait
elif [ "$n" -ge "$MAX" ]; then
    _fin echec
fi
exit "$rc"
```

3.435 /opt/GSM/osmo-operator/tools/osmo-telecom-boot.sh

3626 octets, 75 lignes → 75 lignes

```
#!/bin/bash
# ============================================================================
# osmo-telecom-boot.sh - applique la politique « aucun service telecom au boot »
#
# [2026-10-23] Pose services/80-osmo-operator.preset dans
# <racine>/usr/lib/systemd/system-preset/ puis desactive, parmi les unites
# PRESENTES, celles que ce preset vise (systemctl --root pour un rootfs d ISO).
# Rien n est arrete : un service en marche continue jusqu a son arret, il ne
# repartira simplement plus au demarrage.
#
#   tools/osmo-telecom-boot.sh                cette machine
#   tools/osmo-telecom-boot.sh --root DIR     un rootfs (chroot d ISO)
#   tools/osmo-telecom-boot.sh --garder U...  unites a laisser activees
#                                             (ex. osmo-interstp pour le role
#                                             interstp, osmo-banc si
#                                             OSMO_ISO_BANC=1)
#   tools/osmo-telecom-boot.sh --liste        dit seulement ce qui serait fait
# ============================================================================
set -uo pipefail
DIR="$(cd "$(dirname "$(readlink -f "$0")")/.." && pwd)"
PRESET="$DIR/services/80-osmo-operator.preset"
ROOT=""; LISTE=0; GARDER=()
while [ $# -gt 0 ]; do
    case "$1" in
        --root)   ROOT="${2%/}"; shift ;;
        --garder) shift; while [ $# -gt 0 ] && [ "${1#--}" = "$1" ]; do GARDER+=("${1%.service}"); shift; done;
         ↪  continue ;;
        --liste)  LISTE=1 ;;
        -h|--help) sed -n '2,19p' "$0"; exit 0 ;;
    esac
    shift
done
[ -f "$PRESET" ] || { echo "preset introuvable : $PRESET" >&2; exit 1; }

# les motifs « disable » du preset
mapfile -t MOTIFS < <(awk '$1 == "disable" {print $2}' "$PRESET")

# les unites presentes sur la cible, qui correspondent a un motif
unites=()
for d in "$ROOT/etc/systemd/system" "$ROOT/lib/systemd/system" "$ROOT/usr/lib/systemd/system"; do
    [ -d "$d" ] || continue
    for f in "$d"/*.service; do
        [ -e "$f" ] || continue
        u="$(basename "$f")"
        for m in "${MOTIFS[@]}"; do
            # shellcheck disable=SC2053
            [[ $u == $m ]] && { unites+=("$u"); break; }
        done
    done
done
```

```
mapfile -t unites < <(printf '%s\n' "${unites[@]}" | sort -u | grep -v '^$')

[ "$LISTE" = 1 ] || install -D -m644 "$PRESET" "$ROOT/usr/lib/systemd/system-preset/80-osmo-operator.preset"
sc=(systemctl); [ -n "$ROOT" ] && sc+=(--root="$ROOT")
n=0
for u in "${unites[@]}"; do
    for g in "${GARDER[@]}"; do [ "${u%.service}" = "$g" ] && continue 2; done
    if [ -n "$ROOT" ]; then
        # rootfs : active = un lien *.wants/ vers l unite (is-enabled hors systemd n est pas fiable)
        compgen -G "$ROOT/etc/systemd/system/*.wants/$u" >/dev/null || continue
        etat=enabled
    else
        etat="$("${sc[@]}" is-enabled "$u" 2>/dev/null)"
        case "$etat" in enabled|enabled-runtime|linked) ;; *) continue ;; esac
    fi
    if [ "$LISTE" = 1 ]; then echo "  desactiverait : $u ($etat)"; n=$((n + 1)); continue; fi
    if "${sc[@]}" disable "$u" >/dev/null 2>&1; then echo "  boot : $u desactive"; n=$((n + 1))
    else
        # chroot sans systemd : on retire les liens nous-memes (ce que fait disable)
        find "$ROOT/etc/systemd/system" -path "*.wants/$u" -delete 2>/dev/null && echo "  boot : $u desactive
        ↪  (liens)" && n=$((n + 1))
    fi
done
[ "${#GARDER[@]}" -gt 0 ] && echo "  boot : gardes actives : ${GARDER[*]}"
if [ "$LISTE" = 1 ]; then echo "  politique telecom au boot : $n unite(s) a desactiver"
else echo "  politique telecom au boot : $n unite(s) desactivee(s), preset 80-osmo-operator.preset pose"; fi
exit 0
```

3.436 /opt/GSM/osmo-operator/tools/osmo-topzone.py

16527 octets, 380 lignes → 380 lignes

```
#!/usr/bin/env python3
# osmo-topzone.py - LA ZONE HAUTE « LAB GSM », vivante et cliquable.
#
# Remplace la carte LAB GSM autrefois CUITE dans le fond d ecran
# (tools/wallpaper-render.py, desormais lance avec OSMO_LIVE_BANNER=1 pour ne
# plus la dessiner). C est une fenetre de type BUREAU (comme osmo-panel.py) posee
# EXACTEMENT sur la boite de la carte (320,60 .. 1430,560 en 1920x1080), qui
# affiche configs/conky/labgsm.html via WebKit.
#
# DEUX ROLES :
#   1. AU REPOS (rien d heberge) : la banniere anime ses 8 timeslots. Elle lit
#      $OSMO_FFT_DIR/timeslots.json (ecrit par tools/osmo-ts-probe.py : conf
#      osmo-bsc + VTY) et fait CLIGNOTER EN ORANGE tout timeslot qui porte des
#      bursts - un TCH ne clignote que s il y a de la voix. ARFCN / IMSI / A5
#      sont les valeurs LIVE. Le chronogramme du burst scintille avec l activite.
#   2. HEBERGEMENT : quand osmo-launcher.py envoie « en haut » une appli, il
#      ecrit "HOST <winid>" dans $XDG_RUNTIME_DIR/osmo-topzone.cmd. On fait alors
#      le FONDU de la banniere et on cale la fenetre de l appli sur la zone. Une
#      petite barre au-dessus (toujours devant) porte une CROIX de fermeture et
#      un bascule PLEIN ECRAN / RECTANGLE HAUT.
import json
import os
import subprocess
import sys
import time

# [2026-09-06] MEME PIEGE QUE osmo-panel.py / osmo-dino.py (voir leurs notes) :
# sous Wayland, GTK3 ignore move() et le type BUREAU ; une fois passe par
# XWayland, « toujours dessous » fait avaler les clics par la couche de bureau
# de GNOME Shell. Backend X11, fenetre normale.
X11_FORCE = os.environ.get("XDG_SESSION_TYPE") == "wayland" and bool(os.environ.get("DISPLAY"))
if X11_FORCE:
```

```
    os.environ["GDK_BACKEND"] = "x11"

import gi

gi.require_version("Gtk", "3.0")
gi.require_version("Gdk", "3.0")
gi.require_version("WebKit2", "4.1")
from gi.repository import Gdk, GLib, Gtk, WebKit2  # noqa: E402

# [2026-09-06] LA BANNIERE, ELLE, RESTE AU FOND. Contrairement a l encart et au
# lanceur, on la VEUT collee au bureau, derriere tout : c est un decor, pas un
# panneau de commande. Sur XWayland ca coute les clics (GNOME Shell range les
# fenetres de type BUREAU sous sa couche d icones, qui les avale) - donc
# l hebergement de fenetre dans la zone haute ne repond plus a la souris. C est
# le choix assume ici ; OSMO_TOPZONE_CLIQUABLE=1 pour reprendre une fenetre
# normale (devant, mais cliquable) si on a besoin de s en servir.
_CLIQUABLE = os.environ.get("OSMO_TOPZONE_CLIQUABLE") == "1"
TYPE_HINT = Gdk.WindowTypeHint.NORMAL if _CLIQUABLE else Gdk.WindowTypeHint.DESKTOP
KEEP_BELOW = not _CLIQUABLE

FW, FH = 1920, 1080
# [2026-09-06] ON ENLEVE LE VIDE DU BAS, PAS LE CONTENU.
# La boite (1110x500, soit 1644x740 sur l ecran du banc) etait bien plus haute
# que ce que la carte remplit : tout le bas etait du fond vide qui mangeait le
# bureau. Deux fausses pistes, ecartees :
#   - rapetisser la boite : la page est ecrite en unites vh/vw (#card fait
#     height:100%, les titres 8.5vh, les etiquettes 1.3vh...), donc raccourcir
#     la fenetre RAPETISSE tout l interieur - ce n est pas ce qu on veut ;
#   - zoomer la vue a 0.5 : meme resultat, l interieur maigrit.
# Ce qu on fait : la page est rendue A SA TAILLE D ORIGINE (les vh gardent la
# hauteur de reference ci-dessous, donc l interieur est identique a avant), le
# cadre s arrete apres son contenu (CSS injecte : #card passe en height:auto),
# et la fenetre se cale sur la hauteur reellement occupee, mesuree DANS la page
# une fois chargee. Plus de vide, et pas un pixel de contenu perdu.
TOP_BOX = (320, 60, 1110, 500)          # carte LAB GSM (x,y,w,h) - le RENDU
# La feuille qui coupe le vide. Attention au piege : #card est en
# « position:absolute; inset:0 » - ce n est pas une hauteur qui l etire mais son
# bottom:0. Lui poser height:auto ne change donc RIEN tant qu on ne relache pas
# le bas ; il faut les deux.
INJECT_CSS = "#card{bottom:auto!important;height:auto!important}"
REPO = os.environ.get("OSMO_REPO", "/opt/GSM/osmo-operator")
HTML = os.environ.get("OSMO_LABGSM_HTML", os.path.join(REPO, "configs/conky/labgsm.html"))
FFT_RUN = os.environ.get("OSMO_FFT_DIR", "/run/osmo-fft")


# ── OU LIRE : LA MEME REGLE QUE LA SONDE, ET RELUE A CHAQUE TOUR ──────────
# [2026-09-10] Ce chemin etait fige au demarrage, et sa regle n etait pas tout
# a fait celle de l ecrivain (tools/osmo-ts-probe.py, run_dir()) : la sonde se
# repliait sur $XDG_RUNTIME_DIR, jamais regarde ici. Deux regles pour un seul
# fichier, et la banniere restait en demonstration devant un banc qui portait
# des appels, sans une ligne d erreur nulle part. C est la MEME regle des deux
# cotes, mot pour mot : le premier repertoire qui existe et ou l on peut
# ecrire, entre $OSMO_FFT_DIR et /tmp - relue a chaque lecture, parce que
# /run/osmo-fft est un RuntimeDirectory qui va et vient avec son service.
def ts_json():
    for d in (FFT_RUN, "/tmp"):
        if os.path.isdir(d) and os.access(d, os.W_OK):
            return os.path.join(d, "timeslots.json")
    return "/tmp/timeslots.json"
RUN = os.environ.get("XDG_RUNTIME_DIR") or "/run/user/%d" % os.getuid()
CMD_FILE = os.path.join(RUN if os.path.isdir(RUN) else "/tmp", "osmo-topzone.cmd")
# [2026-10-12] L ETAT DU BANC, ECRIT PAR osmo-panel.py ("1" : une 2G ou une 4G
# tourne, "0" : rien). Banc a l arret, la banniere s efface et la zone ne porte
# que le titre « Telecom Operator » ; au lancement de la 2G, le titre fond vers
# la banniere (labgsm.html, osmoBanc). Fichier absent : au repos.
ETAT_FILE = os.path.join(RUN if os.path.isdir(RUN) else "/tmp", "osmo-banc.etat")
```

```
def banc_etat():
    try:
        return open(ETAT_FILE).read().strip() == "1"
    except OSError:
        return False


def screen_geom():
    disp = Gdk.Display.get_default()
    mon = disp.get_primary_monitor() or disp.get_monitor(0)
    g = mon.get_geometry()
    sw, sh = g.width, g.height
    try:
        from gi.repository import Gio
        mode = Gio.Settings.new("org.gnome.desktop.background").get_string("picture-options")
    except Exception:
        mode = "zoom"
    if mode == "stretched":
        sx, sy, ox, oy = sw / FW, sh / FH, 0, 0
    else:
        s = min(sw / FW, sh / FH) if mode == "scaled" else max(sw / FW, sh / FH)
        sx = sy = s
        ox, oy = (FW * s - sw) / 2, (FH * s - sh) / 2
    return (g.x, g.y, sw, sh), (sx, sy, ox, oy)


def box_to_screen(box):
    (gx, gy, sw, sh), (sx, sy, ox, oy) = screen_geom()
    bx, by, bw, bh = box
    return (gx + int(bx * sx - ox), gy + int(by * sy - oy), int(bw * sx), int(bh * sy))


def _place(winid, geom):
    x, y, w, h = geom
    subprocess.run(["wmctrl", "-i", "-r", winid, "-b",
                    "remove,maximized_vert,maximized_horz,fullscreen"], timeout=3)
    subprocess.run(["wmctrl", "-i", "-r", winid, "-e", f"0,{x},{y},{w},{h}"], timeout=3)


def _fullscreen(winid, on=True):
    subprocess.run(["wmctrl", "-i", "-r", winid, "-b",
                    ("add" if on else "remove") + ",fullscreen"], timeout=3)


def _close_win(winid):
    subprocess.run(["wmctrl", "-i", "-c", winid], timeout=3)


class Strip(Gtk.Window):
    """Barre de controle TOUJOURS DEVANT (croix + bascule plein ecran)."""
    def __init__(self, on_close, on_toggle):
        super().__init__(type=Gtk.WindowType.TOPLEVEL)
        self.set_decorated(False)
        self.set_keep_above(True)
        self.set_skip_taskbar_hint(True)
        self.set_skip_pager_hint(True)
        self.set_type_hint(Gdk.WindowTypeHint.UTILITY)
        self.set_accept_focus(False)
        prov = Gtk.CssProvider()
        prov.load_from_data(b".strip{background:rgba(19,16,24,0);border-radius:8px}"
                            b".strip button{color:#efe6d2;background:rgba(40,34,52,.5);text-shadow:0 1px 2px
                             ↪  rgba(0,0,0,.9);"
                            b"border:1px solid rgba(239,230,210,.25);border-radius:6px;"
                            b"margin:2px;padding:0 8px;font:11pt 'DejaVu Sans Mono'}"
```

```
                        b".strip button:hover{border-color:#ff9a3c}")
        Gtk.StyleContext.add_provider_for_screen(
            self.get_screen(), prov, Gtk.STYLE_PROVIDER_PRIORITY_APPLICATION)
        box = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=0)
        box.get_style_context().add_class("strip")
        self.b_mode = Gtk.Button(label="▭")
        self.b_mode.set_tooltip_text("plein écran / rectangle haut")
        self.b_mode.connect("clicked", lambda *_a: on_toggle())
        b_close = Gtk.Button(label="×")
        b_close.set_tooltip_text("fermer et rendre la bannière")
        b_close.connect("clicked", lambda *_a: on_close())
        box.pack_start(self.b_mode, False, False, 0)
        box.pack_start(b_close, False, False, 0)
        self.add(box)

    def show_at(self, x, y):
        self.show_all()
        self.set_keep_above(True)
        self.move(x, y)


class TopZone(Gtk.Window):
    def __init__(self):
        super().__init__(title="osmo-topzone")
        x, y, w, h = box_to_screen(TOP_BOX)
        self.box_geo = (x, y, w, h)
        self.hosted = None       # winid de l appli hebergee
        self.fs = False          # mode plein ecran de l appli hebergee
        self.set_type_hint(TYPE_HINT)
        self.set_decorated(False)
        self.set_resizable(False)
        self.set_keep_below(KEEP_BELOW)
        self.set_skip_taskbar_hint(True)
        self.set_skip_pager_hint(True)
        self.stick()
        self.set_default_size(w, h)
        self.set_size_request(w, h)
        self.move(x, y)
        screen = self.get_screen()
        visual = screen.get_rgba_visual()
        if visual and screen.is_composited():
            self.set_visual(visual)
        self.set_app_paintable(True)
        self.connect("destroy", Gtk.main_quit)

        ucm = WebKit2.UserContentManager()
        ucm.add_style_sheet(WebKit2.UserStyleSheet(
            INJECT_CSS, WebKit2.UserContentInjectedFrames.ALL_FRAMES,
            WebKit2.UserStyleLevel.USER, None, None))
        self.view = WebKit2.WebView.new_with_user_content_manager(ucm)
        # Une fois la page chargee, on lui demande la hauteur que la carte
        # occupe VRAIMENT et on rabat la fenetre dessus.
        self.view.connect("load-changed", self.on_load)
        self.view.set_background_color(Gdk.RGBA(0, 0, 0, 0))
        self.add(self.view)
        if os.path.exists(HTML):
            self.view.load_uri(GLib.filename_to_uri(HTML, None))
        else:
           self.view.load_html("<h1 style='color:#fff;font-family:sans-serif'>labgsm.html introuvable</h1>", None)

        self.strip = Strip(self.on_close, self.on_toggle)

        self._cmd_mtime = None
        self._etat_mtime = None
        self._etat = False           # ce que la page montre : False = le titre
        self.show_all()
```

```
    self.move(x, y)
    # [2026-09-06] LE TEMPS REEL SE PERD AUSSI ICI. On poussait l etat une
    # fois par seconde, en aveugle : ajoute a la seconde de la sonde, un
    # timeslot pouvait mettre deux secondes a clignoter. On regarde
    # maintenant le fichier cinq fois par seconde et on ne pousse QUE s il
    # a change - plus reactif, et moins de JS inutile qu avant.
    self._ts_mtime = None
    GLib.timeout_add(200, self.push_state)
    GLib.timeout_add(500, self.poll_cmd)

# ── rabattre la fenetre sur la hauteur utile ──────────────────────
def on_load(self, _view, event):
    if event != WebKit2.LoadEvent.FINISHED:
        return
    # l etat du banc tel qu il est, SANS fondu (la page part « au repos »)
    self._banc(banc_etat(), instant=True)
    # Un cheveu de marge sous le cadre, sinon la bordure du bas rase le
    # bord de la fenetre.
    self.view.run_javascript(
        "(function(){var c=document.getElementById('card')||document.body;"
        "return Math.ceil(c.getBoundingClientRect().bottom)+2;})()",
        None, self.on_height, None)

def on_height(self, view, res, _data):
    try:
        val = view.run_javascript_finish(res).get_js_value().to_double()
    except Exception as e:                      # page remplacee, JS coupe...
        print("[topzone] hauteur non mesuree : %s" % e, file=sys.stderr, flush=True)
        return
    h = int(val)
    x, y, w, h_full = self.box_geo
    if not (0 < h < h_full):                    # mesure absurde : on garde tout
        print("[topzone] hauteur mesuree %d ignoree (fenetre %d)" % (h, h_full),
              file=sys.stderr, flush=True)
        return
    print("[topzone] hauteur utile %d px (au lieu de %d)" % (h, h_full), flush=True)
    self.box_geo = (x, y, w, h)
    self.set_size_request(w, h)
    self.resize(w, h)
    self.move(x, y)

# ── au repos : pousser l etat des timeslots dans la page ──────────
def push_state(self):
    chemin = ts_json()
    try:
        mt = (chemin, os.stat(chemin).st_mtime)
    except OSError:
        mt = None
    if mt is not None and mt == self._ts_mtime:
        return True                     # rien de neuf depuis le dernier tour
    self._ts_mtime = mt
    try:
        with open(chemin) as f:
            state = f.read().strip()
    except OSError:
        state = ""
    if state:
        js = "window.__osmoLive=true; if(window.osmoUpdate){osmoUpdate(%s);}" % state
        try:
            self.view.run_javascript(js, None, None, None)
        except Exception:
            pass
    return True

def _fade(self, on):
    try:
```

```
        self.view.run_javascript("if(window.osmoFade)osmoFade(%s);" % ("true" if on else "false"),
                                 None, None, None)
    except Exception:
        pass

# ── titre « Telecom Operator » (repos) <-> banniere (banc debout) ──────────
def _banc(self, on, instant=False):
    self._etat = on
    try:
        self.view.run_javascript("if(window.osmoBanc)osmoBanc(%s,%s);"
                                 % ("true" if on else "false", "true" if instant else "false"),
                                 None, None, None)
    except Exception:
        pass

def poll_etat(self):
    try:
        mt = os.stat(ETAT_FILE).st_mtime
    except OSError:
        mt = None
    if mt == self._etat_mtime:
        return
    self._etat_mtime = mt
    on = banc_etat()
    if on != self._etat:
        print("[topzone] %s" % ("un banc tourne : banniere" if on else "banc a l arret : titre"), flush=True)
        self._banc(on)

# ── hebergement : lire le canal osmo-launcher -> osmo-topzone ──────────────
def poll_cmd(self):
    self.poll_etat()
    try:
        mt = os.stat(CMD_FILE).st_mtime
    except OSError:
        return True
    if mt == self._cmd_mtime:
        return True
    self._cmd_mtime = mt
    try:
        line = open(CMD_FILE).read().strip()
    except OSError:
        return True
    parts = line.split()
    if len(parts) >= 2 and parts[0] == "HOST":
        self.host(parts[-1], full=("FS" in parts))
    return True

def host(self, winid, full=False):
    self.hosted = winid
    self.fs = full
    self._fade(True)
    x, y, w, h = self.box_geo
    if full:
        _fullscreen(winid, True)
    else:
        _place(winid, self.box_geo)
    # la barre de controle, au coin haut-droit de la zone
    self.strip.b_mode.set_label("▭" if full else "⛶")
    self.strip.show_at(x + w - 74, max(0, y - 2))

def on_toggle(self):
    if not self.hosted:
        return
    self.fs = not self.fs
    if self.fs:
        _fullscreen(self.hosted, True)
```

```
            self.strip.b_mode.set_label("▭")
        else:
            _place(self.hosted, self.box_geo)
            self.strip.b_mode.set_label("□")
        x, y, w, h = self.box_geo
        self.strip.show_at(x + w - 74, max(0, y - 2))

    def on_close(self):
        if self.hosted:
            _close_win(self.hosted)
        self.hosted = None
        self.fs = False
        self.strip.hide()
        self._fade(False)


if __name__ == "__main__":
    TopZone()
    Gtk.main()
```

3.437 /opt/GSM/osmo-operator/tools/osmo-ts-probe.py

19196 octets, 459 lignes → 459 lignes

```
#!/usr/bin/env python3
# osmo-ts-probe.py - QUI PORTE DES BURSTS, MAINTENANT ? Ecrit l etat des 8
# timeslots que la banniere LAB GSM (configs/conky/labgsm.html, via
# osmo-topzone.py) fait clignoter en orange.
#
# Deux sources :
#   1. LA CONF osmo-bsc : phys_chan_config de chaque timeslot (CCCH+SDCCH4,
#      SDCCH8, TCH/F ...). Donne le TYPE, donc le libelle de chaque case et le
#      fait qu un TCH puisse porter de la VOIX. Cherchee dans, au choix :
#         $OSMO_BSC_CFG, /etc/osmocom/osmo-bsc.cfg, $OSMO_REPO/configs/osmo-bsc.cfg
#   2. LE VTY osmo-bsc (telnet 127.0.0.1 4242, `show lchan summary`) : quels
#      canaux logiques sont ETABLIS, sur quel timeslot, de quel type. Un TCH
#      etabli = de la voix -> le timeslot clignote. Pas de VTY (banc arrete) ->
#      bsc=false, la banniere passe en demonstration.
#
# Sortie : $OSMO_FFT_DIR/timeslots.json (defaut /run/osmo-fft, /tmp en repli
# - voir run_dir(), la MEME regle que le lecteur), reecrit chaque
# seconde, de forme :
#   {"bsc":true,"arfcn":"ARFCN 514 · DCS 1800",
#    "ts":[{"n":0,"type":"CCCH+SDCCH4","active":false,"voice":false}, ...]}
import json
import os
import re
import socket
import sys
import time

REPO = os.environ.get("OSMO_REPO", "/opt/GSM/osmo-operator")
# ── OU ECRIRE : LA MEME REGLE QUE LE LECTEUR, ET RELUE A CHAQUE TOUR ─────────
# [2026-09-10] Le repli etait « $XDG_RUNTIME_DIR sinon /tmp », alors que le
# lecteur (tools/osmo-topzone.py) ne regarde QUE $OSMO_FFT_DIR puis /tmp : sans
# /run/osmo-fft, la sonde ecrivait dans /run/user/1001 un fichier que PERSONNE
# n allait lire, et la banniere restait en demonstration sans qu aucune erreur
# ne le dise. Deux chemins pour un seul fichier, c est un de trop : la regle
# est ici et dans osmo-topzone.py, mot pour mot - le premier repertoire qui
# EXISTE ET OU L ON PEUT ECRIRE, entre $OSMO_FFT_DIR et /tmp.
#
# « ou l on peut ecrire » et pas seulement « qui existe » : /run/osmo-fft est
# cree par osmo-fft-snap.service (RuntimeDirectory, 0775, groupe sudo) mais
# tools/osmo-op.sh a longtemps pu le creer avant lui, en 0755 root:root - la
```

```
# sonde de la session ne pouvait alors plus jamais y ecrire.
#
# Et la regle est REEVALUEE A CHAQUE ECRITURE : ce repertoire est un
# RuntimeDirectory, il disparait quand l unite s arrete et revient quand elle
# repart. Une sonde qui aurait resolu son chemin une fois pour toutes ecrirait
# ensuite dans le vide jusqu au prochain redemarrage.
RUN_PREFERE = os.environ.get("OSMO_FFT_DIR", "/run/osmo-fft")


def run_dir():
    for d in (RUN_PREFERE, "/tmp"):
        if os.path.isdir(d) and os.access(d, os.W_OK):
            return d
    return "/tmp"
VTY_HOST = os.environ.get("OSMO_BSC_VTY_HOST", "127.0.0.1")
VTY_PORT = int(os.environ.get("OSMO_BSC_VTY_PORT", "4242"))
# [2026-09-06] LA BANNIERE SUIT UNE BTS PRECISE, ET UNE SEULE. Les huit cases
# sont la trame TDMA d UN porteur : celle de la BTS 0, TRX 0 - la BTS du banc.
# La lecture du VTY ne regardait que « Timeslot N » sans se soucier du reste de
# la ligne : sur un banc multi-operateur (plusieurs BTS declarees dans le meme
# osmo-bsc), un canal etabli sur la BTS 1 allumait la case de la BTS 0. Les
# cases suivaient donc « une » BTS, pas LA BTS.
BTS_NR = int(os.environ.get("OSMO_BTS_NR", "0"))
TRX_NR = int(os.environ.get("OSMO_TRX_NR", "0"))


def cfg_path():
    for p in (os.environ.get("OSMO_BSC_CFG"),
              "/etc/osmocom/osmo-bsc.cfg",
              os.path.join(REPO, "configs/osmo-bsc.cfg")):
        if p and os.path.exists(p):
            return p
    return None


def read_types():
    """[(n, phys_chan_config)] du premier trx du premier bts de la conf."""
    p = cfg_path()
    types = {}
    arfcn = None
    if not p:
        return types, arfcn
    try:
        txt = open(p, errors="replace").read()
    except OSError:
        return types, arfcn
    # Le bloc de LA bts suivie (BTS_NR), pas « la premiere venue » : sur une
    # conf multi-BTS les phys_chan_config different d une BTS a l autre, et les
    # libelles des cases doivent etre ceux du porteur qu on affiche.
    blocks = re.split(r'\n\s*bts\s+(\d+)', txt)
    block = txt
    # re.split avec un groupe rend [avant, "0", bloc0, "1", bloc1, ...]
    for i in range(1, len(blocks) - 1, 2):
        if int(blocks[i]) == BTS_NR:
            block = blocks[i + 1]
            break
    else:
        if len(blocks) > 2:
            block = blocks[2]                # pas de BTS_NR : la premiere
    # arfcn de la premiere trx (on ignore un placeholder de template __ARFCN__)
    m = re.search(r'\n\s*arfcn\s+(\d+)', block)
    if m:
        arfcn = m.group(1)
    # premier trx uniquement
    trx0 = re.split(r'\n\s*trx\s+\d+', block)
    scope = trx0[1] if len(trx0) > 1 else block
```

```
    # [2026-10-21] le phys_chan_config n est pas forcement la ligne qui SUIT
    # « timeslot N » : un commentaire (« ! ... ») entre les deux laissait la
    # case sans type (TS1 affiche « – » alors que la conf dit TCH/F). On prend
    # le premier phys_chan_config du bloc du timeslot, jusqu au timeslot suivant.
    for tm in re.finditer(r'\n\s*timeslot\s+(\d+)((?:\n(?!\s*timeslot\s)[^\n]*)*)', scope):
        pm = re.search(r'\n\s*phys_chan_config\s+(\S+)', tm.group(2))
        if pm:
            types[int(tm.group(1))] = pm.group(1)
    return types, arfcn


# ── LA CONNEXION VTY QUI RESTE OUVERTE ──────────────────────────────
# [2026-09-06] POURQUOI LES TIMESLOTS AVAIENT UNE SECONDE DE RETARD. Chaque
# tour de sonde OUVRAIT une connexion telnet, envoyait la commande, DORMAIT
# 250 ms en esperant que la reponse soit arrivee, puis fermait. Avec la periode
# d une seconde, un canal qui s etablit pouvait mettre plus d une seconde a
# faire clignoter sa case. On garde donc la connexion, et on lit jusqu au
# prompt du VTY au lieu de dormir : la reponse est traitee des qu elle est
# complete, en general en quelques millisecondes.
_PROMPT = re.compile(rb'[\r\n][A-Za-z0-9_.\-]+(?:\([^)]*\))?[>#]\s*$')


class Vty:
    def __init__(self):
        self.s = None

    def _connect(self):
        try:
            self.s = socket.create_connection((VTY_HOST, VTY_PORT), timeout=1.0)
            self.s.settimeout(1.0)
            self._read_until_prompt()          # la banniere d accueil
            return True
        except OSError:
            self.s = None
            return False

    def close(self):
        if self.s is not None:
            try:
                self.s.close()
            except OSError:
                pass
            self.s = None

    def _read_until_prompt(self, limit=262144):
        buf = b""
        while True:
            try:
                chunk = self.s.recv(4096)
            except socket.timeout:
                break                          # on rend ce qu on a
            except OSError:
                self.close()
                return None
            if not chunk:
                self.close()
                return None
            buf += chunk
            if _PROMPT.search(buf) or len(buf) > limit:
                break
        return buf

    def cmd(self, text):
        """Sortie d une commande, ou None si le VTY n est pas joignable."""
        if self.s is None and not self._connect():
            return None
```

```
        try:
            self.s.sendall(text.encode() + b"\r\n")
        except OSError:
            self.close()
            if not self._connect():            # une seule reprise
                return None
            try:
                self.s.sendall(text.encode() + b"\r\n")
            except OSError:
                self.close()
                return None
        out = self._read_until_prompt()
        if out is None:
            return None
        return out.decode(errors="replace")


VTY = Vty()


def vty_lchan():
    """Renvoie (ok, {ts: {'active':bool,'voice':bool}}) depuis le VTY osmo-bsc."""
    act = {}
    text = VTY.cmd("show lchan summary")
    if text is None:
        return False, act
    # lignes type : "BTS 0, TRX 0, Timeslot 2, ... Lchan 0, Type TCH/F, State ESTABLISHED ..."
    for line in text.splitlines():
        m = re.search(r'Timeslot\s+(\d+)', line)
        if not m:
            continue
        # La BTS et le TRX de la ligne : une ligne d une autre BTS (ou d un
        # autre porteur) ne concerne pas cette banniere. Une ligne qui ne les
        # nomme pas est gardee - vieux format de sortie, mieux vaut afficher
        # que perdre l information.
        mb = re.search(r'BTS\s+(\d+)', line)
        mx = re.search(r'TRX\s+(\d+)', line)
        if (mb and int(mb.group(1)) != BTS_NR) or (mx and int(mx.group(1)) != TRX_NR):
            continue
        ts = int(m.group(1))
        d = act.setdefault(ts, {"active": False, "voice": False})
        d["active"] = True
        typ = ""
        mt = re.search(r'Type\s+(\S+)', line)
        if mt:
            typ = mt.group(1).upper()
        state = line.upper()
        established = ("ESTABLISHED" in state or "ACTIVE" in state)
        if "TCH" in typ and established:
            d["voice"] = True
    return True, act


def vty_text(cmd):
    """Sortie brute d une commande VTY osmo-bsc, ou None si injoignable."""
    return VTY.cmd(cmd)


def vty_identity():
    """MCC/MNC, A5, arfcn, band LIVE depuis le VTY (le plus a jour). {} si rien."""
    ident = {}
    net = vty_text("show network")
    if net:
        m = re.search(r'Country Code[:\s]+(\d+).*?Network Code[:\s]+(\d+)', net, re.S)
        if m:
            ident["mcc"], ident["mnc"] = m.group(1), m.group(2)
```

```
        m = re.search(r'[Ee]ncryption[:\s]+A5/?(\d)', net)
        if m:
            ident["a5"] = m.group(1)
    bts = vty_text("show bts 0")
    if bts:
        m = re.search(r'\bARFCN[:\s]+(\d+)', bts) or re.search(r'\bC0[:\s]+(\d+)', bts)
        if m:
            ident["arfcn"] = m.group(1)
        m = re.search(r'[Bb]and[:\s]+(\S+)', bts)
        if m:
            ident["band"] = m.group(1)
    return ident


def net_from_cfg():
    """Repli : MCC/MNC/A5/band depuis le fichier de conf, placeholders ignores."""
    p = cfg_path()
    out = {}
    if not p:
        return out
    try:
        txt = open(p, errors="replace").read()
    except OSError:
        return out
    def grab(rx, key):
        m = re.search(rx, txt)
        if m and "__" not in m.group(1):
            out[key] = m.group(1)
    grab(r'\n\s*network country code\s+(\S+)', "mcc")
    grab(r'\n\s*mobile network code\s+(\S+)', "mnc")
    grab(r'\n\s*encryption\s+a5\s+(\S+)', "a5")
    grab(r'\n\s*band\s+(\S+)', "band")
    return out


def build():
    types, cfg_arfcn = read_types()
    if not types:
        # defaut raisonnable si pas de conf : la topologie classique du banc
        types = {0: "CCCH+SDCCH4", 1: "SDCCH8", 2: "TCH/F", 3: "TCH/F",
                 4: "TCH/F", 5: "TCH/F", 6: "TCH/F", 7: "TCH/F"}
    ok, act = vty_lchan()
    ts = []
    for n in range(8):
        a = act.get(n, {})
        ts.append({"n": n, "type": types.get(n, "NONE"),
                   "active": bool(a.get("active")), "voice": bool(a.get("voice"))})
    out = {"bsc": ok, "ts": ts}

    # ── LES VALEURS LIVE : ARFCN / IMSI (MCC-MNC) / A5 / bande ──────────────
    # Le VTY prime (etat reel du reseau qui tourne) ; le fichier de conf est le
    # repli ; le template du depot ne compte pas (placeholders __MCC__ ignores).
    ident = {}
    if cfg_arfcn and "__" not in cfg_arfcn:
        ident["arfcn"] = cfg_arfcn
    ident.update(net_from_cfg())
    if ok:
        ident.update(vty_identity())  # le live ecrase le fichier

    band = ident.get("band", "")
    band_txt = ""
    if band:
        b = str(band).upper().replace("DCS", "DCS ").replace("PCS", "PCS ")
        band_txt = " · " + b
    if ident.get("arfcn"):
        out["arfcn"] = "ARFCN %s%s" % (ident["arfcn"], band_txt)
```

```
    if ident.get("mcc") and ident.get("mnc"):
        out["imsi"] = "%s-%s" % (str(ident["mcc"]).zfill(3), str(ident["mnc"]).zfill(2))
    if ident.get("a5"):
        out["a5"] = str(ident["a5"])
    out.update(live_info(ts))
    return out


# ── LE BANC EN TEMPS REEL, POUR LA BANNIERE ──────────────────────────
# [2026-10-01] La banniere ne montrait que les timeslots. Elle recoit
# maintenant le banc entier - les demons du coeur, la radio, les abonnes du
# HLR, les appels en cours, depuis quand la pile est debout - lus a bas cout
# (/proc, la base sqlite du HLR en lecture seule) toutes les 2 s : le reste de
# la sonde tourne quatre fois par seconde, et des demons qui vivent des heures
# n ont pas besoin d etre recomptes a ce rythme. Les noms sont ceux que le
# noyau garde (comm, 15 caracteres) - « osmo-sip-connector » s y lit
# « osmo-sip-connec », comme dans tools/conky-osmo-status.sh.
COEUR = (("HLR", "osmo-hlr"), ("MSC", "osmo-msc"), ("BSC", "osmo-bsc"), ("STP", "osmo-stp"),
         ("MGW", "osmo-mgw"), ("BTS", "osmo-bts-trx"), ("SIP", "osmo-sip-connector"),
         ("SMSC", "proto-smsc-daemon"), ("PBX", "asterisk"))
RADIO = (("MOBILE", "mobile"), ("TRXCON", "trxcon"), ("QEMU", "qemu-system-arm"),
         ("GAPK", "osmo-gapk"))
HLR_DB = os.environ.get("OSMO_HLR_DB", "/var/lib/osmocom/hlr.db")
_live = {"quand": 0.0, "val": {}}


def _procs():
    """(noms comm, lignes de commande, {comm: pid}) de tous les processus."""
    comms, cmds, pids = set(), [], {}
    for p in os.listdir("/proc"):
        if not p.isdigit():
            continue
        try:
            with open("/proc/%s/comm" % p) as f:
                c = f.read().strip()
            with open("/proc/%s/cmdline" % p, "rb") as f:
                cmds.append(f.read(300).replace(b"\0", b" ").decode(errors="replace"))
        except OSError:
            continue
        comms.add(c)
        pids.setdefault(c, int(p))
    return comms, cmds, pids


def _depuis(pid):
    """Depuis combien de secondes ce processus tourne (0 si illisible)."""
    try:
        with open("/proc/%d/stat" % pid) as f:
            champs = f.read().rsplit(")", 1)[1].split()
        start = int(champs[19]) / os.sysconf("SC_CLK_TCK")
        with open("/proc/uptime") as f:
            up = float(f.read().split()[0])
        return max(0, int(up - start))
    except (OSError, ValueError, IndexError):
        return 0


def live_info(ts):
    now = time.time()
    if now - _live["quand"] < 2.0:
        return _live["val"]
    val = {}
    try:
        comms, cmds, pids = _procs()
        core = {n: (c[:15] in comms) for n, c in COEUR}
        radio = {n: (c[:15] in comms) for n, c in RADIO}
```

```
        phy = ""
        if any("qemu-system-arm" in c for c in cmds):
            phy = "Calypso émulé (QEMU)"
        if any("fake_trx.py" in c for c in cmds):
            phy = "faketrx"
        if any(re.search(r"pont/pont(_dsp)?\.py", c) for c in cmds):
            phy = (phy + " + pont") if phy else "pont"
        # Le banc de l ISO n a PAS d osmo-trx : le transceiver est faketrx
        # (fake_trx.py) et pont (pont.py / pont_dsp.py) - on les montre tels
        # quels. GAPK : le binaire osmo-gapk n existe que le temps d un appel ;
        # hors appel le veilleur gapk-start.sh EST l etat nominal.
        radio["FAKETRX"] = any("fake_trx.py" in c for c in cmds)
        radio["PONT"] = any(re.search(r"pont/pont(_dsp)?\.py", c) for c in cmds)
        radio["GAPK"] = radio.get("GAPK", False) or any("gapk-start.sh" in c for c in cmds)
        subs = {}
        if core["HLR"] and os.path.exists(HLR_DB):
            try:
                import sqlite3
                con = sqlite3.connect("file:%s?mode=ro" % HLR_DB, uri=True, timeout=0.2)
                tot = con.execute("select count(*) from subscriber").fetchone()[0]
                att = con.execute("select count(*) from subscriber where vlr_number is not null "
                                  "and vlr_number != ''").fetchone()[0]
                con.close()
                subs = {"total": tot, "attached": att}
            except Exception:  # noqa: BLE001 - base verrouillee, illisible : pas d abonnes
                subs = {}
        depuis = 0
        for c in ("osmo-bsc", "osmo-stp", "osmo-msc"):
            if c in pids:
                depuis = _depuis(pids[c])
                break
        val = {"core": core, "radio": radio, "phy": phy, "subs": subs,
               "calls": sum(1 for t in ts if t.get("voice")), "up": depuis}
    except Exception as e:  # noqa: BLE001 - la banniere vit sans ces infos
        print("[ts-probe] infos banc : %s" % e, file=sys.stderr, flush=True)
    _live.update(quand=now, val=val)
    return val


def main():
    # 0.25 s : assez court pour que l allumage d un canal se voie « tout de
    # suite » a l oeil, assez long pour ne pas marteler le VTY d osmo-bsc (4
    # « show lchan summary » par seconde sur une connexion deja ouverte).
    period = float(os.environ.get("OSMO_TS_PERIOD", "0.25"))
    try:
        os.makedirs(RUN_PREFERE, exist_ok=True)
    except OSError:
        pass                                # run_dir() se rabattra sur /tmp
    tmp = None
    dernier = (None, 0.0)                   # (message, quand) - voir plus bas
    while True:
        try:
            out = build()
            dest = os.path.join(run_dir(), "timeslots.json")
            # ── UN TEMPORAIRE PAR PROCESSUS ──────────────────────────────
            # [2026-09-10] Deux sondes ont tourne en meme temps - un widget
            # rescape d une session X precedente, que le gardien du bureau
            # comptait pour vivant (cf. osmo-desktop-panel, osmo_vivant) - et
            # elles partageaient « timeslots.json.tmp » : chacune renommait le
            # temporaire de l autre, d ou un journal rempli de
            # « [Errno 2] ... .tmp -> ... .json » quatre fois par seconde. Le
            # PID dans le nom rend l ecriture atomique meme a plusieurs.
            tmp = "%s.%d.tmp" % (dest, os.getpid())
            with open(tmp, "w") as f:
                json.dump(out, f)
            os.replace(tmp, dest)
```

```
            tmp = None
        except Exception as e:  # noqa: BLE001 - une sonde ne doit jamais mourir
            # Un temporaire orphelin ne doit pas rester dans /run.
            if tmp:
                try:
                    os.unlink(tmp)
                except OSError:
                    pass
                tmp = None
            # ── LE MEME DEFAUT NE SE DIT PAS QUATRE FOIS PAR SECONDE ────────
            # Une panne durable (repertoire non accessible) noyait le journal
            # sous des milliers de lignes identiques. On dit la premiere, puis
            # une par minute tant que rien ne change.
            msg = "%s (destination %s)" % (e, run_dir())
            now = time.time()
            if msg != dernier[0] or now - dernier[1] > 60:
                print("[ts-probe] %s" % msg, file=sys.stderr, flush=True)
                dernier = (msg, now)
        time.sleep(period)


if __name__ == "__main__":
    main()
```

3.438 /opt/GSM/osmo-operator/tools/osmo-wallpaper.sh

18262 octets, 324 lignes → 324 lignes

```
#!/bin/bash
# osmo-wallpaper.sh - pose le fond d ecran du banc avec ses deux bandes dessinees
# (Calvin & Hobbes en bas, une BD geek au hasard en haut)
# du jour incruste. Appele par osmo-wallpaper.service (timer quotidien + boot),
# par l autostart GNOME, et a la main :
#     sudo /usr/local/sbin/osmo-wallpaper
#
# 1. Le strip du jour vient de gocomics.com, comme le fait le gist "hellogist"
#    (curl de la page du jour, balise og:image, curl de l image). Sans reseau,
#    on reprend le dernier strip en cache ; sans cache, le fond part sans strip.
# 2. tools/wallpaper-render.py compose photo + carte LAB GSM + strip.
# 3. Le PNG est ecrit sous DEUX noms : le fichier fixe que le schema GNOME
#    designe par defaut (gsm-lab-wallpaper.png, ce que voit une session qui
#    s ouvre), et un fichier DATE que l on pousse dans chaque session ouverte
#    via gsettings - une URI differente force GNOME Shell a recharger, ce
#    qu une reecriture du meme fichier ne garantit pas.
set -u
# Surchargeables par l environnement (tests hors machine : OSMO_WP_OUT=... etc.).
REPO="${OSMO_WP_REPO:-/opt/GSM/osmo-operator}"
RENDER="$REPO/tools/wallpaper-render.py"
# [2026-09-04] Le format du fichier n est plus fige. La photo du pylone est
# livree TELLE QUELLE (aujourd hui un PNG en pleine definition) : imposer
# .jpg obligerait a la re-encoder, donc a la degrader, pour satisfaire un nom.
# On prend le premier configs/wallpaper/tower.* qui existe.
TOWER="$(ls "$REPO"/configs/wallpaper/tower.* 2>/dev/null | head -1)"
OUT="${OSMO_WP_OUT:-/usr/share/backgrounds/gsm-lab-wallpaper.png}"
DATED_DIR="${OSMO_WP_DATED_DIR:-/usr/share/backgrounds/osmo-lab}"
CACHE="${OSMO_WP_CACHE:-/var/cache/osmo-wallpaper}"
NO_SESSION="${OSMO_WP_NO_SESSION:-0}"
DAY="$(date +%F)"
STRIP="$CACHE/calvin_${DAY}.gif"
UA="Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
H=(-H "Accept: text/html,*/*;q=0.8" -H "Accept-Language: fr,en;q=0.7" -H "Referer: https://www.gocomics.com/")

[ "$(id -u)" -eq 0 ] || { echo "root requis : sudo $0" >&2; exit 1; }
[ -f "$RENDER" ] && [ -f "$TOWER" ] || { echo "[wallpaper] $RENDER ou $TOWER absent" >&2; exit 1; }
```

```
mkdir -p "$CACHE" "$DATED_DIR"

# ── 1. L IMAGE DU JOUR ──────────────────────────────────────────────────
# [2026-09-04] UNE SEULE SOURCE, ET PLUS D IMAGE DU TOUT. gocomics.com a mis un
# controle anti-robot (Bunny Shield) devant ses pages : curl ne recoit plus que
# 2 ko de « Establishing a secure connection », jamais la balise og:image. La
# methode du gist (curl + grep og:image) ne marche donc plus - ni ici, ni a la
# main. Le fond partait sans rien, tous les jours.
#
# On tire donc au sort parmi PLUSIEURS sources, et on prend la premiere qui
# repond. Toutes ont ete verifiees au curl nu, sans cle ni compte :
#   calvin   gocomics.com     - garde en tete de liste : elle marche encore
#                               depuis les reseaux que le bouclier laisse passer
#   xkcd     xkcd.com         - JSON officiel (info.0.json), le plus fiable
#   apod     apod.nasa.gov    - l astronomie du jour, HTML simple
#   bing     bing.com         - la photo du jour, JSON HPImageArchive
#   turnoff  turnoff.us       - bandes dessinees d informaticien (Daniel Stori)
#
# Une source differente a chaque lancement : le fond change meme quand le strip
# du jour n a pas bouge. OSMO_WP_SOURCE=xkcd en force une ; OSMO_WP_SOURCES
# restreint la liste.
# [2026-09-08] DEUX IMAGES, DEUX CADRES. L encart du fond s est coupe en deux
# (tools/wallpaper-render.py) : le BAS reste a Calvin & Hobbes, c est le cadre
# du banc 2G ; le HAUT, celui du banc 4G, porte une bande dessinee GEEK tiree
# au sort a chaque lancement - un xkcd pris au hasard dans toute la
# collection, ou le dernier turnoff.us. Chaque cadre a sa liste de sources et
# son tirage ; a defaut de reseau, chacun reprend sa derniere image en cache.
#   OSMO_WP_SOURCES_BAS   (calvin)                  OSMO_WP_SOURCE_BAS  en force une
#   OSMO_WP_SOURCES_HAUT  (xkcd_alea turnoff xkcd)  OSMO_WP_SOURCE_HAUT en force une
# OSMO_WP_SOURCES / OSMO_WP_SOURCE (l ancien reglage, une seule image) valent
# desormais pour le HAUT.
SOURCES_BAS="${OSMO_WP_SOURCES_BAS:-calvin}"
SOURCES_HAUT="${OSMO_WP_SOURCES_HAUT:-${OSMO_WP_SOURCES:-xkcd_alea turnoff xkcd}}"
[ -n "${OSMO_WP_SOURCE_BAS:-}" ] && SOURCES_BAS="$OSMO_WP_SOURCE_BAS"
[ -n "${OSMO_WP_SOURCE_HAUT:-${OSMO_WP_SOURCE:-}}" ] && SOURCES_HAUT="${OSMO_WP_SOURCE_HAUT:-$OSMO_WP_SOURCE}"

_get() { curl -sL --max-time 30 -A "$UA" "$@" 2>/dev/null; }
# Une image, et pas une page d erreur deguisee : `file` regarde le contenu.
_pose_image() {   # url fichier referer
    _get -H "Referer: ${3:-}" -o "$2.tmp" "$1" || { rm -f "$2.tmp"; return 1; }
    if [ -s "$2.tmp" ] && file -b "$2.tmp" | grep -qiE 'image|GIF|PNG|JPEG'; then
        mv -f "$2.tmp" "$2"; return 0
    fi
    rm -f "$2.tmp"; return 1
}

# Chaque source ecrit l image dans $1 et sa ligne de credit dans $2. Une source
# qui tire au sort pose dans NOM_RETENU un nom qui porte le tirage (« xkcd-1234 »)
# : le fichier de cache et l URI du fond en heritent, donc l ecran change.
NOM_RETENU=""
src_calvin() {
    local page url
    page="$(_get "${H[@]}" "https://www.gocomics.com/calvinandhobbes/$(date +%Y/%m/%d)")"
    if [ -z "$page" ]; then
        echo "[wallpaper] calvin : gocomics.com injoignable (reseau, DNS, proxy ?)"; return 1
    fi
    if printf '%s' "$page" | grep -qiE 'bunny-shield|Establishing a secure
     ↪  connection|challenge-platform|cf-browser-verification'; then
        echo "[wallpaper] calvin : gocomics.com rend une page anti-robot ($(printf '%s' "$page" | wc -c) octets)"
        return 1
    fi
    # L URL directe de featureassets d abord (robuste au HTML), og:image ensuite
    # - c est la balise que lit le gist « hellogist ».
    url="$(printf '%s' "$page" | grep -oE 'https://featureassets\.gocomics\.com/assets/[0-9a-f]+' | head -1)"
    [ -n "$url" ] || url="$(printf '%s' "$page" | grep -oP '<meta property="og:image" content="\K[^"]+' | head -1)"
    [ -n "$url" ] || { echo "[wallpaper] calvin : page recue, aucune image dedans (le site a change ?)"; return 1; }
```

```
    _pose_image "$url" "$1" "https://www.gocomics.com/" || return 1
    printf 'Calvin & Hobbes  ·  Bill Watterson  ·  %s  ·  gocomics.com\n' "$DAY" > "$2"
}
_xkcd_json() {   # json -> image + credit ($1 json, $2 fichier, $3 credit)
    local url num titre
    url="$(printf '%s' "$1" | grep -oP '"img":\s*"\K[^"]+')"
    [ -n "$url" ] || return 1
    num="$(printf '%s' "$1" | grep -oP '"num":\s*\K[0-9]+')"
    titre="$(printf '%s' "$1" | grep -oP '"safe_title":\s*"\K[^"]+')"
    _pose_image "$url" "$2" "https://xkcd.com/" || return 1
    printf 'xkcd #%s  ·  %s  ·  Randall Munroe  ·  xkcd.com\n' "$num" "$titre" > "$3"
    NOM_RETENU="xkcd-$num"
}
src_xkcd() { _xkcd_json "$(_get https://xkcd.com/info.0.json)" "$1" "$2"; }
# Un xkcd AU HASARD dans toute la collection : le dernier numero dit combien il
# y en a, et on en tire un. Le 404 n existe pas (c est une blague de l auteur).
src_xkcd_alea() {
    local dernier num
    dernier="$(_get https://xkcd.com/info.0.json | grep -oP '"num":\s*\K[0-9]+')"
    [ -n "$dernier" ] && [ "$dernier" -gt 1 ] || return 1
    num=$(( (RANDOM * 32768 + RANDOM) % dernier + 1 ))
    [ "$num" -eq 404 ] && num=405
    _xkcd_json "$(_get "https://xkcd.com/$num/info.0.json")" "$1" "$2"
}
src_apod() {
    local page src
    page="$(_get https://apod.nasa.gov/apod/astropix.html)"
    src="$(printf '%s' "$page" | grep -oiP '<img[^>]+src="\K[^"]+' | head -1)"
    [ -n "$src" ] || return 1
    case "$src" in http*) ;; *) src="https://apod.nasa.gov/apod/${src#/}" ;; esac
    _pose_image "$src" "$1" "https://apod.nasa.gov/apod/" || return 1
    printf 'NASA · Astronomy Picture of the Day  ·  %s  ·  apod.nasa.gov\n' "$DAY" > "$2"
}
src_bing() {
    local j url cr
    j="$(_get "https://www.bing.com/HPImageArchive.aspx?format=js&idx=0&n=1&mkt=fr-FR")"
    url="$(printf '%s' "$j" | grep -oP '"url":"\K[^"]+')"
    [ -n "$url" ] || return 1
    case "$url" in http*) ;; *) url="https://www.bing.com${url}" ;; esac
    cr="$(printf '%s' "$j" | grep -oP '"copyright":"\K[^"]+')"
    _pose_image "$url" "$1" "https://www.bing.com/" || return 1
    printf 'Bing · image du jour  ·  %s\n' "${cr:-bing.com}" > "$2"
}
src_turnoff() {
    local url
    url="$(_get https://turnoff.us/feed.xml | grep -oP '<img[^>]+src=.\K[^"'"'"' ]+' | head -1)"
    [ -n "$url" ] || return 1
    _pose_image "$url" "$1" "https://turnoff.us/" || return 1
    printf 'turnoff.us  ·  Daniel Stori  ·  turnoff.us\n' > "$2"
}

# Ordre aleatoire : `shuf` s il est la, un melange maison sinon (busybox, image
# elaguee). Sans cela on interrogerait toujours la meme source en premier, et
# « une source differente a chaque lancement » n aurait aucun sens.
# La source du coup precedent ($2) passe en DERNIER : sans cela, un tirage sur
# cinq retombait sur elle et le fond ne changeait pas - alors que c est
# justement a ca qu on voit qu un nouveau banc a demarre. Si elle est la seule
# disponible, elle ressort quand meme (elle est en queue, pas exclue).
_melange() {   # "sources" derniere -> l ordre, sur une ligne
    local cand="" s ordre=""
    for s in $1; do [ "$s" = "$2" ] || cand="$cand $s"; done
    [ -n "$cand" ] || cand="$1"
    if command -v shuf >/dev/null 2>&1; then
        ordre="$(printf '%s\n' $cand | shuf | tr '\n' ' ')"
    else
        for s in $cand; do
```

```
            if [ $((RANDOM % 2)) -eq 0 ]; then ordre="$s $ordre"; else ordre="$ordre $s"; fi
        done
    fi
    [ -n "$2" ] && ordre="$ordre $2"
    printf '%s' "$ordre"
}

# Le tirage d un cadre : la premiere source de la liste ($1) qui repond, en
# evitant celle du coup precedent ($2). Resultat dans R_STRIP / R_CREDIT /
# R_SRC ; 1 si aucune ne repond.
tirer() {
    local s f c
    R_STRIP=""; R_CREDIT=""; R_SRC=""
    for s in $(_melange "$1" "$2"); do
        f="$CACHE/strip_${DAY}_${s}.img"; c="$CACHE/strip_${DAY}_${s}.credit"
        # Deja telecharge aujourd hui : on ne redemande pas au site. Pas pour
        # un tirage au sort (xkcd_alea) : la, c est un nouveau numero a chaque fois.
        case "$s" in
            *_alea) ;;
            *) if [ -s "$f" ]; then
                   R_STRIP="$f"; R_CREDIT="$(cat "$c" 2>/dev/null || true)"; R_SRC="$s"
                   echo "[wallpaper] image du $DAY : source ${s} (deja en cache)"; return 0
               fi ;;
        esac
        NOM_RETENU=""
        if declare -F "src_$s" >/dev/null && "src_$s" "$f" "$c"; then
            if [ -n "$NOM_RETENU" ] && [ "$NOM_RETENU" != "$s" ]; then
                mv -f "$f" "$CACHE/strip_${DAY}_${NOM_RETENU}.img"
                mv -f "$c" "$CACHE/strip_${DAY}_${NOM_RETENU}.credit" 2>/dev/null
                f="$CACHE/strip_${DAY}_${NOM_RETENU}.img"; c="$CACHE/strip_${DAY}_${NOM_RETENU}.credit"
                s="$NOM_RETENU"
            fi
            R_STRIP="$f"; R_CREDIT="$(cat "$c" 2>/dev/null || true)"; R_SRC="$s"
            echo "[wallpaper] image du $DAY : source ${s}"; return 0
        fi
    done
    return 1
}
# Repli : la derniere image en cache dont le nom repond au motif ($1), en
# ecartant un fichier ($2, l image de l autre cadre). Vide s il n y en a pas.
_dernier_cache() {
    local f
    for f in $(ls -1t $CACHE/$1 2>/dev/null); do
        [ -s "$f" ] && [ "$f" != "$2" ] && { printf '%s' "$f"; return 0; }
    done
    return 1
}

DERNIERE_BAS="$(awk -F= '/^SOURCE=/{print $2}' "$CACHE/strip.state" 2>/dev/null)"
DERNIERE_HAUT="$(awk -F= '/^SOURCE_HAUT=/{print $2}' "$CACHE/strip.state" 2>/dev/null)"

# ── LE BAS : Calvin & Hobbes ──────────────────────────────────────────
STRIP_BAS=""; CREDIT_BAS=""; SRC_BAS=""
if tirer "$SOURCES_BAS" "$DERNIERE_BAS"; then
    STRIP_BAS="$R_STRIP"; CREDIT_BAS="$R_CREDIT"; SRC_BAS="$R_SRC"
else
    # gocomics derriere son bouclier : le dernier Calvin en cache, quelle que
    # soit sa date (calvin_*.gif : le nom d avant le tirage au sort).
    STRIP_BAS="$(_dernier_cache 'strip_*_calvin.img' '' || _dernier_cache 'calvin_*.gif' '' || true)"
    if [ -n "$STRIP_BAS" ]; then
        CREDIT_BAS="$(cat "${STRIP_BAS%.img}.credit" 2>/dev/null || true)"; SRC_BAS="calvin"
        echo "[wallpaper] bas : calvin ne repond pas - dernier Calvin en cache : ${STRIP_BAS##*/}"
    elif tirer "apod bing" ""; then
        STRIP_BAS="$R_STRIP"; CREDIT_BAS="$R_CREDIT"; SRC_BAS="$R_SRC"
        echo "[wallpaper] bas : aucun Calvin, ni en ligne ni en cache - image du jour a la place"
    else
```

```
        STRIP_BAS="$(_dernier_cache 'strip_*.img' '' || true)"
        [ -n "$STRIP_BAS" ] && CREDIT_BAS="$(cat "${STRIP_BAS%.img}.credit" 2>/dev/null || true)"
    fi
fi

# ── LE HAUT : la BD geek, au hasard ──────────────────────────────────
STRIP_HAUT=""; CREDIT_HAUT=""; SRC_HAUT=""
if tirer "$SOURCES_HAUT" "$DERNIERE_HAUT"; then
    STRIP_HAUT="$R_STRIP"; CREDIT_HAUT="$R_CREDIT"; SRC_HAUT="$R_SRC"
else
    STRIP_HAUT="$(_dernier_cache 'strip_*_xkcd*.img' "$STRIP_BAS" || _dernier_cache 'strip_*_turnoff.img'
    ↪  "$STRIP_BAS" \
                || _dernier_cache 'strip_*.img' "$STRIP_BAS" || true)"
    if [ -n "$STRIP_HAUT" ]; then
        CREDIT_HAUT="$(cat "${STRIP_HAUT%.img}.credit" 2>/dev/null || true)"
        SRC_HAUT="$(basename "$STRIP_HAUT" .img | sed 's/^strip_[0-9-]*_//')"
        echo "[wallpaper] haut : aucune source ne repond - derniere BD en cache : ${STRIP_HAUT##*/}"
    else
        echo "[wallpaper] haut : aucune source ne repond et le cache est vide - cadre sans image."
        echo "[wallpaper] Deposez n importe quelle image dans $CACHE/strip_${DAY}_local.img et relancez."
    fi
fi
STRIP="$STRIP_BAS"; SRC_RETENUE="$SRC_BAS"

args=()
if [ -n "$STRIP_BAS" ] && [ -s "$STRIP_BAS" ]; then
    args+=(--strip "$STRIP_BAS")
    [ -n "$CREDIT_BAS" ] && args+=(--credit "$CREDIT_BAS")
fi
if [ -n "$STRIP_HAUT" ] && [ -s "$STRIP_HAUT" ]; then
    args+=(--strip-haut "$STRIP_HAUT")
    [ -n "$CREDIT_HAUT" ] && args+=(--credit-haut "$CREDIT_HAUT")
fi
args+=(--date "$DAY")
# ── CE QUE L ENCART DOIT SAVOIR ──────────────────────────────────────
# [2026-09-04] Sans image, le fond n a rien a montrer a cet endroit : le cadre
# est vide. L encart (tools/osmo-fft-snap.py) compose le banc EN TRANSPARENCE
# par-dessus ce cadre - il laissait donc transparaitre du vide, et le spectre
# comme le mobile.log y perdaient en lisibilite pour rien. On lui dit ce qu il
# y a derriere lui ; sans image, il passe en opacite pleine.
# STRIP/SOURCE = le bas (le nom historique), STRIP_HAUT/SOURCE_HAUT = le haut.
# FOND = le rendu compose : c est LUI que l encart decoupe quand il existe ;
# sans cette ligne (fond brut, timer inactif) l encart montre l illustration
# du banc entiere. [2026-10-01]
printf 'STRIP=%s\nDATE=%s\nSOURCE=%s\nSTRIP_HAUT=%s\nSOURCE_HAUT=%s\nFOND=%s\n' \
    "$([ -n "$STRIP_BAS" ] && [ -s "$STRIP_BAS" ] && echo oui || echo non)" "$DAY" "${SRC_BAS:-}" \
    "$([ -n "$STRIP_HAUT" ] && [ -s "$STRIP_HAUT" ] && echo oui || echo non)" "${SRC_HAUT:-}" \
    "$OUT" \
    > "$CACHE/strip.state"
# Le cache ne garde que les 14 dernieres images (deux par lancement).
ls -1t "$CACHE"/strip_*.img 2>/dev/null | tail -n +15 | while read -r _old; do
    rm -f "$_old" "${_old%.img}.credit"
done
ls -1t "$CACHE"/calvin_*.gif 2>/dev/null | tail -n +15 | xargs -r rm -f

# ── 2. le rendu ──────────────────────────────────────────────────────
# LE NOM PORTE LA SOURCE, ET C EST CE QUI FAIT CHANGER L ECRAN.
# [2026-09-04] Le fichier date s appelait gsm-lab-<jour>.png. Deux rendus le
# meme jour ecrivaient donc le MEME chemin - et l URI poussee aux sessions
# (etape 3) ne changeait pas d un poil. Or GNOME Shell ne recharge que sur
# changement d URI : reecrire le fichier sous son ancien nom ne repeint rien.
# Le banc pouvait tirer une nouvelle source a chaque relance, l ecran gardait
# l image du premier demarrage jusqu au lendemain.
# Le nom porte donc la source retenue : elle change a chaque relance (voir le
# tirage plus haut, qui evite celle du coup precedent), donc l URI change, donc
# l ecran suit.
```

```
DATED="$DATED_DIR/gsm-lab-${DAY}${SRC_BAS:+-$SRC_BAS}${SRC_HAUT:+-$SRC_HAUT}.png"
python3 "$RENDER" --tower "$TOWER" "${args[@]}" --out "$DATED" || exit 1
cp -f "$DATED" "$OUT.tmp" && mv -f "$OUT.tmp" "$OUT"
chmod 644 "$OUT" "$DATED"
# On garde les trois derniers rendus. Le fichier en cours n est JAMAIS efface :
# une session qui vient de recevoir son URI le lit encore.
ls -1t "$DATED_DIR"/gsm-lab-*.png 2>/dev/null | tail -n +4 | grep -vxF "$DATED" | xargs -r rm -f

# ── 3. les sessions ouvertes ──────────────────────────────────────
# Une session GNOME lit le schema a l ouverture ; une session deja ouverte ne
# suit que dconf. On pousse l URI datee dans chacune (root sur la cle live,
# l utilisateur Calamares sur le disque).
[ "$NO_SESSION" = "1" ] && exit 0
for bus in /run/user/*/bus; do
    [ -S "$bus" ] || continue
    uid="${bus#/run/user/}"; uid="${uid%/bus}"
    user="$(id -nu "$uid" 2>/dev/null)" || continue
    runuser -u "$user" -- env DBUS_SESSION_BUS_ADDRESS="unix:path=$bus" XDG_RUNTIME_DIR="/run/user/$uid" \
        sh -c "gsettings set org.gnome.desktop.background picture-uri 'file://$DATED' ; \
               gsettings set org.gnome.desktop.background picture-uri-dark 'file://$DATED'" \
        2>/dev/null && echo "[wallpaper] session de $user : $DATED" && _pushed=1
done
# DING (icones du bureau) garde parfois l image precedente par-dessus le fond :
# relance (l extension GNOME le redemarre aussitot).
[ "${_pushed:-0}" = "1" ] && pkill -f "extensions/ding@rastersoft.com/app/ding.js" 2>/dev/null
exit 0
```

3.439 /opt/GSM/osmo-operator/tools/osmo-wg.sh

6138 octets, 160 lignes → 160 lignes

```
#!/bin/bash
# ============================================================================
# osmo-wg.sh - LE TUNNEL WIREGUARD DU WAN (wiki/WAN.md §3) EN UNE COMMANDE.
#
# [2026-10-24] Le wiki faisait poser wg0.conf a la main sur chaque noeud : cles,
# adresses, pairs, pare-feu. Le plan d adressage est fixe (10.99.0.0/24) :
#   10.99.0.1   le hub (inter-STP co-loge), le seul qui ECOUTE (udp 51820)
#   10.99.0.N   les autres noeuds, qui appellent le hub
# Une commande par noeud, et une ligne a recopier sur le hub par noeud ajoute.
# Les cles sont generees ICI, sur la machine : jamais dans l ISO (toutes les
# cles du live seraient les memes).
#
#   wg 1          (= wg 10.99.0.1)       ce noeud devient le hub
#   wg N HUB_IP[:PORT] HUB_PUB           ce noeud devient 10.99.0.N (N = 2..254)
#                                        et rejoint le hub ; imprime la ligne
#                                        « wg peer ... » a taper sur le hub
#   wg peer CLE_PUBLIQUE N               (sur le hub) accepte ce noeud
#   wg up | wg down                      remonte / arrete le tunnel deja pose
#   wg status | wg pub                   N ou 10.99.0.N, partout au choix
# « wg » est la fonction de /etc/profile.d/97-osmo-wg.sh : elle aiguille ces
# formes ici, tout le reste (wg show, wg genkey...) va au vrai wg.
# ============================================================================
set -u

IF="${OSMO_WG_IF:-wg0}"
PORT="${OSMO_WG_PORT:-51820}"
NET=10.99.0
HUB="$NET.1"
DIR=/etc/wireguard
CONF="$DIR/$IF.conf"

RED='\033[0;31m'; GREEN='\033[0;32m'; CYAN='\033[0;36m'; NC='\033[0m'
_ok()  { echo -e "  ${GREEN}✓${NC} $*"; }
```

```
_say() { echo -e "  ${CYAN}→${NC} $*"; }
_err() { echo -e "  ${RED}✗${NC} $*" >&2; }

usage() { sed -n '13,20p' "$0" | sed 's/^# \{0,1\}//'; exit "${1:-1}"; }

case "${1:-}" in ""|-h|--help) usage 0 ;; esac
if [ "$(id -u)" -ne 0 ]; then
    exec sudo "$0" "$@"
fi
command -v wg >/dev/null 2>&1 && command -v wg-quick >/dev/null 2>&1 \
    || { _err "wireguard-tools absent : apt install wireguard-tools"; exit 1; }

# _ip_ok 10.99.0.N : N dans 1..254
_ip_ok() {
    [[ "$1" =~ ^10\.99\.0\.([0-9]{1,3})$ ]] && [ "${BASH_REMATCH[1]}" -ge 1 ] && [ "${BASH_REMATCH[1]}" -le 254 ]
}

_cles() {
    install -d -m700 "$DIR"
    if [ ! -s "$DIR/$IF.priv" ]; then
        (umask 077; wg genkey > "$DIR/$IF.priv")
        _ok "cle privee generee ($DIR/$IF.priv)"
    fi
    wg pubkey < "$DIR/$IF.priv" > "$DIR/$IF.pub"
}

# SCTP (M3UA, SGs) passe dans le tunnel : on l ouvre au sous-reseau, pas plus.
_parefeu() {
    iptables -C INPUT -i "$IF" -s "$NET.0/24" -j ACCEPT 2>/dev/null \
        || iptables -I INPUT -i "$IF" -s "$NET.0/24" -j ACCEPT
    if [ "${1:-}" = hub ]; then
        iptables -C INPUT -p udp --dport "$PORT" -j ACCEPT 2>/dev/null \
            || iptables -I INPUT -p udp --dport "$PORT" -j ACCEPT
    fi
    if command -v ufw >/dev/null 2>&1 && ufw status 2>/dev/null | grep -q '^Status: active'; then
        ufw allow in on "$IF" from "$NET.0/24" >/dev/null
        [ "${1:-}" = hub ] && ufw allow "$PORT/udp" >/dev/null
    fi
}

_monter() {
    systemctl enable "wg-quick@$IF" >/dev/null 2>&1
    systemctl restart "wg-quick@$IF" || { _err "wg-quick@$IF ne demarre pas : journalctl -u wg-quick@$IF"; exit 1;
     ↪  }
    _ok "$IF monte, et a chaque demarrage (wg-quick@$IF)"
}

_ecrire() {  # _ecrire <contenu> : remplace wg0.conf, l ancien garde en .bak
    [ -f "$CONF" ] && cp -a "$CONF" "$CONF.bak" && _say "ancien $CONF garde en $CONF.bak"
    systemctl stop "wg-quick@$IF" 2>/dev/null
    (umask 077; printf '%s\n' "$1" > "$CONF")
}

hub() {
    _cles
    # SaveConfig : les pairs ajoutes par « wg peer » survivent a un redemarrage
    _ecrire "[Interface]
Address    = $HUB/24
ListenPort = $PORT
PrivateKey = $(cat "$DIR/$IF.priv")
SaveConfig = true"
    _parefeu hub
    _monter
    echo
    _say "HUB $HUB pret. Sur chaque autre noeud :"
    echo "      wg N <IP publique de cette machine>:$PORT $(cat "$DIR/$IF.pub")      (N = 2..254 : 10.99.0.N)"
```

```
}

noeud() {
    local ip="$1" ep="$2" pub="$3"
    [[ "$ep" == *:* ]] || ep="$ep:$PORT"
    [[ "$pub" =~ ^[A-Za-z0-9+/]{42,43}=$ ]] || { _err "cle publique du hub invalide : $pub"; exit 1; }
    _cles
    _ecrire "[Interface]
Address    = $ip/24
PrivateKey = $(cat "$DIR/$IF.priv")

[Peer]
# le hub ($HUB)
PublicKey  = $pub
Endpoint   = $ep
AllowedIPs = $NET.0/24
PersistentKeepalive = 25"
    _parefeu
    _monter
    echo
    _say "NOEUD $ip pret. Sur le hub ($HUB), tapez :"
    echo "      wg peer $(cat "$DIR/$IF.pub") ${ip##*.}"
    _say "puis ici : ping -c3 $HUB"
}

peer() {
    local pub="$1" ip="$2"
    [ -f "$CONF" ] && grep -q "^Address *= *$HUB/" "$CONF" \
        || { _err "ce noeud n est pas le hub ($HUB) : « wg 1 » d abord"; exit 1; }
    wg show "$IF" >/dev/null 2>&1 || _monter
    wg set "$IF" peer "$pub" allowed-ips "$ip/32" || exit 1
    wg-quick save "$IF" >/dev/null 2>&1
    _ok "pair $ip accepte (et enregistre dans $CONF)"
}

# « wg 5 » = « wg 10.99.0.5 » (et « wg peer CLE 5 »)
_ip() { [[ "$1" =~ ^[0-9]{1,3}$ ]] && echo "$NET.$1" || echo "$1"; }
case "${1:-}" in
    peer) [ $# -ge 3 ] && set -- "$1" "$2" "$(_ip "$3")" "${@:4}" ;;
    [0-9]*) set -- "$(_ip "$1")" "${@:2}" ;;
esac

case "${1:-}" in
    ""|-h|--help) usage 0 ;;
    up)     [ -f "$CONF" ] || { _err "pas de $CONF : wg 1 (hub) ou wg N ... (noeud)"; exit 1; }
            _monter; wg show "$IF" | sed -n '1,4p' ;;
    status) wg show "$IF" 2>/dev/null || _say "$IF absent" ;;
    down)   systemctl disable --now "wg-quick@$IF" && _ok "$IF arrete (et plus au demarrage)" ;;
    pub)    [ -s "$DIR/$IF.pub" ] && cat "$DIR/$IF.pub" || { _err "pas de cle : wg 1 (hub) ou wg N ... (noeud)";
     ↪  exit 1; } ;;
    peer)
        [ $# -eq 3 ] && _ip_ok "$3" && [ "$3" != "$HUB" ] || usage
        peer "$2" "$3" ;;
    *)
        _ip_ok "$1" || usage
        if [ "$1" = "$HUB" ]; then
            [ $# -eq 1 ] || usage
            hub
        else
            [ $# -eq 3 ] || { _err "un noeud doit connaitre le hub : wg ${1##*.} HUB_IP[:PORT]
             ↪  CLE_PUBLIQUE_DU_HUB"; exit 1; }
            noeud "$1" "$2" "$3"
        fi ;;
esac
```

3.440 /opt/GSM/osmo-operator/tools/osmo-wireshark-root.sh

7717 octets, 123 lignes → 123 lignes

```
#!/bin/bash
# osmo-wireshark-root - Wireshark en root sur le banc : 2G, 4G et le telephone
# (signalisation, appel, SMS, data) d un coup.
#
# [2026-09-08] UNE SEULE ICONE, ET ELLE ECOUTE DEJA LE BON TRAFIC. Wireshark
# ouvert « a la main » demandait de choisir l interface, puis de taper le
# filtre, puis de relancer en root parce que la capture etait refusee
# (wireshark-common/install-setuid est a false, expres). Ici :
#   - root par pkexec, donc l INVITE DE MOT DE PASSE GRAPHIQUE de l agent
#     polkit de GNOME (--disable-internal-agent interdit l agent texte) ;
#     xhost autorise root a ouvrir sa fenetre sur l affichage X11 de la
#     session, QT_QPA_PLATFORM=xcb la force par X meme sous Wayland ;
#   - capture immediate (-k) sur toutes les interfaces (-i any : la boucle
#     locale ou vit tout le coeur, plus le reseau ou partent GSMTAP et
#     l Abis d un vrai BTS) avec le filtre BPF du banc (-f, ce qui est
#     ENREGISTRE) :
#       SCTP en entier   M3UA d osmo-stp (2905), SGs (29118), S1AP (36412),
#                        Diameter s il passe en SCTP - la signalisation A, Lb, S1 ;
#       GSM              GSMTAP udp 4729 (l air, et les logs osmocom en gsmtap),
#                        Abis/IPA tcp 3002-3003, Gb udp 23000-23001 (osmo-pcu),
#                        GTP udp 2123/2152 (sgsn <-> ggsn), MGCP udp 2427-2428,
#                        SIP 5060-5061, RTP d osmo-mgw (4002-16001) et
#                        d asterisk (30000-30199) ;
#       LTE              S1AP par le SCTP, GTP-C/U udp 2123/2152 (mme, sgwc,
#                        smf, upf), PFCP udp 8805, Diameter tcp 3868 (S6a, Gx),
#                        SBI http tcp 7777 (nrf <-> smf).
#     Hors filtre, exprès : les VTY (tcp 42xx), le ZeroMQ radio de srsRAN
#     (tcp 2000/2001 : des echantillons IQ, des Mo par seconde), Prometheus.
#   - un FILTRE D AFFICHAGE (-Y, ce qui est MONTRE : tout reste dans la
#     capture, on l enleve d un clic dans la barre) qui ne garde que le
#     reseau 2G / 4G et la signalisation du telephone :
#       2G   GSMTAP (Um : RR, MM, CC, SMS, GMM/SM du telephone), LAPDm,
#            Abis RSL/OML, A (SCTP/M3UA/SCCP/BSSAP/BSSMAP/DTAP), GSUP
#            (osmo-hlr), Gb (NS/BSSGP/LLC/SNDCP), GTP, MGCP, SMPP ;
#       4G   S1AP + NAS-EPS (EMM/ESM), GTPv2 (S11), GTP-U, PFCP, Diameter
#            (S6a, Gx), SBI HTTP/2, SGs (CSFB et SMS over SGs), et LTE RRC /
#            MAC / RLC / PDCP quand on ouvre un pcap de srsRAN ;
#       appel   DTAP CC, MGCP, SIP, RTP / RTCP / DTMF (rtpevent) ;
#       SMS     gsm_sms partout ou il passe (DTAP, SGs, NAS), SMPP.
#     Sans le bruit qui noie tout ca : SACK / HEARTBEAT du SCTP, BEAT du
#     M3UA, Device-Watchdog du Diameter, Echo GTP / GTPv2, Heartbeat PFCP,
#     et sur l air les System Information et les Measurement Report.
#   - des BOUTONS DE FILTRE dans la barre de Wireshark (Banc, 2G, 4G, Appel,
#     SMS, Data, Attach) pour ne voir qu une chose : poses dans le profil de
#     root (~root/.config/wireshark/dfilter_buttons) s il n y en a pas deja.
#   - avec des arguments (un .pcap ouvert depuis le bureau, ou les options que
#     passe launch.sh), on ne touche a rien : ils sont transmis tels quels.
# /usr/local/bin/wireshark est un lien vers ce script : « wireshark » au
# clavier comme a l icone, c est celui-ci. Le vrai binaire reste /usr/bin.
# OSMO_WS_FILTRE (BPF), OSMO_WS_DFILTRE (affichage ; vide = tout montrer) et
# OSMO_WS_IFACE pour changer le filtre, l affichage ou l interface.
set -u
WS=/usr/bin/wireshark
[ -x "$WS" ] || { command -v zenity >/dev/null 2>&1 && \
    zenity --error --text="Wireshark introuvable (apt install wireshark)" 2>/dev/null; exit 1; }

# Ce qui est enregistre (BPF).
FILTRE="${OSMO_WS_FILTRE:-sctp\
 or udp port 4729 or tcp port 3002 or tcp port 3003 or udp port 23000 or udp port 23001\
 or udp port 2123 or udp port 2152 or udp port 2427 or udp port 2428\
 or port 5060 or port 5061 or udp portrange 4002-16001 or udp portrange 30000-30199\
 or udp port 8805 or tcp port 3868 or tcp port 7777}"
```

```
IFACE="${OSMO_WS_IFACE:-any}"

# Ce qui est montre (filtre d affichage Wireshark). Les parentheses : d abord
# les protocoles du banc, puis, retranche, le bruit des battements de coeur.
D_PROTOS="sctp || m3ua || sccp || bssap || gsm_a.bssmap || gsm_a.dtap || gsm_a.rr || gsm_a.ccch || gsm_a.sacch ||
 ↪  gsm_a.gm || gsm_sms || gsm_map || tcap || gsup || sgsap || smpp\
 || gsmtap || lapdm || gsm_abis_rsl || gsm_abis_oml\
 || gprs-ns || bssgp || llcgprs || sndcp || gtp || gtpv2 || pfcp\
 || s1ap || nas-eps || lte_rrc || mac-lte || rlc-lte || pdcp-lte || diameter || http2\
 || mgcp || sip || rtp || rtcp || rtpevent"
D_BRUIT="(sctp && !(sctp.chunk_type == 0) && sctp.chunk_type in {3, 4, 5})\
 || (m3ua.message_class == 3 && m3ua.message_type in {3, 6})\
 || diameter.cmd.code == 280\
 || gtp.message in {1, 2} || gtpv2.message_type in {1, 2} || pfcp.msg_type in {1, 2}\
 || gsm_a.dtap.msg_rr_type in {0x00, 0x02, 0x03, 0x05, 0x06, 0x07, 0x15, 0x19, 0x1a, 0x1b, 0x1c, 0x1d, 0x1e}"
DFILTRE="${OSMO_WS_DFILTRE-($D_PROTOS) && !($D_BRUIT)}"

# Les boutons de la barre de filtre, dans le profil de root (c est lui qui
# capture). Un fichier deja la (l utilisateur a fait les siens) est respecte.
osmo_ws_boutons() {
    local f="${HOME:-/root}/.config/wireshark/dfilter_buttons"
    [ -e "$f" ] && return 0
    mkdir -p "${f%/*}" 2>/dev/null || return 0
    cat > "$f" <<BTN
# Boutons de filtre du banc osmo-operator (osmo-wireshark-root).
"TRUE","Banc","$DFILTRE","Reseau 2G + 4G et telephone, sans les battements de coeur"
"TRUE","2G","(gsmtap || lapdm || gsm_abis_rsl || gsm_abis_oml || bssap || gsm_a.bssmap || gsm_a.dtap || gsup ||
 ↪  gprs-ns || bssgp || llcgprs || sndcp || mgcp || smpp) && !($D_BRUIT)","Um, Abis, A, Gb, GSUP, MGCP"
"TRUE","4G","(s1ap || nas-eps || gtpv2 || pfcp || diameter || http2 || sgsap || lte_rrc || mac-lte || pdcp-lte) &&
 ↪  !($D_BRUIT)","S1, S11, Sx, S6a/Gx, SBI, SGs"
"TRUE","Appel","gsm_a.dtap.protocol_discriminator == 3 || mgcp || sip || rtpevent || sgsap ||
 ↪  nas-eps.nas_msg_emm_type == 0x4c || gsm_a.bssmap.msgtype in {0x01, 0x02, 0x03, 0x20, 0x21, 0x22, 0x52}","CC
 ↪  (DTAP), MGCP, SIP, DTMF, CSFB (SGs, Extended Service Request), paging, assignation et liberation BSSMAP"
"TRUE","SMS","gsm_sms || gsm_a.dtap.protocol_discriminator == 9 || smpp || sgsap.msg_type in {0x07, 0x08}","SMS
 ↪  en DTAP, SGs, NAS et SMPP"
"TRUE","Data","gsm_a.gm || llcgprs || sndcp || bssgp || gprs-ns || gtp || gtpv2 || pfcp ||
 ↪  nas-eps.nas_msg_esm_type || gsm_a.dtap.protocol_discriminator in {8, 10}","GMM/SM, Gb, GTP, PFCP, ESM"
"TRUE","Attach","gsm_a.dtap.protocol_discriminator == 5 || gsm_a.dtap.msg_gmm_type || nas-eps.nas_msg_emm_type ||
 ↪  gsup || diameter.cmd.code in {316, 318} || gsm_a.bssmap.msgtype in {0x52, 0x57, 0x20}","MM, GMM, EMM, GSUP,
 ↪  S6a (ULR/AIR), Paging / Complete Layer 3 / Clear"
BTN
    return 0
}

if [ $# -eq 0 ]; then
    set -- -i "$IFACE" -k -f "$FILTRE"
    [ -n "$DFILTRE" ] && set -- "$@" -Y "$DFILTRE"
fi

if [ "$(id -u)" -eq 0 ]; then
    osmo_ws_boutons
    exec "$WS" "$@"
fi

# Pas root : on se relance nous-memes en root (pour poser les boutons dans le
# profil de root), puis Wireshark.
MOI="$(readlink -f "$0" 2>/dev/null || echo "$0")"
[ -x "$MOI" ] || MOI="$WS"
command -v xhost >/dev/null 2>&1 && xhost +SI:localuser:root >/dev/null 2>&1
if command -v pkexec >/dev/null 2>&1; then
    exec pkexec --disable-internal-agent env \
        DISPLAY="${DISPLAY:-:0}" \
        XAUTHORITY="${XAUTHORITY:-$HOME/.Xauthority}" \
        WAYLAND_DISPLAY="${WAYLAND_DISPLAY:-}" \
        XDG_RUNTIME_DIR="${XDG_RUNTIME_DIR:-}" \
        QT_QPA_PLATFORM=xcb \
```

```
        HOME=/root \
        "$MOI" "$@"
fi
exec sudo -E HOME=/root "$MOI" "$@"
```

3.441 /opt/GSM/osmo-operator/tools/overlay-draw.py

24887 octets, 635 lignes → 635 lignes

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# overlay-draw.py - dessin vectoriel façon Inkscape, en OVERLAY transparent
# TOUJOURS AU PREMIER PLAN par-dessus le bureau. Tout tient dans UNE fenêtre :
# le canevas plein écran + une barre d'outils verticale collée EN BAS À DROITE.
#
# Feutre (outil par défaut) : on MAINTIENT le clic gauche pour tracer à main
# levée ; le trait reste au premier plan et il est effaçable (gomme, ou la case
# « tout effacer » du coin bas-droit).
#
# Barre d'outils (de bas en haut, collée au bord droit et au bord bas) :
#   [C]  Tout effacer   (case rouge, dans le coin)
#   Gomme · Couleur (cliquer = couleur suivante) · − · +  (épaisseur)
#   Ellipse · Rectangle · Segment · Feutre · Déplacer · Annuler
#   Traverser (clic-à-travers) · Quitter
#
# Molette   = épaisseur du trait.
# Clavier   : b feutre · l segment · r rect · e ellipse · x gomme · s déplacer
#             z annuler · Suppr efface la sélection · Espace traverser · Échap/q quitter
#
# X11 + compositeur (GNOME/mutter) requis pour la transparence par pixel.
import gi
gi.require_version('Gtk', '3.0')
from gi.repository import Gtk, Gdk, GLib
import cairo
import math
import copy
import time
import os

GRAB_R = 14
HIT_TOL = 8
BS = 46          # côté d'un bouton de la barre
GAP = 6          # espace entre boutons
MARGIN = 12      # marge aux bords droit / bas
DOT_R = 24       # rayon de la pastille de rappel (état dormant)
DOT_MARGIN = 22  # marge de la pastille aux bords
HOLD_MS = 220    # durée de maintien du 2e clic pour réveiller depuis la pastille

SWATCHES = [
    (0.00, 0.82, 1.00), (1.00, 0.58, 0.00), (0.20, 0.90, 0.35),
    (1.00, 0.25, 0.35), (1.00, 0.90, 0.10), (0.75, 0.45, 1.00),
    (1.00, 1.00, 1.00), (0.05, 0.05, 0.05),
]

# ordre de la barre, du COIN bas-droit vers le HAUT
TOOLBAR = ["clear", "eraser", "color", "width_dn", "width_up",
           "ellipse", "rect", "line", "pencil", "select", "undo",
           "pass", "quit"]
TOOL_IDS = {"pencil", "line", "rect", "ellipse", "eraser", "select", "node"}


def dist(ax, ay, bx, by):
    return math.hypot(ax - bx, ay - by)
```

```
def dist_point_seg(px, py, ax, ay, bx, by):
    dx, dy = bx - ax, by - ay
    L2 = dx * dx + dy * dy
    if L2 == 0:
        return dist(px, py, ax, ay)
    t = max(0.0, min(1.0, ((px - ax) * dx + (py - ay) * dy) / L2))
    return dist(px, py, ax + t * dx, ay + t * dy)


def obj_bbox(o):
    xs = [p[0] for p in o["pts"]]; ys = [p[1] for p in o["pts"]]
    return min(xs), min(ys), max(xs), max(ys)


def obj_hit(o, x, y, tol):
    t = o["pts"]; typ = o["type"]; tol = tol + o["width"] / 2.0
    if typ in ("line", "path"):
        if len(t) == 1:
            return dist(x, y, t[0][0], t[0][1]) <= tol
        return any(dist_point_seg(x, y, a[0], a[1], b[0], b[1]) <= tol
                   for a, b in zip(t, t[1:]))
    x0, y0, x1, y1 = obj_bbox(o)
    if o.get("fill") and x0 - tol <= x <= x1 + tol and y0 - tol <= y <= y1 + tol:
        return True
    if typ == "rect":
        v = (abs(x - x0) <= tol or abs(x - x1) <= tol) and (y0 - tol <= y <= y1 + tol)
        h = (abs(y - y0) <= tol or abs(y - y1) <= tol) and (x0 - tol <= x <= x1 + tol)
        return v or h
    if typ == "ellipse":
        cx, cy = (x0 + x1) / 2, (y0 + y1) / 2
        rx, ry = max((x1 - x0) / 2, 1), max((y1 - y0) / 2, 1)
        d = math.sqrt(((x - cx) / rx) ** 2 + ((y - cy) / ry) ** 2)
        return abs(d - 1.0) <= (tol / min(rx, ry) + 0.06)
    return False


def obj_translate(o, dx, dy):
    o["pts"] = [(px + dx, py + dy) for px, py in o["pts"]]


def _hex(c):
    return "#%02x%02x%02x" % (int(c[0] * 255), int(c[1] * 255), int(c[2] * 255))


class Overlay(Gtk.Window):
    def __init__(self):
        super().__init__(type=Gtk.WindowType.TOPLEVEL)
        self.set_title("overlay-draw")
        self.set_app_paintable(True)
        screen = self.get_screen()
        vis = screen.get_rgba_visual()
        self.has_alpha = vis is not None
        if self.has_alpha:
            self.set_visual(vis)
        self.set_decorated(False)
        self.set_keep_above(True)
        self.set_skip_taskbar_hint(True)
        self.set_skip_pager_hint(True)
        self.stick()

        # état du document
        self.objects = []
        self.selected = None
        self.tool = "pencil"           # le feutre par défaut
        self.color_idx = 0
```

```
    self.color = SWATCHES[0]
    self.width = 4.0
    self.fill = False
    self.passthrough = False
    self.undo_stack = []
    self.redo_stack = []
    self.status = ""
    self.status_until = 0
    self.dormant = False           # rangé dans la pastille du coin
    self._dot_pressed = False      # 2e clic de la pastille maintenu ?

    # état d'interaction
    self.drag = None
    self.start = None
    self.temp = None
    self.pen_pts = None
    self.buttons = []              # [(id, x, y, w, h)] recalculé au rendu

    self.add_events(
        Gdk.EventMask.BUTTON_PRESS_MASK | Gdk.EventMask.BUTTON_RELEASE_MASK
        | Gdk.EventMask.POINTER_MOTION_MASK | Gdk.EventMask.KEY_PRESS_MASK
        | Gdk.EventMask.SCROLL_MASK)
    self.connect("draw", self.on_draw)
    self.connect("button-press-event", self.on_press)
    self.connect("button-release-event", self.on_release)
    self.connect("motion-notify-event", self.on_motion)
    self.connect("key-press-event", self.on_key)
    self.connect("scroll-event", self.on_scroll)
    self.connect("destroy", Gtk.main_quit)

    self.fullscreen()
    self.show_all()
    GLib.timeout_add(300, lambda: (self.set_keep_above(True), False)[1])

# ── barre d'outils : géométrie et clics ──────────────────────────
def layout_buttons(self, W, H):
    self.buttons = []
    x = W - MARGIN - BS
    for i, bid in enumerate(TOOLBAR):
        y = H - MARGIN - BS - i * (BS + GAP)
        self.buttons.append((bid, x, y, BS, BS))

def button_at(self, x, y):
    for bid, bx, by, bw, bh in self.buttons:
        if bx <= x <= bx + bw and by <= y <= by + bh:
            return bid
    return None

def toolbar_region(self):
    reg = cairo.Region()
    for _bid, bx, by, bw, bh in self.buttons:
        reg.union(cairo.RectangleInt(int(bx), int(by), int(bw), int(bh)))
    return reg

def full_region(self):
    # ce binding refuse None pour « pas de forme » : on redonne l'entree a
    # toute la fenetre avec une region pleine (equivalent a aucune forme).
    a = self.get_allocation()
    return cairo.Region(cairo.RectangleInt(0, 0, max(a.width, 1), max(a.height, 1)))

def do_button(self, bid):
    if bid in TOOL_IDS:
        self.tool = bid
        self.finish_pen()
    elif bid == "clear":
        self.clear_all()
```

```
    elif bid == "color":
        self.color_idx = (self.color_idx + 1) % len(SWATCHES)
        self.color = SWATCHES[self.color_idx]
        self.apply_style()
    elif bid == "width_dn":
        self.set_width(self.width - 1)
    elif bid == "width_up":
        self.set_width(self.width + 1)
    elif bid == "undo":
        self.undo()
    elif bid == "pass":
        self.set_passthrough(not self.passthrough)
    elif bid == "quit":
        self.go_dormant()
    self.queue_draw()

# ── état dormant : tout se range dans une pastille au coin bas-droit ─────
def dot_center(self):
    a = self.get_allocation()
    return a.width - DOT_MARGIN - DOT_R, a.height - DOT_MARGIN - DOT_R

def dot_region(self):
    cx, cy = self.dot_center()
    r = DOT_R + 6
    return cairo.Region(cairo.RectangleInt(int(cx - r), int(cy - r),
                                           int(2 * r), int(2 * r)))

def go_dormant(self):
    # la croix (X) ne quitte pas : elle range. Le dessin disparaît, seule
    # la pastille reste, toujours au-dessus. Le travail est gardé en mémoire.
    self.dormant = True
    self.drag = None
    self.temp = None
    self.pen_pts = None
    gw = self.get_window()
    if gw is not None:
        gw.input_shape_combine_region(self.dot_region(), 0, 0)
    self.queue_draw()

def wake(self):
    self.dormant = False
    self.passthrough = False
    gw = self.get_window()
    if gw is not None:
        gw.input_shape_combine_region(self.full_region(), 0, 0)
    self.set_keep_above(True)
    self.queue_draw()

def _dot_wake_if_held(self):
    # appelé HOLD_MS après le 2e clic : on ne reveille que s'il est encore
    # enfonce -> « double-clic dont le deuxieme appui reste long ».
    if self.dormant and self._dot_pressed:
        self.wake()
    return False

# ── helpers document ──────────────────────────────────────────────────
def new_obj(self, typ, pts):
    return {"type": typ, "pts": pts, "stroke": self.color,
            "width": self.width, "fill": self.color if self.fill else None}

def topmost_at(self, x, y):
    for o in reversed(self.objects):
        if obj_hit(o, x, y, HIT_TOL):
            return o
    return None
```

```
def set_width(self, v):
    self.width = max(1.0, min(40.0, v))
    self.apply_style()
    self.flash("épaisseur %d" % self.width)

def apply_style(self):
    o = self.selected
    if o is not None and o in self.objects:
        o["stroke"] = self.color
        o["width"] = self.width
        self.queue_draw()

def flash(self, msg):
    self.status = msg
    self.status_until = time.time() + 1.5
    self.queue_draw()

# ── historique ──────────────────────────────────────────────────────
def push_undo(self):
    self.undo_stack.append(copy.deepcopy(self.objects))
    if len(self.undo_stack) > 120:
        self.undo_stack.pop(0)
    self.redo_stack.clear()

def undo(self):
    if self.undo_stack:
        self.redo_stack.append(copy.deepcopy(self.objects))
        self.objects = self.undo_stack.pop()
        self.selected = None

def redo(self):
    if self.redo_stack:
        self.undo_stack.append(copy.deepcopy(self.objects))
        self.objects = self.redo_stack.pop()
        self.selected = None

def delete_selected(self):
    if self.selected in self.objects:
        self.push_undo()
        self.objects.remove(self.selected)
        self.selected = None
        self.queue_draw()

def clear_all(self):
    if self.objects:
        self.push_undo()
        self.objects = []
        self.selected = None
        self.flash("tableau effacé")

# ── clic-à-travers : seule la barre reste cliquable ─────────────────
def set_passthrough(self, on):
    self.passthrough = on
    gw = self.get_window()
    if gw is not None:
        gw.input_shape_combine_region(self.toolbar_region() if on else self.full_region(), 0, 0)
    self.flash("clic-à-travers " + ("ON" if on else "OFF"))

# ── entrées souris ──────────────────────────────────────────────────
def on_press(self, w, e):
    x, y = e.x, e.y
    if self.dormant:
        cx, cy = self.dot_center()
        if e.button == 1 and dist(x, y, cx, cy) <= DOT_R + 6:
            self._dot_pressed = True
            if e.type == Gdk.EventType._2BUTTON_PRESS:
```

```
                GLib.timeout_add(HOLD_MS, self._dot_wake_if_held)
            return True
        bid = self.button_at(x, y)
        if bid is not None and e.button == 1:
            self.do_button(bid)
            return True
        if self.passthrough:
            return False
        if e.button == 3:
            self.finish_pen()
            return True
        if e.button != 1:
            return False

        tool = self.tool
        if tool == "select":
            o = self.topmost_at(x, y)
            self.selected = o
            if o is not None:
                self.push_undo()
                self.drag = ("move",)
                self.start = (x, y)
            self.queue_draw()
            return True
        if tool == "eraser":
            o = self.topmost_at(x, y)
            if o is not None:
                self.push_undo()
                self.objects.remove(o)
                if self.selected is o:
                    self.selected = None
            self.drag = ("erase",)
            self.queue_draw()
            return True
        if tool == "pencil":                        # le feutre
            self.push_undo()
            self.temp = self.new_obj("path", [(x, y)])
            self.drag = ("free",)
            return True
        if tool in ("rect", "ellipse", "line"):
            self.push_undo()
            self.temp = self.new_obj(tool, [(x, y), (x, y)])
            self.drag = ("create",)
            self.start = (x, y)
            return True
        return False

    def on_motion(self, w, e):
        if self.dormant:
            return False
        if self.drag is None:
            return False
        x, y = e.x, e.y
        k = self.drag[0]
        if k == "move" and self.selected:
            obj_translate(self.selected, x - self.start[0], y - self.start[1])
            self.start = (x, y)
        elif k == "free" and self.temp:
            self.temp["pts"].append((x, y))
        elif k == "create" and self.temp:
            self.temp["pts"][1] = (x, y)
        elif k == "erase":
            o = self.topmost_at(x, y)
            if o is not None:
                self.objects.remove(o)
        self.queue_draw()
```

```
        return True

    def on_release(self, w, e):
        if self.dormant:
            self._dot_pressed = False
            return True
        if self.drag and self.drag[0] in ("free", "create") and self.temp:
            keep = True
            if self.temp["type"] == "line":
                p = self.temp["pts"]
                keep = dist(p[0][0], p[0][1], p[1][0], p[1][1]) >= 3
            if self.temp["type"] == "path" and len(self.temp["pts"]) < 2:
                keep = False
            if keep:
                self.objects.append(self.temp)
                self.selected = self.temp
        self.temp = None
        self.drag = None
        self.queue_draw()
        return True

    def finish_pen(self):
        if self.pen_pts and len(self.pen_pts) >= 2:
            self.objects.append(self.new_obj("path", self.pen_pts[:]))
        self.pen_pts = None
        self.queue_draw()

    def on_scroll(self, w, e):
        if e.direction == Gdk.ScrollDirection.UP:
            self.set_width(self.width + 1)
        elif e.direction == Gdk.ScrollDirection.DOWN:
            self.set_width(self.width - 1)
        return True

    def on_key(self, w, e):
        k = e.keyval
        if k in (Gdk.KEY_Escape, Gdk.KEY_q):
            if self.pen_pts is not None:
                self.finish_pen()
            else:
                self.destroy()
        elif k in (Gdk.KEY_Delete, Gdk.KEY_BackSpace):
            self.delete_selected()
        elif k == Gdk.KEY_z:
            self.undo(); self.queue_draw()
        elif k == Gdk.KEY_y:
            self.redo(); self.queue_draw()
        elif k == Gdk.KEY_space:
            self.set_passthrough(not self.passthrough)
        else:
            m = {Gdk.KEY_b: "pencil", Gdk.KEY_l: "line", Gdk.KEY_r: "rect",
                 Gdk.KEY_e: "ellipse", Gdk.KEY_x: "eraser", Gdk.KEY_s: "select"}
            if k in m:
                self.tool = m[k]; self.finish_pen(); self.queue_draw()
        return True

    # ── rendu des objets ──────────────────────────────────────────
    def draw_obj(self, cr, o, ghost=False):
        pts = o["pts"]; r, g, b = o["stroke"]; a = 0.45 if ghost else 0.97
        cr.set_line_cap(cairo.LINE_CAP_ROUND); cr.set_line_join(cairo.LINE_JOIN_ROUND)

        def path():
            typ = o["type"]
            if typ in ("line", "path"):
                cr.move_to(*pts[0])
                for p in pts[1:]:
```

```
                cr.line_to(*p)
            elif typ == "rect":
                x0, y0, x1, y1 = obj_bbox(o); cr.rectangle(x0, y0, x1 - x0, y1 - y0)
            elif typ == "ellipse":
                x0, y0, x1, y1 = obj_bbox(o)
                cx, cy = (x0 + x1) / 2, (y0 + y1) / 2
                rx, ry = max((x1 - x0) / 2, .5), max((y1 - y0) / 2, .5)
                cr.save(); cr.translate(cx, cy); cr.scale(rx, ry)
                cr.arc(0, 0, 1, 0, 2 * math.pi); cr.restore()

        if o.get("fill") and o["type"] in ("rect", "ellipse", "path"):
            path(); fr, fg, fb = o["fill"]
            cr.set_source_rgba(fr, fg, fb, 0.3 if ghost else 0.5); cr.fill()
        path(); cr.set_source_rgba(0, 0, 0, 0.5)
        cr.set_line_width(o["width"] + 2.5); cr.stroke()
        path(); cr.set_source_rgba(r, g, b, a)
        cr.set_line_width(o["width"]); cr.stroke()

    def on_draw(self, w, cr):
        a = self.get_allocation()

        # fond
        cr.set_operator(cairo.OPERATOR_SOURCE)
        if self.dormant:
            cr.set_source_rgba(0, 0, 0, 0.0)      # rien que la pastille
        else:
            cr.set_source_rgba(0, 0, 0, 0.0 if self.has_alpha else 0.08)
        cr.paint()
        cr.set_operator(cairo.OPERATOR_OVER)

        if self.dormant:
            self.draw_dot(cr)
            return False

        self.layout_buttons(a.width, a.height)
        for o in self.objects:
            self.draw_obj(cr, o)
        if self.temp:
            self.draw_obj(cr, self.temp, ghost=True)

        # sélection
        sel = self.selected
        if sel is not None and sel in self.objects and self.tool == "select":
            x0, y0, x1, y1 = obj_bbox(sel)
            cr.set_dash([5, 4]); cr.set_line_width(1); cr.set_source_rgba(1, 1, 1, .8)
            cr.rectangle(x0 - 4, y0 - 4, x1 - x0 + 8, y1 - y0 + 8); cr.stroke(); cr.set_dash([])

        self.draw_toolbar(cr)

        if self.status and time.time() < self.status_until:
            self._text(cr, self.status, 18, 30, (1, 1, 1))
        return False

    # ── la pastille de rappel (état dormant) ──────────────────────────
    def draw_dot(self, cr):
        cx, cy = self.dot_center()
        cr.arc(cx, cy, DOT_R, 0, 2 * math.pi)          # halo sombre
        cr.set_source_rgba(0, 0, 0, 0.5); cr.fill()
        cr.arc(cx, cy, DOT_R - 3, 0, 2 * math.pi)      # pastille couleur
        cr.set_source_rgba(*self.color, 0.95); cr.fill()
        cr.arc(cx, cy, DOT_R - 3, 0, 2 * math.pi)
        cr.set_source_rgba(1, 1, 1, 0.85); cr.set_line_width(1.5); cr.stroke()
        # petit trait de feutre en signe
        cr.set_source_rgba(1, 1, 1, 0.95)
        cr.set_line_width(2.4); cr.set_line_cap(cairo.LINE_CAP_ROUND)
        cr.move_to(cx - 10, cy + 5)
```

```
    cr.curve_to(cx - 3, cy - 10, cx + 3, cy + 10, cx + 10, cy - 5)
    cr.stroke()

# ── barre d'outils dessinée dans le canevas ──────────────────────────
def draw_toolbar(self, cr):
    for bid, x, y, w, h in self.buttons:
        active = (bid == self.tool)
        # fond du bouton
        self._round_rect(cr, x, y, w, h, 9)
        if bid == "clear":
            cr.set_source_rgba(0.80, 0.12, 0.16, 0.92)
        elif active:
            cr.set_source_rgba(0.16, 0.55, 0.95, 0.95)
        else:
            cr.set_source_rgba(0.14, 0.15, 0.18, 0.85)
        cr.fill()
        self._round_rect(cr, x, y, w, h, 9)
        cr.set_source_rgba(1, 1, 1, 0.22); cr.set_line_width(1); cr.stroke()
        self.draw_icon(cr, bid, x, y, w, h)

def draw_icon(self, cr, bid, x, y, w, h):
    cx, cy = x + w / 2, y + h / 2
    white = (0.95, 0.95, 0.97)
    cr.set_line_width(2.2); cr.set_line_cap(cairo.LINE_CAP_ROUND)
    cr.set_source_rgba(*white, 1)

    if bid == "clear":
        self._glyph(cr, "C", cx, cy, 20)
    elif bid == "eraser":
        cr.save(); cr.translate(cx, cy); cr.rotate(-0.5)
        cr.rectangle(-11, -6, 22, 12); cr.stroke()
        cr.move_to(0, -6); cr.line_to(0, 6); cr.stroke(); cr.restore()
    elif bid == "color":
        cr.arc(cx, cy, 12, 0, 2 * math.pi)
        cr.set_source_rgba(*self.color, 1); cr.fill()
        cr.arc(cx, cy, 12, 0, 2 * math.pi)
        cr.set_source_rgba(1, 1, 1, .8); cr.set_line_width(1.5); cr.stroke()
    elif bid == "width_dn":
        cr.move_to(cx - 9, cy); cr.line_to(cx + 9, cy); cr.stroke()
    elif bid == "width_up":
        cr.move_to(cx - 9, cy); cr.line_to(cx + 9, cy)
        cr.move_to(cx, cy - 9); cr.line_to(cx, cy + 9); cr.stroke()
    elif bid == "ellipse":
        cr.save(); cr.translate(cx, cy); cr.scale(1.4, 1.0)
        cr.arc(0, 0, 9, 0, 2 * math.pi); cr.restore(); cr.stroke()
    elif bid == "rect":
        cr.rectangle(cx - 11, cy - 8, 22, 16); cr.stroke()
    elif bid == "line":
        cr.move_to(cx - 11, cy + 9); cr.line_to(cx + 11, cy - 9); cr.stroke()
    elif bid == "pencil":                       # feutre : petite vague
        cr.move_to(cx - 12, cy + 6)
        cr.curve_to(cx - 4, cy - 12, cx + 4, cy + 12, cx + 12, cy - 6)
        cr.stroke()
    elif bid == "select":
        cr.move_to(cx - 8, cy - 10); cr.line_to(cx - 8, cy + 8)
        cr.line_to(cx - 3, cy + 3); cr.line_to(cx + 1, cy + 11)
        cr.line_to(cx + 4, cy + 9); cr.line_to(cx, cy + 1)
        cr.line_to(cx + 7, cy + 1); cr.close_path(); cr.stroke()
    elif bid == "undo":
        cr.arc(cx, cy + 1, 9, math.radians(40), math.radians(300)); cr.stroke()
        cr.move_to(cx - 9, cy - 4); cr.line_to(cx - 12, cy + 2)
        cr.line_to(cx - 4, cy + 1); cr.close_path(); cr.fill()
    elif bid == "pass":
        cr.move_to(cx - 11, cy); cr.line_to(cx + 8, cy)
        cr.move_to(cx + 2, cy - 6); cr.line_to(cx + 10, cy); cr.line_to(cx + 2, cy + 6)
        cr.stroke()
```

```
            if self.passthrough:
                cr.set_source_rgba(0.2, 0.9, 0.35, 1)
                cr.arc(x + w - 9, y + 9, 4, 0, 2 * math.pi); cr.fill()
        elif bid == "quit":
            cr.move_to(cx - 8, cy - 8); cr.line_to(cx + 8, cy + 8)
            cr.move_to(cx + 8, cy - 8); cr.line_to(cx - 8, cy + 8); cr.stroke()

        # valeur d'épaisseur affichée sur les boutons − / +
        if bid == "width_up":
            self._text(cr, "%d" % self.width, x + 4, y - 4, (0.8, 0.85, 0.9), 12)

    def _round_rect(self, cr, x, y, w, h, r):
        cr.new_sub_path()
        cr.arc(x + w - r, y + r, r, -math.pi / 2, 0)
        cr.arc(x + w - r, y + h - r, r, 0, math.pi / 2)
        cr.arc(x + r, y + h - r, r, math.pi / 2, math.pi)
        cr.arc(x + r, y + r, r, math.pi, 1.5 * math.pi)
        cr.close_path()

    def _glyph(self, cr, s, cx, cy, size):
        cr.select_font_face("sans-serif", cairo.FONT_SLANT_NORMAL, cairo.FONT_WEIGHT_BOLD)
        cr.set_font_size(size)
        xb, yb, tw, th, _, _ = cr.text_extents(s)
        cr.move_to(cx - tw / 2 - xb, cy - yb - th / 2 + th)
        cr.show_text(s)

    def _text(self, cr, s, x, y, rgb, size=15):
        cr.select_font_face("monospace", cairo.FONT_SLANT_NORMAL, cairo.FONT_WEIGHT_BOLD)
        cr.set_font_size(size)
        cr.move_to(x + 1, y + 1); cr.set_source_rgba(0, 0, 0, .8); cr.show_text(s)
        cr.move_to(x, y); cr.set_source_rgba(*rgb, 1); cr.show_text(s)

    # ── export (bonus, touche non mappée dans la barre pour rester compact) ──
    def export(self):
        ts = time.strftime("%Y%m%d-%H%M%S")
        base = os.path.expanduser("~/overlay-draw-%s" % ts)
        a = self.get_allocation()
        surf = cairo.ImageSurface(cairo.FORMAT_ARGB32, a.width, a.height)
        cr = cairo.Context(surf)
        for o in self.objects:
            self.draw_obj(cr, o)
        surf.write_to_png(base + ".png")
        self.flash("exporté %s.png" % base)


def main():
    Overlay()
    Gtk.main()


if __name__ == "__main__":
    main()
```

3.442 /opt/GSM/osmo-operator/tools/paysage-render.py

23376 octets, 454 lignes → 451 lignes (1 groupes compactés)

```
#!/usr/bin/env python3
# paysage-render.py - UNE scene, DEUX images : le fond d ecran du banc et
# l illustration de l encart central, decoupee dedans.
#
#   python3 tools/paysage-render.py [--scale 2] \
#       --fond configs/gsm-lab-fond.jpg --encart configs/gsm-lab-wallpaper.jpg
#
```

```
# [2026-10-11] POURQUOI. Le bureau montrait deux images qui ne se connaissaient
# pas : un paysage polygonal (le fond GNOME) et, au milieu, l illustration
# vectorielle « pylones au soleil couchant » (bleu-vert, jaune), ajustee dans
# l encart avec deux bandes floues au-dessus et au-dessous. L encart faisait
# trou. Ici les deux viennent du MEME dessin :
#
#   - le FOND est la scene entiere, en 16:9 (3840x2160 a --scale 2) : un
#     paysage low-poly a la maniere du fond Ubuntu « Province of the south of
#     France » (triangulation de Delaunay sur une peinture lisse, facettes
#     plates), avec le sujet du banc : trois pylones, le soleil, les collines ;
#   - l ENCART est la decoupe exacte de la boite (510,220 .. 1410,1010) du cadre
#     1920x1080 dans cette scene. osmo-launcher.py / osmo-panel.py posent la
#     fenetre de l encart a cette boite, projetee sur l ecran avec la MEME
#     transformation que GNOME applique au fond en mode « zoom » : l image de
#     l encart tombe donc pixel pour pixel sur le fond qu elle recouvre, quel
#     que soit l ecran. osmo-fft-snap.py l ajuste « entiere » dans W x H : comme
#     elle a deja ce format, plus de bandes.
#
# La composition est pensee pour l ecran du banc : le grand arbre a gauche,
# derriere le dock et les icones ; le village et les cypres a droite, derriere
# le conky (translucide) ; les pylones et le soleil DANS la boite de l encart ;
# le pied de l image (champs) sous la barre du lanceur.
#
# Ne depend que de Pillow, numpy et scipy (Delaunay), presents sur l image ISO
# comme sur l hote. Rendu deterministe (graine fixe).

import argparse
import math
import random

import numpy as np
from PIL import Image, ImageDraw, ImageFilter
from scipy.spatial import Delaunay

FW, FH = 1920, 1080               # le cadre de reference (tools/osmo-launcher.py)
BOX = (510, 220, 1410, 1010)      # l encart (osmo-fft-snap.py : BOX)

# ── LA PALETTE (celle du fond « Province », mesuree) ──────────────────────
P = {
    "ciel_haut": (62, 140, 155), "ciel_bas": (138, 184, 186),
    "horizon": (226, 214, 176), "soleil": (252, 238, 160), "halo": (238, 200, 96),
    "nuage": (234, 228, 210), "nuage_clair": (244, 240, 228), "nuage_ombre": (196, 196, 184),
    "nuage_bleu": (166, 188, 190),
    "lointain": (112, 152, 156), "lointain2": (92, 134, 138),
    "colline": (122, 142, 76), "colline2": (150, 164, 88), "colline3": (96, 118, 66),
    "ocre": (214, 168, 70), "ble": (230, 196, 92), "orange": (216, 124, 52),
    "rouge": (176, 84, 44), "brun": (108, 48, 36), "olive": (142, 150, 70),
    "vert": (104, 130, 60), "vert_sombre": (62, 86, 52), "route": (232, 222, 190),
    "feuille": (74, 104, 60), "feuille2": (108, 134, 64), "feuille3": (44, 66, 44),
    "feuille_rouge": (110, 42, 30), "tronc": (70, 40, 30), "tronc2": (38, 24, 20),
    "cypres": (50, 70, 48), "cypres2": (32, 46, 34),
    "mur": (226, 200, 140), "mur_ombre": (190, 160, 100), "toit": (178, 96, 54),
    "pylone": (34, 42, 48), "pylone2": (58, 70, 78), "parabole": (236, 236, 228),
}


# [2026-10-11] MOINS DE COULEUR, PLUS BRUT. La palette ci-dessus est celle du
# fond « Province », franche ; sur le banc, derriere des panneaux sombres, elle
# criait. On la rabat vers son gris (SATURATION : 1.0 = telle quelle, 0 = gris)
# et on l assombrit un rien ; les facettes sont plus grandes et plus inegales.
SATURATION = 0.55
SOMBRE = 0.93


def rabattre(c):
    lum = 0.299 * c[0] + 0.587 * c[1] + 0.114 * c[2]
```

```
    return tuple(int(max(0, min(255, (lum + (v - lum) * SATURATION) * SOMBRE))) for v in c)


P = {k: rabattre(v) for k, v in P.items()}


def lerp(a, b, t):
    return tuple(int(round(a[i] + (b[i] - a[i]) * t)) for i in range(3))


def shade(c, k):
    return tuple(max(0, min(255, int(round(v * k)))) for v in c)


# ── LES LIGNES DU PAYSAGE (en coordonnees du cadre 1920x1080) ──────────────
def ligne(x, base, amp, freqs, phase=0.0):
    """Une silhouette douce : somme de sinus."""
    y = base
    for i, (a, f) in enumerate(zip(amp, freqs)):
        y += a * math.sin(f * x / FW * 2 * math.pi + phase + i * 1.7)
    return y


HORIZON = 640.0
LIGNES = {
    # nom : (fonction y(x), couleur)
    "lointain": (lambda x: ligne(x, 612, (46, 20, 7), (1.3, 3.1, 7.0), 0.4), P["lointain"]),
    "lointain2": (lambda x: ligne(x, 648, (30, 12, 5), (2.2, 4.3, 9.0), 3.6), P["lointain2"]),
    "colline": (lambda x: ligne(x, 690, (40, 16, 6), (0.9, 2.6, 6.0), 1.2), P["colline"]),
    "colline2": (lambda x: ligne(x, 735, (30, 14, 5), (1.1, 3.4, 8.0), 3.3), P["colline2"]),
    "colline3": (lambda x: ligne(x, 770, (24, 10, 4), (1.4, 2.9, 7.5), 0.9), P["colline3"]),
    "ocre": (lambda x: ligne(x, 820, (26, 10, 4), (0.8, 2.2, 5.0), 2.6), P["ocre"]),
    "ble": (lambda x: ligne(x, 870, (20, 8, 4), (1.2, 3.0, 6.5), 1.5), P["ble"]),
    "orange": (lambda x: ligne(x, 925, (24, 10, 4), (0.7, 2.4, 5.5), 4.0), P["orange"]),
    "rouge": (lambda x: ligne(x, 985, (18, 8, 4), (1.0, 2.8, 6.0), 0.2), P["rouge"]),
    "brun": (lambda x: ligne(x, 1040, (14, 6, 3), (1.3, 3.2, 7.0), 2.9), P["brun"]),
}
ORDRE = ["lointain", "lointain2", "colline", "colline2", "colline3",
         "ocre", "ble", "orange", "rouge", "brun"]


def bande_polygone(f, s, pas=24):
    """Le polygone (echelle s) sous la ligne f, jusqu au bas du cadre."""
    pts = [(x * s, f(x) * s) for x in range(-40, FW + 41, pas)]
    return pts + [((FW + 40) * s, (FH + 40) * s), (-40 * s, (FH + 40) * s)]


# ── LA PEINTURE LISSE (ce que la triangulation va facetter) ────────────────
def ciel(w, h, s):
    """Degrade du ciel, chaud vers l horizon, et le halo du soleil."""
    yy, xx = np.mgrid[0:h, 0:w].astype(np.float32)
    t = np.clip(yy / (HORIZON * s), 0, 1)[..., None]
    haut = np.array(P["ciel_haut"], np.float32)
    bas = np.array(P["ciel_bas"], np.float32)
    hor = np.array(P["horizon"], np.float32)
    img = haut + (bas - haut) * np.clip(t / 0.75, 0, 1)
    th = np.clip((t - 0.62) / 0.38, 0, 1) ** 1.6
    img = img + (hor - img) * th
    # le halo : un soleil bas, au centre de l encart
    sx, sy, r = SOLEIL
    d = np.sqrt(((xx - sx * s) ** 2) + ((yy - sy * s) ** 2) * 1.35) / (r * s)
    halo = np.exp(-(d / 3.0) ** 2)[..., None] * 0.6
    halo += np.exp(-(d / 1.35) ** 2)[..., None] * 0.45
    img = img + (np.array(P["halo"], np.float32) - img) * np.clip(halo, 0, 1)
    return img
```

```
SOLEIL = (935, 548, 82)           # centre x, centre y, rayon (cadre 1920x1080)


def nuages(rng, s):
    """Des nuages plats, en bancs etires, comme des coupures de papier."""
    out = []
    bancs = [(230, 140, 640, 70), (1520, 110, 560, 64), (860, 190, 460, 52),
             (1250, 290, 360, 40), (470, 320, 420, 44), (1720, 350, 300, 32),
             (1020, 400, 260, 26), (320, 450, 260, 22), (1480, 450, 220, 18)]
    for (cx, cy, lw, lh) in bancs:
        for k in range(2):
            pts = []
            n = rng.randint(7, 10)
            ox, oy = rng.uniform(-lw * 0.2, lw * 0.2), rng.uniform(-lh * 0.3, lh * 0.3) - k * lh * 0.18
            for i in range(n):
                a = 2 * math.pi * i / n
                rx = lw * (0.5 - 0.14 * k) * rng.uniform(0.8, 1.05)
                ry = lh * (0.5 - 0.1 * k) * rng.uniform(0.7, 1.15)
                pts.append(((cx + ox + rx * math.cos(a)) * s, (cy + oy + ry * math.sin(a)) * s))
            c = [P["nuage"], P["nuage_clair"]][k]
            out.append((pts, c))
        # une ombre bleutee sous le banc
        pts = [((cx + lw * 0.45 * math.cos(a) - lw * 0.08) * s, (cy + lh * 0.55 + lh * 0.22 * math.sin(a)) * s)
               for a in np.linspace(0, 2 * math.pi, 7, endpoint=False)]
        out.insert(len(out) - 2, (pts, P["nuage_bleu"]))
    return out


def peinture(w, h, s, rng):
    img = Image.fromarray(np.clip(ciel(w, h, s), 0, 255).astype(np.uint8))
    d = ImageDraw.Draw(img)
    nuag = nuages(rng, s)
    for pts, c in nuag:
        d.polygon(pts, fill=c)
    # le disque du soleil (facette ensuite, puis redessine net par-dessus)
    sx, sy, r = SOLEIL
    d.ellipse((sx * s - r * s, sy * s - r * s, sx * s + r * s, sy * s + r * s), fill=P["soleil"])
    for nom in ORDRE:
        f, c = LIGNES[nom]
        d.polygon(bande_polygone(f, s), fill=c)
    # une route claire qui serpente du bas vers le village, a droite
    route = [(1130, 1100), (1200, 1010), (1340, 930), (1500, 860), (1620, 790), (1700, 740), (1740, 712)]
    larg = [40, 32, 26, 18, 12, 8, 5]
    gauche, droite = [], []
    for i, (x, y) in enumerate(route):
        xp, yp = route[max(0, i - 1)]
        xn, yn = route[min(len(route) - 1, i + 1)]
        dx, dy = xn - xp, yn - yp
        n = math.hypot(dx, dy) or 1.0
        nx, ny = -dy / n, dx / n          # la normale
        gauche.append(((x + nx * larg[i]) * s, (y + ny * larg[i]) * s))
        droite.append(((x - nx * larg[i]) * s, (y - ny * larg[i]) * s))
    d.polygon(gauche + droite[::-1], fill=P["route"])
    # taches de vert et d olive dans les champs (bosquets lointains, haies)
    for (cx, cy, rw, rh, c) in [(260, 790, 90, 18, P["vert"]), (1180, 760, 120, 14, P["vert_sombre"]),
                                (640, 805, 70, 12, P["olive"]), (1800, 700, 90, 16, P["vert"]),
                                (120, 905, 160, 30, P["olive"]), (480, 940, 120, 22, P["vert_sombre"]),
                                (1950, 870, 140, 28, P["olive"]), (80, 1010, 130, 26, P["vert_sombre"])]:
        pts = [((cx + rw * math.cos(a) * rng.uniform(0.8, 1.1)) * s,
                (cy + rh * math.sin(a) * rng.uniform(0.7, 1.2)) * s)
               for a in np.linspace(0, 2 * math.pi, 8, endpoint=False)]
        d.polygon(pts, fill=c)
    return img, nuag
```

```
# ── LES POINTS, PUIS LES FACETTES ──────────────────────────────────────
def points_scene(w, h, s, rng, nuag):
    pts = []
    # un semis irregulier : large dans le ciel, serre dans les champs
    y = 0.0
    while y < FH + 60:
        pas = 210 if y < 520 else (96 if y < 800 else 78)
        x = -40.0
        while x < FW + 60:
            pts.append(((x + rng.uniform(-0.6, 0.6) * pas) * s, (y + rng.uniform(-0.6, 0.6) * pas) * s))
            x += pas
        y += pas * 0.85
    # les silhouettes : des points le long de chaque ligne, pour que les
    # facettes s y arretent net (la crete reste une crete)
    for nom in ORDRE:
        f, _ = LIGNES[nom]
        x = -40.0
        while x < FW + 40:
            pts.append((x * s, f(x) * s))
            x += rng.uniform(34, 64)
    # le contour des nuages et du soleil
    for poly, _ in nuag:
        pts.extend(poly)
    sx, sy, r = SOLEIL
    for a in np.linspace(0, 2 * math.pi, 14, endpoint=False):
        pts.append(((sx + r * math.cos(a)) * s, (sy + r * math.sin(a)) * s))
    for a in np.linspace(0, 2 * math.pi, 9, endpoint=False):
        pts.append(((sx + 2.2 * r * math.cos(a)) * s, (sy + 1.6 * r * math.sin(a)) * s))
    # les coins
    pts += [(-60 * s, -60 * s), ((FW + 60) * s, -60 * s), (-60 * s, (FH + 60) * s), ((FW + 60) * s, (FH + 60) * s)]
    return np.array(pts, np.float64)


def facettes(peint, pts, rng):
    """Chaque triangle de Delaunay prend la couleur moyenne de la peinture
    sous lui, un peu nuancee : les facettes plates du low-poly."""
    arr = np.asarray(peint).astype(np.float32)
    h, w = arr.shape[:2]
    tri = Delaunay(pts)
    out = Image.new("RGB", (w, h), P["ciel_haut"])
    d = ImageDraw.Draw(out)
    # points d echantillonnage barycentriques (interieur du triangle)
    bary = np.array([(1 / 3, 1 / 3, 1 / 3), (0.6, 0.2, 0.2), (0.2, 0.6, 0.2), (0.2, 0.2, 0.6),
                     (0.45, 0.45, 0.1), (0.1, 0.45, 0.45), (0.45, 0.1, 0.45)], np.float32)
    for simplex in tri.simplices:
        p = pts[simplex]
        samp = bary @ p
        xs = np.clip(samp[:, 0], 0, w - 1).astype(int)
        ys = np.clip(samp[:, 1], 0, h - 1).astype(int)
        cols = arr[ys, xs]
        # la mediane plutot que la moyenne : un triangle a cheval sur une crete
        # prend la couleur du cote qui le remplit, pas un melange gris
        c = np.median(cols, axis=0)
        k = rng.uniform(0.90, 1.10)
        c = np.clip(c * k, 0, 255).astype(int)
        d.polygon([tuple(q) for q in p], fill=tuple(int(v) for v in c))
    return out


# ── LES SUJETS, NETS, PAR-DESSUS ───────────────────────────────────────
def pylone(d, s, x, base, haut, larg, deck=False, paraboles=(), antennes=True):
    """Un pylone en treillis : deux montants qui se resserrent, des traverses
    en croix, une plateforme et des paraboles pour le grand."""
    c, c2 = P["pylone"], P["pylone2"]
    ht = base - haut
    n = max(6, int(ht / 42))
```

```
    pts_g, pts_d = [], []
    for i in range(n + 1):
        t = i / n
        y = base - ht * t
        lw = larg * (1 - 0.82 * t) + 2
        pts_g.append((x - lw, y))
        pts_d.append((x + lw, y))
    lw_px = max(2, int(2.2 * s))
    # montants
    d.line([(px * s, py * s) for px, py in pts_g], fill=c, width=lw_px)
    d.line([(px * s, py * s) for px, py in pts_d], fill=c, width=lw_px)
    # traverses en X et paliers
    for i in range(n):
        (gx0, gy0), (gx1, gy1) = pts_g[i], pts_g[i + 1]
        (dx0, dy0), (dx1, dy1) = pts_d[i], pts_d[i + 1]
        d.line([(gx0 * s, gy0 * s), (dx1 * s, dy1 * s)], fill=c2, width=max(1, int(1.1 * s)))
        d.line([(dx0 * s, dy0 * s), (gx1 * s, gy1 * s)], fill=c2, width=max(1, int(1.1 * s)))
        d.line([(gx0 * s, gy0 * s), (dx0 * s, dy0 * s)], fill=c, width=max(1, int(1.3 * s)))
    # le pied : un socle polygonal
    d.polygon([((x - larg - 6) * s, base * s), ((x + larg + 6) * s, base * s),
               ((x + larg + 2) * s, (base - 10) * s), ((x - larg - 2) * s, (base - 10) * s)], fill=c)
    if deck:
        y = haut + ht * 0.30
        d.polygon([((x - larg * 1.5) * s, y * s), ((x + larg * 1.5) * s, y * s),
                   ((x + larg * 1.15) * s, (y - 22) * s), ((x - larg * 1.15) * s, (y - 22) * s)], fill=c)
        d.polygon([((x - larg * 1.3) * s, (y - 22) * s), ((x + larg * 1.3) * s, (y - 22) * s),
                   ((x + larg * 0.8) * s, (y - 36) * s), ((x - larg * 0.8) * s, (y - 36) * s)], fill=c2)
        d.polygon([((x - larg * 1.5) * s, y * s), ((x + larg * 1.5) * s, y * s),
                   ((x + larg * 0.9) * s, (y + 24) * s), ((x - larg * 0.9) * s, (y + 24) * s)], fill=c2)
    for (dy, cote, r) in paraboles:
        y = haut + ht * dy
        lw = larg * (1 - 0.82 * (base - y) / ht) + 2
        cx = x + cote * (lw + r * 0.7)
        d.ellipse(((cx - r * 0.55) * s, (y - r) * s, (cx + r * 0.55) * s, (y + r) * s), fill=P["parabole"])
        d.ellipse(((cx - r * 0.25) * s, (y - r * 0.85) * s, (cx + r * 0.35) * s, (y + r * 0.85) * s), fill=(214,
↪ 214, 206))
    if antennes:
        # la fleche en tete
        d.line([(x * s, haut * s), (x * s, (haut - ht * 0.14) * s)], fill=c, width=max(1, int(1.6 * s)))
        d.polygon([((x - 3) * s, (haut - ht * 0.14) * s), ((x + 3) * s, (haut - ht * 0.14) * s),
                   (x * s, (haut - ht * 0.21) * s)], fill=c)
        for k in (0.04, 0.09):
            y = haut + ht * k
            d.line([((x - 9) * s, y * s), ((x + 9) * s, y * s)], fill=c, width=max(1, int(1.3 * s)))
            d.rectangle(((x - 11) * s, (y - 5) * s, (x - 8) * s, (y + 5) * s), fill=c)
            d.rectangle(((x + 8) * s, (y - 5) * s, (x + 11) * s, (y + 5) * s), fill=c)


def arbre(d, s, rng, x, base, hauteur, envergure):
    """Le grand arbre de gauche : tronc tordu, branches, feuillage en eclats."""
    tx = x
    # tronc en plusieurs segments qui s inclinent
    seg = [(tx, base), (tx + 10, base - hauteur * 0.25), (tx - 8, base - hauteur * 0.5),
           (tx + 14, base - hauteur * 0.7), (tx + 2, base - hauteur * 0.86)]
    larg = [34, 26, 20, 15, 10]
    for i in range(len(seg) - 1):
        (x0, y0), (x1, y1) = seg[i], seg[i + 1]
        l0, l1 = larg[i], larg[i + 1]
        d.polygon([((x0 - l0) * s, y0 * s), ((x0 + l0) * s, y0 * s), ((x1 + l1) * s, y1 * s), ((x1 - l1) * s, y1 *
↪ s)],
                  fill=P["tronc"])
        d.polygon([((x0 - l0) * s, y0 * s), ((x0 - l0 * 0.3) * s, y0 * s), ((x1 - l1 * 0.3) * s, y1 * s), ((x1 -
↪ l1) * s, y1 * s)],
                  fill=P["tronc2"])
    # les racines
```

```
    d.polygon([((tx - 60) * s, base * s), ((tx + 70) * s, base * s), ((tx + 30) * s, (base - 24) * s), ((tx - 28) *
↪   s, (base - 26) * s)],
              fill=P["tronc"])
    # branches
    branches = [((tx - 4, base - hauteur * 0.55), (tx - 120, base - hauteur * 0.78), 9),
                ((tx + 8, base - hauteur * 0.68), (tx + 150, base - hauteur * 0.9), 8),
                ((tx + 2, base - hauteur * 0.84), (tx - 70, base - hauteur * 1.02), 6),
                ((tx + 2, base - hauteur * 0.86), (tx + 60, base - hauteur * 1.05), 6),
                ((tx - 120, base - hauteur * 0.78), (tx - 210, base - hauteur * 0.88), 5),
                ((tx + 150, base - hauteur * 0.9), (tx + 260, base - hauteur * 0.98), 5)]
    for (a, b, w) in branches:
        d.line([(a[0] * s, a[1] * s), (b[0] * s, b[1] * s)], fill=P["tronc"], width=int(w * s))
    # feuillage : des eclats triangulaires autour de quelques centres
    centres = [(tx - 180, base - hauteur * 0.95, 150), (tx + 40, base - hauteur * 1.05, 190),
               (tx + 230, base - hauteur * 1.0, 160), (tx - 60, base - hauteur * 1.15, 130),
               (tx + 120, base - hauteur * 1.2, 120), (tx - 240, base - hauteur * 0.8, 90),
               (tx + 300, base - hauteur * 0.9, 100)]
    couleurs = [P["feuille"], P["feuille2"], P["feuille3"], P["feuille"], P["feuille2"], P["feuille"],
                P["feuille3"], P["feuille_rouge"]]
    for (cx, cy, r) in centres:
        for _ in range(26):
            a = rng.uniform(0, 2 * math.pi)
            rr = r * rng.uniform(0.2, 1.0)
            px, py = cx + rr * math.cos(a), cy + rr * math.sin(a) * 0.75
            n = rng.randint(3, 5)
            size = rng.uniform(28, 70)
            pts = []
            for i in range(n):
                aa = 2 * math.pi * i / n + rng.uniform(-0.3, 0.3)
                pts.append(((px + size * math.cos(aa)) * s, (py + size * 0.8 * math.sin(aa)) * s))
            c = rng.choice(couleurs)
            d.polygon(pts, fill=shade(c, rng.uniform(0.85, 1.12)))


def cypres(d, s, x, base, h, w):
    d.polygon([((x - w) * s, base * s), ((x + w) * s, base * s), ((x + w * 0.6) * s, (base - h * 0.55) * s),
               (x * s, (base - h) * s), ((x - w * 0.6) * s, (base - h * 0.55) * s)], fill=P["cypres"])
    d.polygon([((x - w) * s, base * s), ((x - w * 0.1) * s, base * s), ((x - w * 0.05) * s, (base - h * 0.6) * s),
               (x * s, (base - h) * s), ((x - w * 0.6) * s, (base - h * 0.55) * s)], fill=P["cypres2"])


def maison(d, s, x, base, w, h):
    d.polygon([(x * s, base * s), ((x + w) * s, base * s), ((x + w) * s, (base - h) * s), (x * s, (base - h) * s)],
↪   fill=P["mur"])
    d.polygon([((x + w * 0.62) * s, base * s), ((x + w) * s, base * s), ((x + w) * s, (base - h) * s), ((x + w *
↪   0.62) * s, (base - h) * s)],
              fill=P["mur_ombre"])
    d.polygon([((x - w * 0.08) * s, (base - h) * s), ((x + w * 1.08) * s, (base - h) * s),
               ((x + w * 0.95) * s, (base - h * 1.35) * s), ((x + w * 0.05) * s, (base - h * 1.35) * s)],
↪   fill=P["toit"])
    d.polygon([((x - w * 0.08) * s, (base - h) * s), ((x + w * 0.5) * s, (base - h) * s),
               ((x + w * 0.45) * s, (base - h * 1.35) * s), ((x + w * 0.05) * s, (base - h * 1.35) * s)],
↪   fill=shade(P["toit"], 1.15))
    # fenetres et porte, en ombre
    for fx in (0.14, 0.40):
        d.rectangle(((x + w * fx) * s, (base - h * 0.78) * s, (x + w * (fx + 0.1)) * s, (base - h * 0.55) * s),
↪   fill=P["cypres2"])
    d.rectangle(((x + w * 0.27) * s, (base - h * 0.38) * s, (x + w * 0.37) * s, base * s), fill=P["cypres2"])
  ×4
def dessiner(scale, seed=7, antennes=True):
    rng = random.Random(seed)
    w, h = FW * scale, FH * scale
    peint, nuag = peinture(w, h, scale, rng)
    pts = points_scene(w, h, scale, rng, nuag)
    img = facettes(peint, pts, rng)
```

```
    # le soleil net (un disque a facettes), puis les terres repassent devant
    avant = img.copy()
    s = scale
    d = ImageDraw.Draw(img)
    sx, sy, r = SOLEIL
    d.polygon([((sx + r * math.cos(a)) * s, (sy + r * math.sin(a)) * s)
               for a in np.linspace(0, 2 * math.pi, 11, endpoint=False)], fill=P["soleil"])
    masque = Image.new("L", img.size, 0)
    ImageDraw.Draw(masque).polygon(bande_polygone(LIGNES["lointain"][0], scale), fill=255)
    img.paste(avant, (0, 0), masque)
    # le village a droite (derriere le conky), sur la crete « colline »
    fc = LIGNES["colline"][0]
    maison(d, s, 1705, fc(1725) + 6, 84, 42)
    for (x, hh, ww) in [(1660, 120, 10), (1680, 96, 7), (1820, 104, 8)]:
        cypres(d, s, x, fc(x) + 10, hh, ww)
    # [2026-10-11] LES PYLONES NE SONT QUE DANS L ENCART. Le fond d ecran est le
    # paysage nu ; c est la fenetre du milieu qui y plante les antennes, dans sa
    # boite - les trois sont donc entierement dans BOX (510..1410, 220..1010).
    if antennes:
        fl2 = LIGNES["lointain2"][0]
        pylone(d, s, 592, fl2(592) + 4, 505, 7, antennes=True)
        pylone(d, s, 705, fl2(705) + 6, 420, 11, antennes=True)
        pylone(d, s, 1255, fc(1255) + 2, 330, 30, deck=True,
               paraboles=((0.42, -1, 18), (0.50, 1, 15), (0.60, -1, 12)))
    # le grand arbre a gauche, derriere le dock et les icones
    fo = LIGNES["ocre"][0]
    arbre(d, s, rng, 170, fo(170) + 40, 560, 300)
    # des touffes d herbe au pied, sous la barre du lanceur
    for _ in range(60):
        x = rng.uniform(-20, FW + 20)
        y = rng.uniform(1030, 1085)
        hh = rng.uniform(10, 26)
        c = shade(P["brun"] if y > 1040 else P["rouge"], rng.uniform(0.7, 0.95))
       d.polygon([((x - 5) * s, y * s), ((x + 5) * s, y * s), ((x + rng.uniform(-6, 6)) * s, (y - hh) * s)], fill=c)
    return img


def main():
    ap = argparse.ArgumentParser()
    ap.add_argument("--scale", type=int, default=2, help="2 : fond 3840x2160, encart 1800x1580")
    ap.add_argument("--fond", required=True, help="le fond d ecran GNOME (scene entiere)")
    ap.add_argument("--encart", required=True, help="l illustration de l encart (decoupe BOX)")
    ap.add_argument("--seed", type=int, default=7)
    a = ap.parse_args()

    # deux rendus de la meme graine : le paysage nu (fond) et le meme avec les
    # pylones, dont on ne garde que la boite de l encart
    img = dessiner(a.scale, a.seed, antennes=False)
    img.save(a.fond, "JPEG", quality=92, optimize=True)
    s = a.scale
    enc = dessiner(a.scale, a.seed, antennes=True).crop((BOX[0] * s, BOX[1] * s, BOX[2] * s, BOX[3] * s))
    enc.save(a.encart, "JPEG", quality=92, optimize=True)
    print(f"fond   : {a.fond} {img.size}")
    print(f"encart : {a.encart} {enc.size} (boite {BOX} x{s})")


if __name__ == "__main__":
    main()
```

3.443 /opt/GSM/osmo-operator/tools/plymouth-render.py

9007 octets, 191 lignes → 191 lignes

```
#!/usr/bin/env python3
# plymouth-render.py - fabrique les images du theme Plymouth « osmo-bts ».
#
#     python3 tools/plymouth-render.py --out configs/plymouth/osmo-bts
#
# LE THEME EST GENERE, PAS STOCKE. Un theme Plymouth, ce sont des PNG : les
# garder dans le depot, c est une trentaine de binaires qu on ne peut ni relire
# ni corriger en diff. Ici le dessin EST le code - on change une couleur, on
# rejoue le script, et l ISO suivante en herite. iso_modules/86-finitions.sh
# l appelle au build.
#
# CE QU ON VOIT AU DEMARRAGE ([2026-10-23] plus simple, 2D pure) :
#
#                        (  (  ( • )  )  )
#                              /|\
#                             / | \         une antenne stylisee au centre,
#                            /__|__\        des cercles fins qui s ouvrent
#                                           depuis son sommet
#                           LAB GSM
#                  ──────── (barre) ────────
#
#   Un fond uni, une seule antenne en traits epais arrondis, et des cercles
#   concentriques a trait net qui s ouvrent et s effacent. Plus de pylone en
#   treillis, de mobile ni d anneaux « 3D » (ellipses en perspective, halo
#   neon) : trop charge, ils se recouvraient.
#
# --gif FICHIER sort en plus un apercu anime (fond + 24 frames, reduit), pour
# le wiki ou pour juger le rendu sans redemarrer. Il ne va PAS dans --out :
# tout ce qui est dans le dossier du theme part dans l initrd.
#
# Ne depend que de Pillow et des polices DejaVu, comme wallpaper-render.py.
import argparse
import math
import os

from PIL import Image, ImageDraw, ImageFont

W, H = 1920, 1080
FONT_DIR = "/usr/share/fonts/truetype/dejavu"
BG_TOP, BG_BOT = (8, 12, 26), (16, 26, 52)
CYAN, GREEN, VIOLET, GREY = (88, 166, 255), (63, 185, 80), (166, 122, 255), (139, 148, 158)
NEON = (64, 224, 255)            # le cyan des anneaux : vif, comme un neon
FRAMES = 24


def font(name, size):
    try:
        return ImageFont.truetype(os.path.join(FONT_DIR, name), size)
    except OSError:
        return ImageFont.load_default()


def background():
    """Fond UNI bleu nuit ([2026-10-23] 2D pure : plus de degrade vertical)."""
    return Image.new("RGB", (W, H), BG_TOP)


def mast(d, cx, top_y, h, col=(200, 210, 225)):
    """L antenne, en 2D pure : un mat, deux jambages et deux traverses, traits
    epais a bouts ronds, et un point lumineux au sommet (d ou partent les ondes)."""
    w, lw = h * 0.26, 7
    foot = top_y + h
    def seg(a, b):
        d.line([a, b], fill=col, width=lw)
        for x, y in (a, b):
            d.ellipse([x - lw / 2, y - lw / 2, x + lw / 2, y + lw / 2], fill=col)
```

```
    seg((cx, top_y + 18), (cx, foot))
    seg((cx, top_y + 30), (cx - w, foot))
    seg((cx, top_y + 30), (cx + w, foot))
    for t in (0.55, 0.82):                      # traverses entre les jambages
        y = top_y + 30 + (foot - top_y - 30) * t
        dx = w * t
        seg((cx - dx, y), (cx + dx, y))
    r = 10
    d.ellipse([cx - r, top_y - r, cx + r, top_y + r], fill=NEON)
    return (cx, top_y)


def ripples(d, at, phase, rmax=300, n=4, col=NEON):
    """Les ondes, en 2D pure : des CERCLES concentriques centres sur l antenne,
    qui s ouvrent et s effacent en s eloignant. Un seul trait net par cercle :
    plus d ellipse en perspective, de cone, de halo ni de coeur clair (le
    « neon 3D » du 2026-10-01, retire le 2026-10-23)."""
    x, y = at
    for k in range(n):
        t = ((phase + k / n) % 1.0)
        r = 28 + t * rmax
        a = int(255 * (1 - t) ** 0.9)
        d.ellipse([x - r, y - r, x + r, y + r], outline=(col[0], col[1], col[2], a), width=4)


def main():
    ap = argparse.ArgumentParser()
    ap.add_argument("--out", required=True)
    ap.add_argument("--template", default=None,
                    help="le gabarit osmo-bts.script (defaut : celui du depot)")
    ap.add_argument("--gif", default=None,
                    help="ecrit aussi un apercu anime (GIF) a ce chemin")
    ap.add_argument("--gif-width", type=int, default=960,
                    help="largeur de l apercu GIF (defaut 960)")
    a = ap.parse_args()
    os.makedirs(a.out, exist_ok=True)

    bg = background()
    d = ImageDraw.Draw(bg)
    ant = mast(d, W // 2, int(H * 0.36), int(H * 0.25))
    f_title, f_sub = font("DejaVuSans-Bold.ttf", 64), font("DejaVuSansMono.ttf", 24)
    d.text((W / 2, 900), "LAB GSM", font=f_title, fill=(230, 237, 243), anchor="mm")
    d.text((W / 2, 952), "reseau 2G complet, sans materiel  ·  osmocom  ·  calypso emule",
           font=f_sub, fill=GREY, anchor="mm")
    bg.save(os.path.join(a.out, "background.png"), "PNG", optimize=True)

    # Les frames : seulement la BANDE entre les deux antennes, en RGBA. Plymouth
    # les pose par-dessus le fond - une image par frame plutot qu un fond
    # complet redessine 24 fois (24 x 1,3 Mo tiendrait mal dans un initrd).
    # La bande englobe les cercles ENTIERS (rayon maximal + trait) : un cercle
    # n est jamais coupe par son bord. Hors des traits elle est transparente,
    # donc legere dans l initrd.
    R, M = 230, 12
    bx0, bx1 = ant[0] - 28 - R - M, ant[0] + 28 + R + M
    strip_y = ant[1] - 28 - R - M
    strip_h = 2 * (28 + R + M)
    tip = (ant[0] - bx0, ant[1] - strip_y)                # le sommet de l antenne, dans la bande
    frames = []
    for i in range(FRAMES):
        t = i / FRAMES
        im = Image.new("RGBA", (int(bx1 - bx0), strip_h), (0, 0, 0, 0))
        dd = ImageDraw.Draw(im)
        # les ondes, et rien d autre
        ripples(dd, tip, t, rmax=R, n=3)
        im.save(os.path.join(a.out, "burst-%02d.png" % i), "PNG", optimize=True)
        frames.append(im)
```

```
    # L apercu anime : le fond, la bande posee dessus a sa place, une image par
    # frame, a ~12 i/s comme dans le theme (une frame toutes les 4 passes a
    # 50 Hz). Reduit : un GIF 1920x1080 de 24 images pese plusieurs Mo pour
    # rien.
    if a.gif:
        gw = max(160, a.gif_width)
        gh = int(H * gw / W)
        seq = []
        for im in frames:
            full = bg.convert("RGBA")
            full.alpha_composite(im, (int(bx0), int(strip_y)))
            seq.append(full.convert("RGB").resize((gw, gh), Image.LANCZOS)
                       .quantize(colors=128, method=Image.Quantize.MEDIANCUT))
        seq[0].save(a.gif, save_all=True, append_images=seq[1:], loop=0,
                    duration=int(1000 * 4 / 50), optimize=True)
        print("[plymouth] apercu : %s (%dx%d, %d images)" % (a.gif, gw, gh, FRAMES))

    # La barre d avancement : un fond et une pastille qui court dessus.
    bar = Image.new("RGBA", (520, 8), (255, 255, 255, 0))
    ImageDraw.Draw(bar).rounded_rectangle([0, 0, 519, 7], 4, fill=(38, 48, 76, 255))
    bar.save(os.path.join(a.out, "progress-box.png"), "PNG")
    dot = Image.new("RGBA", (64, 8), (0, 0, 0, 0))
    ImageDraw.Draw(dot).rounded_rectangle([0, 0, 63, 7], 4, fill=CYAN + (255,))
    dot.save(os.path.join(a.out, "progress-bar.png"), "PNG")
    # Le champ de saisie de la phrase LUKS (theme script : c est a nous de le
    # dessiner). Sans ces deux images, une installation chiffree demande sa
    # phrase sur un ecran vide.
    ent = Image.new("RGBA", (620, 56), (0, 0, 0, 0))
    ImageDraw.Draw(ent).rounded_rectangle([0, 0, 619, 55], 10, fill=(18, 26, 48, 235),
                                          outline=CYAN + (170,), width=2)
    ent.save(os.path.join(a.out, "entry.png"), "PNG")
    bul = Image.new("RGBA", (18, 18), (0, 0, 0, 0))
    ImageDraw.Draw(bul).ellipse([0, 0, 17, 17], fill=(230, 237, 243, 255))
    bul.save(os.path.join(a.out, "bullet.png"), "PNG")
    # ── LE SCRIPT SORT D ICI AUSSI ──────────────────────────────────────
    # Il a besoin de savoir OU est la bande entre les deux antennes ; ces
    # nombres viennent d etre calcules ci-dessus. Les recopier a la main dans
    # le .script, c est la garantie qu ils divergeront au premier reglage.
    tpl = os.path.join(os.path.dirname(os.path.abspath(__file__)),
                       "..", "configs", "plymouth", "osmo-bts", "osmo-bts.script")
    tpl = a.template or os.path.normpath(tpl)
    if os.path.isfile(tpl):
        with open(tpl, encoding="utf-8") as f:
            sc = f.read()
        for k, v in (("@@BX@@", bx0 / W), ("@@BY@@", strip_y / H),
                     ("@@BW@@", (bx1 - bx0) / W), ("@@BH@@", strip_h / H)):
            sc = sc.replace(k, "%.4f" % v)
        with open(os.path.join(a.out, "osmo-bts.script"), "w", encoding="utf-8") as f:
            f.write(sc)
        print("[plymouth] script : bande a x=%.3f y=%.3f  %.3f x %.3f (fractions d ecran)"
              % (bx0 / W, strip_y / H, (bx1 - bx0) / W, strip_h / H))
    else:
        print("[plymouth] gabarit %s introuvable - script NON regenere" % tpl)
    print("[plymouth] %s : fond + %d frames + barre + saisie" % (a.out, FRAMES))


if __name__ == "__main__":
    main()
```

3.444 /opt/GSM/osmo-operator/tools/qemu_bcch_grgsm.py

8133 octets, 171 lignes → 172 lignes (1 groupes compactés)

```
#!/usr/bin/env python3
"""
tools/qemu_bcch_grgsm.py — pont gr-gsm : I/Q DL du BTS (tee qemu) → bursts → demapper
+ decoder gr-gsm → GSMTAP → le shunt qemu (4730) → a_cd → le mobile campe.

ZÉRO hack : le VRAI signal du BTS, démodulé et décodé par la chaîne gr-gsm
officielle (gsm_bcch_ccch_demapper + control_channels_decoder, le même code
que grgsm_decode).

Chaîne :
  UDP I/Q (tee BSP qemu, port 6703)  ──parse TRXD (8 hdr + 148 cs16)──>
    ──GMSK demod différentiel (1 SPS)──>  148 bits durs
    ──gsmtap_hdr(UM_BURST) + 148 octets──>  post au demapper gr-gsm
    ──gsm_bcch_ccch_demapper (assemble 4 bursts du 51-MF)──>
    ──control_channels_decoder (deinterleave+Viterbi+FIRE)──>  GSMTAP L2
    ──network.socket_pdu(UDP_CLIENT 127.0.0.1:4730)──>  shunt feed_si → a_cd

Le démod produit les 148 bits du burst complet (3 tail | 57 | 1 | 26 train |
1 | 57 | 3 tail) ; le decoder gr-gsm extrait/corrige tout seul. Si décode jamais
(le decoder est silencieux sur CRC fail), inverser BIT_SIGN.

Usage : source /root/.env/bin/activate ; python tools/qemu_bcch_grgsm.py
Env   : IQ_TEE_PORT(6703) GSMTAP_HOST/PORT(127.0.0.1/4730) ARFCN(1)
        TS(0) BIT_SIGN(1)
"""
import os, socket, struct
import numpy as np
from gnuradio import gr, gsm, network

IQ_TEE_PORT = int(os.environ.get("IQ_TEE_PORT", "6703"))
GSMTAP_HOST = os.environ.get("GSMTAP_HOST", "127.0.0.1")
GSMTAP_PORT = int(os.environ.get("GSMTAP_PORT", "4730"))
ARFCN       = int(os.environ.get("ARFCN", "1"))
TS          = int(os.environ.get("TS", "0"))
BIT_SIGN    = int(os.environ.get("BIT_SIGN", "1"))

# --- gsmtap.h (gr-gsm) ---
GSMTAP_VERSION       = 0x02
GSMTAP_TYPE_UM_BURST = 0x03
GSMTAP_BURST_NORMAL  = 0x06
BURST_SIZE           = 148

# Les 8 training sequences GSM (26 bits) — pour valider le démod live.
ALL_TSC = [
    "⟨1⟩", "⟨2⟩",  ×4
    ⟨⟩ = (00100101110000100010010111,00101101110111100010110111)
        (01000011101110100100001110,01000111101101000100011110)
        (00011010111001000001101011,01001110101100000100111010)
        (10100111110110001010011111,11101111000100101110111100)
]
dbg_tsc = [0]   # compteur de bursts BCCH instrumentés (mutable)
# [2026-08-04] Manquait : le bloc de dump I/Q plus bas (udp_loop) faisait
# dbg_dump.append(...) sans que la liste existe -> NameError a la PREMIERE
# rafale BCCH (fn%51 dans 2..5), donc pont demarre puis mort aussitot.
dbg_dump = []   # bursts BCCH bruts en attente de vidage (mutable)

def demod_burst(iq):
    """iq : np.complex (>=148) -> string de 148 bits durs '0'/'1', ou None.

    Démod GMSK validé bout-en-bout sur capture réelle (26/26 TSC, FIRE CRC ok) :
      - dérotation -pi/2 par sample  (y = s * exp(-j pi/2 k))
      - slicer sur imag(y)>0
      - inversion de polarité (BIT_SIGN=1 -> invert ; -1 -> non)
      - décalage gauche d'1 bit : le burst osmo-trx arrive avec TSC@62,
        gr-gsm attend TSC@61 (3 tail|57|1|26 TSC@61..86|1|57|3 tail).
    """
```

```
    if len(iq) < BURST_SIZE:
        return None
    s = iq[:BURST_SIZE].astype(np.complex64)
    k = np.arange(BURST_SIZE)
    y = s * np.exp(-1j * np.pi / 2.0 * k)  # dérotation -pi/2
    bits = (y.imag > 0).astype(int)
    if BIT_SIGN > 0:
        bits = 1 - bits                    # polarité (capture réelle: inversée)
    bits = np.concatenate([bits[1:], [bits[-1]]])  # shift -1 -> TSC@61
    return "".join(str(b) for b in bits)

def decode_block(fns, datas):
    """Run one-shot : burst_source(4 bursts BCCH) -> demapper -> decoder ->
    socket_pdu(GSMTAP 4730). gr-gsm officiel, pas de _post. fns/datas = listes
    parallèles (frame numbers + strings de 148 bits)."""
    tb  = gr.top_block("bcch block")
    src = gsm.burst_source(fns, [TS] * len(fns), datas)
    src.set_arfcn(ARFCN)
    dm  = gsm.gsm_bcch_ccch_demapper(TS)
    dec = gsm.control_channels_decoder()
    snk = network.socket_pdu("UDP_CLIENT", GSMTAP_HOST, str(GSMTAP_PORT), 10000)
    tb.msg_connect((src, "out"), (dm, "bursts"))
    tb.msg_connect((dm, "bursts"), (dec, "bursts"))
    tb.msg_connect((dec, "msgs"), (snk, "pdus"))
    tb.start()
    tb.wait()                              # burst_source émet puis poste 'done'

def udp_loop():
    rx = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    rx.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    rx.bind(("0.0.0.0", IQ_TEE_PORT))
    print(f"[grgsm] I/Q in udp:{IQ_TEE_PORT} -> GSMTAP {GSMTAP_HOST}:{GSMTAP_PORT} "
          f"arfcn={ARFCN} ts={TS}", flush=True)
    # On NE filtre PAS par mf : on nourrit le demapper avec un 51-multiframe
    # COMPLET (toutes positions). Le demapper gr-gsm mappe lui-même BCCH/CCCH
    # selon fn%51 et extrait SI1/2/3/4 à la vraie position planifiée (TC).
    # (Le démod produit du bruit aux positions FCCH/SCH, que le demapper ignore.)
    # On décode par FENÊTRE de WINDOW_MF multiframes (pas 1 seul) : le demapper
    # gr-gsm a besoin d'un run CONTIGU de plusieurs 51-multiframes pour locker
    # son compteur TC et extraire les blocs BCCH. 1 multiframe isolé → 0 décode
    # (régression). 12 mf couvrent un cycle TC complet (0..7) → SI1/2/3/4 inclus.
    WINDOW_MF = int(os.environ.get("WINDOW_MF", "12"))
    cur_mf  = None                         # index multiframe courant (fn//51)
    fns, datas = [], []
    mf_count = 0
    npkt = nfed = ndec = 0
    while True:
        pkt, _ = rx.recvfrom(4096)
        npkt += 1
        if len(pkt) < 8 + BURST_SIZE * 4:
            if npkt <= 3:
                print(f"[grgsm] paquet court len={len(pkt)} (attendu>={8+BURST_SIZE*4})", flush=True)
            continue
        fn  = struct.unpack(">I", pkt[1:5])[0]
        raw = np.frombuffer(pkt[8:8 + BURST_SIZE * 4], dtype="<i2").astype(np.float32)
        bits = demod_burst(raw[0::2] + 1j * raw[1::2])
        if bits is None:
            continue
        nfed += 1
        if nfed <= 5 or nfed % 1000 == 0:
            print(f"[grgsm] burst #{nfed} fn={fn} mf={fn%51} pwr={np.mean(np.abs(raw)):.0f}", flush=True)
        # DEBUG : valide le démod sur les bursts BCCH live (TSC7 @61). Si le
        # match n'est pas ~26/26, le démod est faux sur le signal courant →
        # explique 0 GSMTAP. Compare aussi les 8 TSC pour repérer un décalage.
        if (fn % 51) in (2, 3, 4, 5) and dbg_tsc[0] < 24:
            dbg_tsc[0] += 1
```

```
            mid = bits[61:87]
            best = max((sum(a == b for a, b in zip(mid, t)), i)
                       for i, t in enumerate(ALL_TSC))
            print(f"[grgsm][TSC] fn={fn} mf={fn%51} best TSC{best[1]}={best[0]}/26 "
                  f"(TSC7={sum(a==b for a,b in zip(mid, ALL_TSC[7]))}/26) mid={mid}", flush=True)
            # DUMP I/Q brut des bursts BCCH pour brute-force offline du démod.
            dbg_dump.append((int(fn), raw.copy()))
            if len(dbg_dump) >= 8:
                import numpy as _np
                d = {}
                for i, (f, r) in enumerate(dbg_dump):
                    d[f"b{i}_iq"] = r.astype(_np.int16); d[f"b{i}_fn"] = f
                _np.savez("/opt/GSM/live_bursts.npz", **d)
                print(f"[grgsm][DUMP] 8 bursts BCCH live -> /opt/GSM/live_bursts.npz", flush=True)
                dbg_dump.clear()
        mfi = fn // 51
        if cur_mf is None:
            cur_mf = mfi
        if mfi != cur_mf:                  # nouvelle frontière de multiframe
            mf_count += 1
            cur_mf = mfi
            if mf_count >= WINDOW_MF and fns:   # assez de contexte -> décode le lot
                ndec += 1
                try:
                    decode_block(fns, datas)    # demapper -> GSMTAP (SI) si CRC ok
                    if ndec <= 5 or ndec % 20 == 0:
                        print(f"[grgsm] décode #{ndec} ({len(fns)} bursts / {mf_count} mf, "
                              f"fn {fns[0]}..{fns[-1]})", flush=True)
                except Exception as e:
                    print(f"[grgsm] decode_block err: {e}", flush=True)
                fns, datas = [], []
                mf_count = 0
        fns.append(fn)
        datas.append(bits)

def main():
    udp_loop()

if __name__ == "__main__":
    main()
```

3.445 /opt/GSM/osmo-operator/tools/screenfetch-art.py

6380 octets, 180 lignes → 180 lignes

```
#!/usr/bin/env python3
# screenfetch-art.py - ecrit configs/screenfetch/osmo-lab.sh : le logo en
# PIXEL ART COULEUR que screenfetch affiche a l ouverture de chaque terminal
# (osmo-banner, iso_modules/86-finitions.sh : « screenfetch -a »).
#
#     python3 tools/screenfetch-art.py
#
# [2026-10-01] Une parabole (la BTS) sur son socle vert, son bras et son
# illuminateur rouge, et le telephone a cote, ecran vert. Le dessin est CALCULE
# ici (ellipses, segments) sur une grille de pixels, puis rendu en demi-blocs
# « ▀ » : un caractere = deux pixels (couleur du haut en avant-plan, celle du
# bas en fond), en vraies couleurs 24 bits.
# screenfetch exige des lignes de meme largeur VISIBLE : chaque ligne fait W
# colonnes, les couleurs sont des sequences \e[...m (largeur nulle) que le
# printf de screenfetch interprete.
import math
import os

W, H = 30, 30
```

```
PAL = {
    "k": (16, 16, 24),     # contour
    "d": (84, 96, 130),    # parabole, creux
    "m": (138, 148, 180),  # parabole, flanc
    "l": (200, 208, 228),  # parabole, reflet
    "t": (216, 204, 160),  # bras de l illuminateur
    "r": (200, 70, 60),    # rouge : illuminateur et rotule
    "R": (232, 110, 100),
    "g": (140, 180, 140),  # socle
    "G": (90, 138, 90),
    "s": (150, 156, 176),  # support
    "p": (60, 66, 88),     # telephone, boitier
    "P": (96, 104, 134),
    "e": (70, 220, 120),   # telephone, ecran
    "E": (150, 255, 190),
    "y": (255, 176, 0),    # ondes
}
g = [["."] * W for _ in range(H)]


def put(x, y, c):
    x, y = int(round(x)), int(round(y))
    if 0 <= x < W and 0 <= y < H:
        g[y][x] = c


def line(x0, y0, x1, y1, c, w=1):
    n = int(max(abs(x1 - x0), abs(y1 - y0)) * 2) + 1
    for i in range(n + 1):
        t = i / n
        for dx in range(w):
            for dy in range(w):
                put(x0 + (x1 - x0) * t + dx, y0 + (y1 - y0) * t + dy, c)


def ell(cx, cy, a, b, ang):
    ca, sa = math.cos(ang), math.sin(ang)

    def f(x, y):
        dx, dy = x - cx, y - cy
        u, v = dx * ca + dy * sa, -dx * sa + dy * ca
        return (u / a) ** 2 + (v / b) ** 2
    return f


def paint_ell(f, c, only=None):
    for y in range(H):
        for x in range(W):
            if f(x, y) <= 1 and (only is None or g[y][x] in only):
                g[y][x] = c


# -- la parabole : coque claire, creux sombre decale vers le haut-gauche ---
ang = math.radians(40)
coque = ell(15, 9, 10.8, 5.6, ang)
creux = ell(13.8, 8.2, 8.8, 4.0, ang)
bord = ell(15, 9, 12.0, 6.8, ang)
paint_ell(bord, "k")
paint_ell(coque, "m")
for y in range(H):                      # reflet : le flanc haut de la coque
    for x in range(W):
        if coque(x, y) <= 1 and creux(x, y) > 1 and creux(x + 1.4, y + 1.4) <= 1:
            g[y][x] = "l"
paint_ell(creux, "d")
# -- bras de l illuminateur : du creux vers le haut a droite, bout rouge ---
line(14, 9, 22, 2.5, "k", 3)
```

```
line(14.5, 9.5, 21.6, 3.2, "t", 1)
for (x, y) in ((22, 1), (23, 1), (22, 2), (23, 2), (24, 2), (23, 3), (24, 3)):
    put(x, y, "k")
for (x, y) in ((22, 2), (23, 2), (23, 3)):
    put(x, y, "r")
put(23, 2, "R")
# -- support : du dessous de la parabole a la rotule, puis au socle ---------
line(11, 14, 8, 18, "k", 4)
line(11.5, 14.5, 9, 18, "s", 2)
for y in range(18, 26):                 # pied trapezoidal
    half = 2 + (y - 18) * 0.55
    for x in range(int(8 - half) - 1, int(8 + half) + 2):
        put(x, y, "k")
for y in range(18, 26):
    half = 2 + (y - 18) * 0.55
    for x in range(int(8 - half), int(8 + half) + 1):
        put(x, y, "s" if x < 8 else "m")
for (x, y) in ((7, 17), (8, 17), (9, 17), (6, 18), (10, 18), (6, 19), (10, 19), (7, 20), (8, 20), (9, 20)):
    put(x, y, "k")
for (x, y) in ((7, 18), (8, 18), (9, 18), (7, 19), (9, 19)):
    put(x, y, "r")
put(8, 19, "k")
# -- socle vert -------------------------------------------------------------
for x in range(1, 20):
    put(x, 25, "k"); put(x, 29, "k")
for y in range(26, 29):
    put(1, y, "k"); put(19, y, "k")
    for x in range(2, 19):
        put(x, y, "g" if y < 28 else "G")
for x in range(2, 19):
    put(x, 26, "l" if x % 7 else "g")
# -- le telephone, a droite, ecran vert, avec ses ondes ---------------------
for y in range(17, 28):
    for x in range(22, 29):
        put(x, y, "k")
for y in range(18, 27):
    for x in range(23, 28):
        put(x, y, "p")
for y in range(19, 22):
    for x in range(24, 27):
        put(x, y, "e")
put(24, 19, "E"); put(25, 19, "E")
for (x, y) in ((24, 23), (26, 23), (24, 24), (26, 24), (24, 25), (26, 25)):
    put(x, y, "P")
for (x, y) in ((25, 23), (25, 24), (25, 25)):
    put(x, y, "P")
line(27, 17, 28, 14, "k", 1)            # antenne
for (x, y) in ((24, 12), (26, 11), (25, 14), (28, 12)):
    put(x, y, "y")                      # ondes qui partent du telephone


def cell(top, bot):
    def fg(c): return "\\e[38;2;%d;%d;%dm" % PAL[c]
    def bg(c): return "\\e[48;2;%d;%d;%dm" % PAL[c]
    if top == "." and bot == ".":
        return " "
    if bot == ".":
        return fg(top) + "▀\\e[0m"
    if top == ".":
        return fg(bot) + "▄\\e[0m"
    return fg(top) + bg(bot) + "▀\\e[0m"


EN_TETE = r'''#!/bin/bash
# configs/screenfetch/osmo-lab.sh - le logo pixel art du banc, pour screenfetch -a.
# GENERE par tools/screenfetch-art.py : c est la qu on retouche le dessin.
```

```
# Source par screenfetch (asciiText) : startline, fulloutput (une ligne = un %s
# en queue, @W@ colonnes visibles chacune), labelcolor / textcolor.
#
# Les couleurs des INFOS ne se posent pas ici : screenfetch a deja fige ses
# lignes quand il source ce fichier. Elles sont posees par osmo-banner, qui
# filtre la sortie (utilisateur jaune, hote vert ; root rouge, hote jaune).
labelcolor=$'\e[1;33m'
textcolor=$'\e[0m'
startline="0"
logowidth="@W@"
fulloutput=(
'''


def main():
    out = []
    for r in range(0, H, 2):
        s = "".join(cell(g[r][x], g[r + 1][x]) for x in range(W))
        out.append('"' + s + '  %s"')
    dst = os.path.normpath(os.path.join(os.path.dirname(os.path.abspath(__file__)),
                                        "..", "configs", "screenfetch", "osmo-lab.sh"))
    os.makedirs(os.path.dirname(dst), exist_ok=True)
    with open(dst, "w") as f:
        f.write(EN_TETE.replace("@W@", str(W + 2)) + "\n".join(out) + ")\n")
    print("[screenfetch-art] %s : %d lignes de %d colonnes" % (dst, H // 2, W + 2))


if __name__ == "__main__":
    main()
```

3.446 /opt/GSM/osmo-operator/tools/si.py

2097 octets, 36 lignes → 36 lignes

```
#!/usr/bin/env python3
# Bridge : grgsm_decode -v (qui DECODE le vrai SI du cfile continu) -> parse les
# lignes SI (PD=0x06) -> GSMTAP 16B + L2 -> 4730 (shunt feed_si -> DATA_IND -> mobile).
import subprocess, socket, struct, re, sys
GSMTAP = struct.pack(">BBBBHbbIBBBB", 2,4,0x01,0,0,0,0,0,0x01,0,0,0)  # 16B, type UM, BCCH
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
CF = sys.argv[1] if len(sys.argv)>1 else "/tmp/iq_grgsm.fifo"
p = subprocess.Popen(["grgsm_decode","-m","BCCH","-t","0","-a","514","-c",CF,"-s","1083333","-v"],
                     stdout=subprocess.PIPE, stderr=subprocess.DEVNULL, text=True)
SI={0x1b:"SI3",0x1a:"SI2",0x1c:"SI4",0x21:"SI13",0x19:"SI1",0x1d:"SI2bis",0x1e:"SI2ter",0x06:"RR"}
n=0; nsch=0
for line in p.stdout:
    # SCH REEL : le receiver gr-gsm patche imprime "SCHBSIC <bsic> <fn> <toa>" sur
    # stdout (le port measurements est avale par le CLI grgsm_decode). On le
    # forwarde au shunt en UDP 4731 {magic 'SCH1', int32 bsic, int32 fn, int32 toa}
    # -> feed_sb -> shunt_dispatch_sb (BSIC reel; TOA reel si hors CALYPSO_CANNED).
    msch = re.search(r"SCHBSIC\s+(\d+)\s+(\d+)\s+(-?\d+)", line)
    if msch:
        bsic, fn, toa = int(msch.group(1)), int(msch.group(2)), int(msch.group(3))
        s.sendto(b"SCH1" + struct.pack("<iii", bsic, fn, toa), ("127.0.0.1", 4731))
        nsch += 1
        if nsch<=10 or nsch%100==0:
            print("[si-bridge] SCH -> shunt 4731 : BSIC=%d (ncc=%d bcc=%d) FN=%d TOA=%d  #%d"
                  %(bsic,(bsic>>3)&7,bsic&7,fn,toa,nsch),flush=True)
        continue
    m = re.search(r":\s+([0-9a-fA-F][0-9a-fA-F ]+)$", line.strip())
    if not m: continue
    try: by = bytes(int(x,16) for x in m.group(1).split())
    except: continue
    if len(by) >= 3 and by[1]==0x06 and by[2] in SI:   # RR PD + SI type
```

```
        L2 = (by[:23] + b"\x2b"*23)[:23]
        s.sendto(GSMTAP + L2, ("127.0.0.1",4730))
        n+=1
        if n<=20 or n%50==0:
            print("[si-bridge] %s (mt=0x%02x) -> feed_si (4730)  #%d"%(SI[by[2]],by[2],n),flush=True)
print("[si-bridge] fini, %d SI transmis"%n,flush=True)
```

3.447 /opt/GSM/osmo-operator/tools/status.sh

identique à /opt/GSM/osmo-operator/scripts/status.sh

3.448 /opt/GSM/osmo-operator/tools/vbox-wan-lab.sh

34463 octets, 654 lignes → 654 lignes

```
#!/bin/bash
# ═══════════════════════════════════════════════════════════════════════
# tools/vbox-wan-lab.sh - N machines VirtualBox, un noeud de WAN chacune
#
# Monte un WAN complet sur une seule machine : N VM (1 a 9, 3 par defaut)
# bootent la MEME ISO osmo-operator, sur un reseau interne commun, et chacune se
# reconnait comme le noeud dont l'IP est la sienne. Depuis un MS de la VM 2,
# composer <indicatif de la VM 1> + le numero joint un MS de la VM 1.
#
# POURQUOI UNE SEULE ISO POUR N VM
# Le numero du noeud n'est PAS fige dans l'image : la table l'est, et
# wan_nodes_detect_self compare les IP locales a cette table au demarrage.
# Une image par VM couterait N builds d'une heure pour une ligne de conf.
#
# C'est le DHCP de VirtualBox qui rend l'IP previsible : une adresse fixe par
# (VM, NIC), donc l'ISO retrouve toujours le meme numero de noeud. Sans ca, une
# VM changerait d'identite au gre des baux.
#
# Usage :
#   tools/vbox-wan-lab.sh check                # l'hote est-il pret ? (a lancer en premier)
#   tools/vbox-wan-lab.sh create   [--nodes 3] [--iso chemin.iso] [--build-iso]
#   tools/vbox-wan-lab.sh start | stop | status | destroy
#   tools/vbox-wan-lab.sh usb                  # peripheriques branches + filtres
#   tools/vbox-wan-lab.sh usb --vm 2 --add 0bda:2838:RTL-SDR
#   tools/vbox-wan-lab.sh table                # imprime la table sans rien creer
#
# Options de create :
#   --nodes N        nombre de VM/noeuds, 1 a 9 (defaut 3)
#   --iso FICHIER    ISO a demarrer (defaut : le plus recent du depot)
#   --build-iso      construit l'ISO avec la table deja dedans (sudo, long)
#   --subnet PREFIXE prefixe /24 du segment (defaut 192.168.56 - la seule plage
#                    que VirtualBox accepte en host-only sans configuration)
#   --ram Mo         memoire par VM (defaut 4096)
#   --cpus N         vCPU par VM (defaut 2)
#   --indicatifs "11 22 33"   indicatifs, un par noeud (defaut 11 22 33 ...)
#   --no-nat         supprime la 2e carte (NAT) - pas d'acces Internet dans les VM
#
# Materiel passe aux VM (tout est actif par defaut) :
#   --no-usb         pas de controleur USB du tout
#   --usb-filter vid:pid[:nom]   capte ce peripherique (repetable, s'ajoute a la
#                    liste connue : RTL-SDR, HackRF, bladeRF, LimeSDR, USRP B2xx,
#                    cables Calypso FT232/CP210x/PL2303/CH340...)
#   --no-audio       pas de carte son (la chaine audio GSM en a besoin)
#   --audio-driver X pulse | alsa | none (defaut : detecte)
#   --no-vrde        pas de console distante RDP
#   --share CHEMIN   dossier partage (necessite les additions invite dans l'ISO)
# ═══════════════════════════════════════════════════════════════════════
set -euo pipefail
```

```
RED='\033[0;31m'; GREEN='\033[0;32m'; YELLOW='\033[1;33m'
CYAN='\033[0;36m'; BOLD='\033[1m'; NC='\033[0m'

DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")/.." && pwd)"
# shellcheck source=network/wan-nodes.sh
. "$DIR/network/wan-nodes.sh"

NODES=3
# 192.168.56 : c'est la plage que VirtualBox autorise d'office en host-only
# (/etc/vbox/networks.conf est requis pour toute autre). On l'utilise aussi en
# reseau interne, ou rien ne l'impose, pour que les deux modes se ressemblent -
# passer de l'un a l'autre ne doit pas changer les adresses des noeuds.
SUBNET="192.168.56"
RAM=4096
CPUS=2
ISO=""
BUILD_ISO=0
WITH_NAT=1
INDS=""
NETNAME="osmowan"
VM_PREFIX="osmo-wan-"

# intnet  : les VM se parlent, l'hote NE VOIT RIEN du segment.
# hostonly: l'hote a une patte dessus - indispensable quand l'hote est lui-meme
#           un noeud du WAN (start-docker.sh --virtualbox). VirtualBox n'autorise alors
#           que 192.168.56.0/21, sauf /etc/vbox/networks.conf.
# Numero du noeud porte par l'HOTE lui-meme (vide = aucun, les N noeuds sont
# des VM). Des qu'il est pose, le segment doit etre host-only : l'hote a besoin
# d'une patte dessus pour parler a ses pairs.
# ISO du hub SS7. Fournie → une VM de plus, a ${SUBNET}.1, qui ne fait QUE
# router du M3UA entre les noeuds. Absente → pas de SS7 inter-noeud, seulement
# la voix et les SMS (c'est le comportement d'origine).
INTERSTP_ISO=""
INTERSTP_VM="osmo-wan-interstp"
HOST_NODE=""
NET_MODE="intnet"
HOSTONLY_IF=""
WITH_USB=1
WITH_AUDIO=1
WITH_VRDE=1
AUDIO_DRIVER=""
SHARE=""
EXTRA_FILTERS=()
USB_VM=""
USB_ADD=""
FIX=0

# ── Peripheriques captes automatiquement ──────────────────────────────
# Sans filtre, VirtualBox laisse TOUT a l'hote : brancher une cle SDR pendant
# que la VM tourne ne fait rien du tout, et rien ne le signale. Un filtre par
# modele regle ca une fois pour toutes - le peripherique bascule dans la VM des
# qu'il est branche, meme a chaud.
#
# Liste indicative : c'est "vbox-wan-lab.sh usb" qui dit la VERITE sur ce qui
# est branche et sous quel vid:pid. Tout ce qui manque s'ajoute avec
# --usb-filter vid:pid:nom, sans toucher a ce fichier.
USB_KNOWN=(
    "0bda:2838:RTL-SDR (RTL2838)"
    "0bda:2832:RTL-SDR (RTL2832U)"
    "1d50:6089:HackRF One"
    "1d50:604b:HackRF Jawbreaker"
    "1d50:6066:bladeRF x40/x115"
    "2cf0:5250:bladeRF 2.0 micro"
    "1d50:6108:LimeSDR-USB"
    "2500:0020:USRP B200"
```

```
    "2500:0021:USRP B210"
    "2500:0022:USRP B2xx"
    "0403:6001:Cable FT232 (Calypso)"
    "10c4:ea60:Cable CP210x (Calypso)"
    "067b:2303:Cable PL2303 (Calypso)"
    "1a86:7523:Cable CH340 (Calypso)"
    "16c0:0762:Osmocom SIMtrace"
    "1d50:60e3:Osmocom SIMtrace2"
)

ACTION="${1:-}"; [ $# -gt 0 ] && shift || true
while [ $# -gt 0 ]; do
    case "$1" in
        --nodes)       NODES="${2:-3}"; shift ;;
        --nodes=*)     NODES="${1#*=}" ;;
        --iso)         ISO="${2:-}"; shift ;;
        --iso=*)       ISO="${1#*=}" ;;
        --build-iso)   BUILD_ISO=1 ;;
        --subnet)      SUBNET="${2:-}"; shift ;;
        --subnet=*)    SUBNET="${1#*=}" ;;
        --ram)         RAM="${2:-}"; shift ;;
        --ram=*)       RAM="${1#*=}" ;;
        --cpus)        CPUS="${2:-}"; shift ;;
        --cpus=*)      CPUS="${1#*=}" ;;
        --indicatifs)  INDS="${2:-}"; shift ;;
        --indicatifs=*) INDS="${1#*=}" ;;
        --interstp-iso)   INTERSTP_ISO="${2:-}"; shift ;;
        --interstp-iso=*) INTERSTP_ISO="${1#*=}" ;;
        --no-nat)      WITH_NAT=0 ;;
        --net-mode)    NET_MODE="${2:-}"; shift ;;
        --net-mode=*)  NET_MODE="${1#*=}" ;;
        --hostonly)    NET_MODE="hostonly" ;;
        --host-node)   HOST_NODE="${2:-}"; NET_MODE="hostonly"; shift ;;
        --host-node=*) HOST_NODE="${1#*=}"; NET_MODE="hostonly" ;;
        --fix)         FIX=1 ;;
        --no-usb)      WITH_USB=0 ;;
        --no-audio)    WITH_AUDIO=0 ;;
        --no-vrde)     WITH_VRDE=0 ;;
        --audio-driver)   AUDIO_DRIVER="${2:-}"; shift ;;
        --audio-driver=*) AUDIO_DRIVER="${1#*=}" ;;
        --usb-filter)     EXTRA_FILTERS+=("${2:-}"); shift ;;
        --usb-filter=*)   EXTRA_FILTERS+=("${1#*=}") ;;
        --share)       SHARE="${2:-}"; shift ;;
        --share=*)     SHARE="${1#*=}" ;;
        --vm)          USB_VM="${2:-}"; shift ;;
        --vm=*)        USB_VM="${1#*=}" ;;
        --add)         USB_ADD="${2:-}"; shift ;;
        --add=*)       USB_ADD="${1#*=}" ;;
        --net)         NETNAME="${2:-}"; shift ;;
        --net=*)       NETNAME="${1#*=}" ;;
        -h|--help)     sed -n '2,33p' "$0" | sed 's/^# \{0,1\}//'; exit 0 ;;
        *) echo -e "${RED}option inconnue : $1${NC}" >&2; exit 2 ;;
    esac
    shift
done

[[ "$NODES" =~ ^[1-9]$ ]] || { echo -e "${RED}--nodes doit valoir 1 a 9${NC}" >&2; exit 2; }

VBM="$(command -v VBoxManage || command -v vboxmanage || true)"
[ -n "$VBM" ] || { echo -e "${RED}VBoxManage introuvable - installez VirtualBox.${NC}" >&2; exit 1; }
vbm() { "$VBM" "$@" 2>&1 | grep -v 'vboxdrv\|vboxconfig\|recompile the kernel\|You will not be able\|available for
 ↪  the current kernel\|load. Please\|^$' || true; }

# ── Piege `pipefail` + `grep -q` ─────────────────────────────────────────
# `grep -q` sort des la PREMIERE correspondance ; le producteur du pipeline
# recoit alors SIGPIPE et meurt en 141, et sous `set -o pipefail` c'est 141 que
```

```
# le pipeline renvoie. Resultat : "trouve" se lit "pas trouve", de facon
# intermittente - selon que la sortie tenait ou non dans le tampon du tube.
# C'est exactement ce qui faisait dire a `check` que vboxdrv n'etait pas charge
# alors qu'il l'etait. On lit donc toute l'entree, et on jette la sortie.
qgrep() { grep "$@" >/dev/null; }

# Ce que l'hote sait faire - determine UNE fois, utilise partout.
HAS_EXTPACK=0
"$VBM" list extpacks 2>/dev/null | qgrep 'Usable:.*true' && HAS_EXTPACK=1
if [ -z "$AUDIO_DRIVER" ]; then
    if pactl info >/dev/null 2>&1 || [ -S "/run/user/$(id -u)/pulse/native" ]; then AUDIO_DRIVER="pulse"
    elif [ -d /proc/asound ]; then AUDIO_DRIVER="alsa"
    else AUDIO_DRIVER="none"; fi
fi
USB_FILTER_COUNT=0

vm_name() { echo "${VM_PREFIX}$1"; }
node_ip()  { echo "${SUBNET}.$((10 + $1))"; }

# ── Table des noeuds ──────────────────────────────────────────────
build_table() {
    local i ind
    local -a ind_arr=()
    [ -n "$INDS" ] && read -r -a ind_arr <<< "$INDS"
    WAN_IP=(); WAN_IND=(); WAN_NODE_LIST=()
    for i in $(seq 1 "$NODES"); do
        ind="${ind_arr[$((i-1))]:-$(wan_default_ind "$i")}"
        # Un plan d'adressage sans exception : noeud N → .1N, que ce soit une VM
        # ou l'hote. L'hote recoit simplement la sienne sur l'interface
        # host-only au lieu de la recevoir par DHCP.
        WAN_IP[$i]="$(node_ip "$i")"
        WAN_IND[$i]="$ind"; WAN_NODE_LIST+=("$i")
    done
    WAN_NODE_COUNT="$NODES"; WAN_NODE_ID="${HOST_NODE:-1}"
    wan_nodes_validate || exit 1
}

find_iso() {
    local latest
    latest="$(ls -t "$DIR"/osmo-operator-*.iso 2>/dev/null | head -1)"
    [ -n "$latest" ] && { echo "$latest"; return 0; }
    return 1
}

# ── Reseau host-only : l'hote prend une patte sur le segment ──────
# Necessaire des que l'hote est lui-meme un noeud du WAN. Avec un reseau
# "interne", les VM se parlent mais l'hote ne voit RIEN - un lab docker sur
# l'hote ne pourrait pas joindre ses pairs, et rien ne le dirait.
setup_hostonly() {
    local existing
    existing="$("$VBM" list hostonlyifs 2>/dev/null | awk -F': *' '/^Name:/{print $2; exit}')"
    if [ -n "$existing" ]; then
        HOSTONLY_IF="$existing"
    else
        "$VBM" hostonlyif create >/dev/null 2>&1
        HOSTONLY_IF="$("$VBM" list hostonlyifs 2>/dev/null | awk -F': *' '/^Name:/{print $2; exit}')"
    fi
    [ -n "$HOSTONLY_IF" ] || { echo -e "${RED}creation de l'interface host-only impossible${NC}" >&2; return 1; }

    # L'interface porte l'adresse du noeud hote (.1N), pas une adresse a part :
    # ce que la table annonce est exactement ce que les pairs joindront.
    local host_ip; host_ip="$(node_ip "${HOST_NODE:-1}")"
    if ! "$VBM" hostonlyif ipconfig "$HOSTONLY_IF" --ip "$host_ip" --netmask 255.255.255.0 >/dev/null 2>&1; then
        # VirtualBox refuse toute plage hors 192.168.56.0/21 tant que
        # /etc/vbox/networks.conf ne l'autorise pas. Message explicite : sinon on
        # ne recolte qu'un "E_ACCESSDENIED" qui ne designe rien.
```

```
        echo -e "${RED}VirtualBox refuse ${host_ip} sur ${HOSTONLY_IF}.${NC}" >&2
        echo -e "  Les reseaux host-only sont limites a ${CYAN}192.168.56.0/21${NC} par defaut." >&2
        echo -e "  Soit : ${CYAN}--subnet 192.168.56${NC}" >&2
        echo -e "  Soit, pour autoriser votre plage :" >&2
        echo -e "    ${CYAN}echo '* ${SUBNET}.0/24' | sudo tee -a /etc/vbox/networks.conf${NC}" >&2
        return 1
    fi
    echo -e "  ${GREEN}✓${NC} ${HOSTONLY_IF} @ ${CYAN}${host_ip}${NC} (l'hote sur le segment, noeud
     ↪  ${HOST_NODE:-1})"
}

# ── Materiel d'une VM : USB, son, presse-papier, console distante ──────────
# Un noeud du lab est une maquette RADIO : sans cle SDR passee a la VM et sans
# carte son, il ne reste qu'un coeur reseau. C'est pour ca que tout est actif par
# defaut ici, contrairement a une VM de test ordinaire.
apply_hardware() {
    local vm="$1" idx="$2" f vid pid name i

    # ── USB ────────────────────────────────────────────────────────────────
    if [ "$WITH_USB" -eq 1 ]; then
        # xHCI (USB 3) demande l'Extension Pack ; sans lui on retombe sur OHCI
        # (USB 1.1), suffisant pour un cable serie Calypso mais PAS pour une
        # cle SDR - le debit ne suit pas et la capture decroche sans message clair.
        if [ "$HAS_EXTPACK" -eq 1 ]; then
            vbm modifyvm "$vm" --usb-xhci on
        else
            vbm modifyvm "$vm" --usb-ohci on
        fi
        i=0
        for f in "${USB_KNOWN[@]}" ${EXTRA_FILTERS[@]+"${EXTRA_FILTERS[@]}"}; do
            vid="${f%%:*}"; name="${f#*:}"; pid="${name%%:*}"; name="${name#*:}"
            [ "$name" = "$pid" ] && name="usb-${vid}-${pid}"
            vbm usbfilter add "$i" --target "$vm" --name "$name" \
                --vendorid "$vid" --productid "$pid" >/dev/null 2>&1 && i=$((i+1))
        done
        USB_FILTER_COUNT="$i"
    fi

    # ── Son ────────────────────────────────────────────────────────────────
    # ENTREE ET SORTIE : la chaine audio GSM (osmo-gapk) porte un appel dans les
    # deux sens. Une VM avec la seule sortie donne un appel a sens unique, ce qui
    # se diagnostique tres mal - on croit a un probleme de codec.
    if [ "$WITH_AUDIO" -eq 1 ]; then
        vbm modifyvm "$vm" --audio-driver "$AUDIO_DRIVER" --audio-controller hda \
            --audio-enabled on --audio-in on --audio-out on
    else
        vbm modifyvm "$vm" --audio-enabled off
    fi

    # ── Confort : presse-papier et glisser-deposer ─────────────────────────
    vbm modifyvm "$vm" --clipboard-mode bidirectional --drag-and-drop bidirectional

    # ── Console distante ───────────────────────────────────────────────────
    # Le lab demarre en headless : sans ca, aucun moyen de voir la console d'une
    # VM qui refuse de booter.
    if [ "$WITH_VRDE" -eq 1 ] && [ "$HAS_EXTPACK" -eq 1 ]; then
        vbm modifyvm "$vm" --vrde on --vrde-address 127.0.0.1 --vrde-port "$((3900 + idx))"
    fi

    # ── Dossier partage ────────────────────────────────────────────────────
    if [ -n "$SHARE" ]; then
        vbm sharedfolder add "$vm" --name osmo --hostpath "$SHARE" --automount
    fi
}

# ── create ─────────────────────────────────────────────────────────────────
```

```
do_create() {
    build_table
    echo -e "${CYAN}${BOLD}"
    echo "╔══════════════════════════════════════════════════════════════════════╗"
    printf "║   Banc WAN VirtualBox - %d noeud(s), reseau interne "%-8s ║\n" "$NODES" "${NETNAME}""
    echo "╚══════════════════════════════════════════════════════════════════════╝"
    echo -e "${NC}"
    wan_nodes_summary
    echo ""

    if [ "$BUILD_ISO" -eq 1 ]; then
        echo -e "${GREEN}[1/4] Construction de l'ISO avec la table embarquee...${NC}"
        echo -e "  ${CYAN}sudo $DIR/build-iso.sh --wan --wan-nodes \"$(wan_nodes_spec)\"${NC}"
        sudo "$DIR/build-iso.sh" --wan --wan-nodes "$(wan_nodes_spec)" \
            || { echo -e "${RED}build-iso.sh a echoue${NC}"; exit 1; }
        ISO="$(find_iso)" || { echo -e "${RED}ISO introuvable apres build${NC}"; exit 1; }
    elif [ -z "$ISO" ]; then
        ISO="$(find_iso)" || {
            echo -e "${RED}Aucune ISO. Passez --iso chemin.iso, ou --build-iso.${NC}" >&2; exit 1; }
        echo -e "${GREEN}[1/4] ISO : ${CYAN}${ISO}${NC}"
        echo -e "  ${YELLOW}Verifiez qu'elle a ete construite avec --wan et CETTE table,${NC}"
        echo -e "  ${YELLOW}sinon les VM demarreront sans WAN (start-native.sh --wan a la main).${NC}"
    else
        echo -e "${GREEN}[1/4] ISO : ${CYAN}${ISO}${NC}"
    fi
    [ -r "$ISO" ] || { echo -e "${RED}ISO illisible : $ISO${NC}" >&2; exit 1; }

    # ── Reseau + DHCP a adresses fixes ──────────────────────────────
    if [ "$NET_MODE" = "hostonly" ]; then
        echo -e "${GREEN}[2/4] Reseau host-only (l'hote est un noeud) + DHCP...${NC}"
        setup_hostonly || exit 1
        NETNAME="HostInterfaceNetworking-${HOSTONLY_IF}"
    else
        echo -e "${GREEN}[2/4] Reseau interne "${NETNAME}" + DHCP...${NC}"
    fi
    if "$VBM" list dhcpservers 2>/dev/null | qgrep "NetworkName:.*${NETNAME}"; then
        vbm dhcpserver modify --network="$NETNAME" \
            --server-ip="${SUBNET}.254" --netmask=255.255.255.0 \
            --lower-ip="${SUBNET}.100" --upper-ip="${SUBNET}.200" --enable
    else
        vbm dhcpserver add --network="$NETNAME" \
            --server-ip="${SUBNET}.254" --netmask=255.255.255.0 \
            --lower-ip="${SUBNET}.100" --upper-ip="${SUBNET}.200" --enable
    fi
    # Le DHCP est en .254, PAS en .1 : .1 est reservee a l'inter-STP, le hub
    # SS7 commun aux noeuds. C'est une adresse que toutes les configs des
    # noeuds designent en dur - elle ne peut pas dependre d'un bail.
    echo -e "  ${GREEN}✓${NC} DHCP ${SUBNET}.254 · inter-STP reserve en ${CYAN}${SUBNET}.1${NC}"

    # ── VM ──────────────────────────────────────────────────────────
    echo -e "${GREEN}[3/4] Creation des ${NODES} VM...${NC}"
    for i in $(seq 1 "$NODES"); do
        if [ "$i" = "$HOST_NODE" ]; then
            echo -e "  ${CYAN}-${NC} noeud ${i} : c'est CETTE machine ($(node_ip "$i")), pas de VM"
            continue
        fi
        local vm; vm="$(vm_name "$i")"
        if "$VBM" list vms 2>/dev/null | qgrep "\"${vm}\""; then
            echo -e "  ${YELLOW}⚠${NC} ${vm} existe deja - ignoree (destroy pour repartir a neuf)"
            continue
        fi
        vbm createvm --name "$vm" --ostype Ubuntu_64 --register
        vbm modifyvm "$vm" --memory "$RAM" --cpus "$CPUS" --ioapic on \
            --boot1 dvd --boot2 disk --boot3 none --boot4 none \
            --graphicscontroller vmsvga --vram 64 \
            --rtc-use-utc on --paravirt-provider kvm \
```

```
        --nested-hw-virt on
    # NIC1 = le WAN. C'est CETTE carte que la table adresse.
    if [ "$NET_MODE" = "hostonly" ]; then
        vbm modifyvm "$vm" --nic1 hostonly --host-only-adapter1 "$HOSTONLY_IF" --nictype1 virtio
    else
        vbm modifyvm "$vm" --nic1 intnet --intnet1 "$NETNAME" --nictype1 virtio
    fi
    apply_hardware "$vm" "$i"
    if [ "$WITH_NAT" -eq 1 ]; then
        # NIC2 = NAT : sortie Internet (apt, osmo-update) et redirection SSH
        # depuis l'hote. Elle ne porte AUCUN trafic WAN - son 10.0.2.15
        # n'est pas dans la table, donc la detection de noeud ne s'y trompe pas.
        vbm modifyvm "$vm" --nic2 nat --nictype2 virtio
        vbm modifyvm "$vm" --natpf2 "ssh${i},tcp,127.0.0.1,$((2220 + i)),,22"
        # Tableau de bord et spectres du lab, atteignables depuis l'hote
        # sans passer par le segment WAN (qui, lui, doit rester propre).
        vbm modifyvm "$vm" --natpf2 "web${i},tcp,127.0.0.1,$((8080 + (i-1)*10)),,8080"
        vbm modifyvm "$vm" --natpf2 "fft${i},tcp,127.0.0.1,$((8081 + (i-1)*10)),,8081"
    fi
    vbm storagectl "$vm" --name IDE --add ide
    vbm storageattach "$vm" --storagectl IDE --port 0 --device 0 --type dvddrive --medium "$ISO"
    # Adresse fixe : c'est elle qui donne son IDENTITE au noeud.
    vbm dhcpserver modify --network="$NETNAME" --vm="$vm" --nic=1 \
        --fixed-address="$(node_ip "$i")" \
        || echo -e "  ${YELLOW}⚠${NC} adresse fixe refusee pour ${vm} - VirtualBox < 6.1 ?"
    echo -e "  ${GREEN}✓${NC} ${vm}  ${CYAN}$(node_ip "$i")${NC}  indicatif ${CYAN}${WAN_IND[$i]}${NC}$([
     ↪  "$WITH_NAT" -eq 1 ] && echo "  ssh 127.0.0.1:$((2220 + i))")"
done

# ── VM inter-STP : le hub SS7, a l'adresse .1 reservee ──────────────────
if [ -n "$INTERSTP_ISO" ]; then
    if "$VBM" list vms 2>/dev/null | qgrep "\"${INTERSTP_VM}\""; then
        echo -e "  ${YELLOW}⚠${NC} ${INTERSTP_VM} existe deja - ignoree"
    elif [ ! -r "$INTERSTP_ISO" ]; then
        echo -e "  ${RED}✗${NC} ISO inter-STP illisible : ${INTERSTP_ISO}"
    else
        vbm createvm --name "$INTERSTP_VM" --ostype Ubuntu_64 --register
        # Le hub ne porte ni radio ni audio : il route du M3UA. On ne lui
        # donne donc ni son, ni USB, ni les 4 Go d'un noeud complet.
        vbm modifyvm "$INTERSTP_VM" --memory 1536 --cpus 1 --ioapic on \
            --boot1 dvd --boot2 disk --boot3 none --boot4 none \
            --graphicscontroller vmsvga --vram 16 --rtc-use-utc on \
            --paravirt-provider kvm --audio-enabled off
        if [ "$NET_MODE" = "hostonly" ]; then
            vbm modifyvm "$INTERSTP_VM" --nic1 hostonly --host-only-adapter1 "$HOSTONLY_IF" --nictype1 virtio
        else
            vbm modifyvm "$INTERSTP_VM" --nic1 intnet --intnet1 "$NETNAME" --nictype1 virtio
        fi
        [ "$WITH_NAT" -eq 1 ] && vbm modifyvm "$INTERSTP_VM" --nic2 nat --nictype2 virtio \
            --natpf2 "sshstp,tcp,127.0.0.1,2229,,22"
        vbm storagectl "$INTERSTP_VM" --name IDE --add ide
        vbm storageattach "$INTERSTP_VM" --storagectl IDE --port 0 --device 0 \
            --type dvddrive --medium "$INTERSTP_ISO"
        vbm dhcpserver modify --network="$NETNAME" --vm="$INTERSTP_VM" --nic=1 \
            --fixed-address="${SUBNET}.1" \
            || echo -e "  ${YELLOW}⚠${NC} adresse fixe refusee pour ${INTERSTP_VM}"
        echo -e "  ${GREEN}✓${NC} ${INTERSTP_VM}  ${CYAN}${SUBNET}.1${NC}  hub SS7 (PC 0.0.0)"
    fi
fi

echo -e "${GREEN}[4/4] Materiel passe aux VM :${NC}"
if [ "$WITH_USB" -eq 1 ]; then
    echo -e "  USB        : ${CYAN}$([ "$HAS_EXTPACK" -eq 1 ] && echo 'xHCI (USB 3)' || echo 'OHCI (USB 1.1,
     ↪  sans Extension Pack)')${NC}, ${USB_FILTER_COUNT} filtre(s)"
    if ! id -nG "${SUDO_USER:-$USER}" 2>/dev/null | qgrep -w vboxusers; then
        echo -e "  ${YELLOW}⚠ ${SUDO_USER:-$USER} n'est pas dans le groupe vboxusers :${NC}"
```

```
            echo -e "    ${YELLOW}les peripheriques USB resteront a l'hote, sans message d'erreur.${NC}"
            echo -e "    ${CYAN}sudo usermod -aG vboxusers ${SUDO_USER:-$USER}${NC}  puis rouvrir la session"
        fi
    else
        echo -e "  USB        : ${YELLOW}desactive (--no-usb)${NC}"
    fi
    echo -e "  Son        : ${CYAN}${AUDIO_DRIVER}${NC} (entree + sortie)"
    [ "$WITH_VRDE" -eq 1 ] && [ "$HAS_EXTPACK" -eq 1 ] \
        && echo -e "  Console    : ${CYAN}RDP 127.0.0.1:390N${NC} (N = numero de noeud)"
    [ -n "$SHARE" ] && echo -e "  Partage    : ${CYAN}${SHARE}${NC} → /media/sf_osmo (additions invite requises)"
    echo ""
    if ! grep -q '^vboxdrv ' /proc/modules 2>/dev/null; then
        echo ""
        echo -e "  ${YELLOW}vboxdrv n'est pas charge : les VM sont creees mais ne demarreront pas.${NC}"
        echo -e "  ${CYAN}sudo /sbin/vboxconfig${NC}  (recompile le module pour le noyau courant)"
    fi
    do_hint
}

do_hint() {
    build_table
    echo ""
    echo -e "  ${BOLD}Demarrer :${NC} ${CYAN}$0 start${NC}"
    echo ""
    echo -e "  ${BOLD}Dans chaque VM (autologin root) :${NC}"
    echo -e "    ${CYAN}/opt/GSM/osmo-operator/start-native.sh${NC}"
    echo -e "    → la table WAN figee dans l'ISO est appliquee toute seule (WAN_AUTO=1)."
    echo -e "    → sans ISO --wan : ${CYAN}/opt/GSM/osmo-operator/start-native.sh --wan${NC}"
    echo ""
    echo -e "  ${BOLD}Appeler d'un noeud a l'autre :${NC}"
    local a b
    a="${WAN_NODE_LIST[0]}"; b="${WAN_NODE_LIST[$(( ${#WAN_NODE_LIST[@]} > 1 ? 1 : 0 ))]}"
    echo -e "    depuis un MS du noeud ${a} : ${CYAN}${WAN_IND[$b]}10001${NC} → MS 10001 du noeud ${b}"
    echo -e "    SMS : meme numerotation (${CYAN}${WAN_IND[$b]}10001${NC})"
    echo ""
    echo -e "  ${BOLD}Diagnostic dans une VM :${NC} ${CYAN}/opt/GSM/osmo-operator/checks/wan_ss7_check.sh${NC}"
    echo ""
}

do_each() {  # $1 = commande vboxmanage sur chaque VM existante
    local i vm found=0
    # Le hub d'abord au demarrage : un noeud qui monte avant lui voit sa SCTP
    # refusee et ne reessaie pas forcement.
    for i in interstp $(seq 1 9); do
        if [ "$i" = "interstp" ]; then vm="$INTERSTP_VM"; else vm="$(vm_name "$i")"; fi
        "$VBM" list vms 2>/dev/null | qgrep "\"${vm}\"" || continue
        found=1
        case "$1" in
            start)   vbm startvm "$vm" --type headless && echo -e "  ${GREEN}▶${NC} ${vm}" ;;
            stop)    vbm controlvm "$vm" acpipowerbutton >/dev/null 2>&1 || true
                     echo -e "  ${YELLOW}■${NC} ${vm}" ;;
            destroy) vbm controlvm "$vm" poweroff >/dev/null 2>&1 || true
                     sleep 1
                     vbm unregistervm "$vm" --delete >/dev/null 2>&1 || true
                     echo -e "  ${RED}✗${NC} ${vm} supprimee" ;;
            status)
                    local st ip; st="$("$VBM" showvminfo "$vm" --machinereadable 2>/dev/null | awk -F'"'
                     ↪  '/^VMState=/{print $2}')"
                    if [ "$i" = "interstp" ]; then ip="${SUBNET}.1 (hub SS7)"; else ip="$(node_ip "$i")"; fi
                    printf '  %-20s %-10s %s\n' "$vm" "${st:-?}" "$ip" ;;
        esac
    done
    [ "$found" = "1" ] || echo -e "  ${YELLOW}aucune VM ${VM_PREFIX}* - lancez "$0 create"${NC}"
}

# Les groupes REELS d'une session ouverte de $1 : on lit /proc/<pid>/status
```

```
# d'un de ses processus. `id -nG` ne conviendrait pas - il interroge la base,
# pas la session, et repondrait "oui" juste apres un usermod alors que rien
# n'a change pour les programmes deja lances.
session_has_group() {
    local user="$1" grp="$2" gid pid
    gid="$(getent group "$grp" | cut -d: -f3)"
    [ -n "$gid" ] || return 1
    for pid in $(pgrep -u "$user" 2>/dev/null | head -40); do
        [ -r "/proc/$pid/status" ] || continue
        awk -v g="$gid" '/^Groups:/ { for (i=2; i<=NF; i++) if ($i == g) { found=1 } }
                         END { exit !found }' "/proc/$pid/status" && return 0
    done
    return 1
}

# ── check : l'hote est-il capable de faire tourner le banc ? ─────────────
do_check() {
    local rc=0 u="${SUDO_USER:-$USER}"
    echo -e "${BOLD}Hote VirtualBox${NC}"
    printf '─%.0s' $(seq 1 60); echo ""

    echo -e "  ${GREEN}✓${NC} VBoxManage $("$VBM" --version 2>/dev/null | tail -1)"

    if grep -q '^vboxdrv ' /proc/modules 2>/dev/null; then
        echo -e "  ${GREEN}✓${NC} module vboxdrv charge"
    else
        echo -e "  ${RED}✗${NC} vboxdrv non charge - aucune VM ne demarrera"
        echo -e "      ${CYAN}sudo /sbin/vboxconfig${NC}"
        if command -v mokutil >/dev/null 2>&1 && mokutil --sb-state 2>/dev/null | qgrep -i enabled; then
            echo -e "      ${YELLOW}Secure Boot est actif : le module doit etre signe et sa cle enrolee (MOK).${NC}"
        fi
        rc=1
    fi

    if [ "$HAS_EXTPACK" -eq 1 ]; then
        echo -e "  ${GREEN}✓${NC} Extension Pack - USB 2.0/3.0 et console RDP disponibles"
    else
        echo -e "  ${YELLOW}⚠${NC} pas d'Extension Pack : USB 1.1 seulement (insuffisant pour une cle SDR), pas de
         ↪  RDP"
    fi

    # Deux etats distincts, et c'est TOUTE la difficulte de ce point :
    #   • le fichier /etc/group peut deja contenir l'utilisateur...
    #   • ...alors que sa SESSION en cours tourne encore avec l'ancienne liste de
    #     groupes. Les processus heritent de leurs groupes au login : `usermod`
    #     ne les rejoint pas retroactivement. VirtualBox lance depuis cette
    #     session-la ne peut toujours pas ouvrir /dev/vboxusb - et l'erreur
    #     affichee parle d'un peripherique "occupe", jamais de groupes.
    # On regarde donc les DEUX : la base, et les groupes reels d'un processus
    # vivant de l'utilisateur.
    if id -nG "$u" 2>/dev/null | qgrep -w vboxusers; then
        if session_has_group "$u" vboxusers; then
            echo -e "  ${GREEN}✓${NC} ${u} est dans vboxusers, et sa session l'a bien pris"
        else
            echo -e "  ${YELLOW}⚠${NC} ${u} est dans vboxusers, mais ${BOLD}sa session ouverte ne l'a pas
             ↪  encore${NC}"
            echo -e "      Les processus gardent les groupes recus au login."
            echo -e "      ${CYAN}Deconnectez-vous et reconnectez-vous${NC} (ou redemarrez) - puis relancez ce
             ↪  check."
            echo -e "      Pour tester tout de suite, sans quitter la session : ${CYAN}newgrp vboxusers${NC}"
            rc=1
        fi
    else
        echo -e "  ${RED}✗${NC} ${u} n'est PAS dans vboxusers - les filtres USB ne capteront rien"
        if [ "$FIX" -eq 1 ] && [ "$(id -u)" -eq 0 ]; then
```

```
            usermod -aG vboxusers "$u" && echo -e "     ${GREEN}✓ ajoute${NC} - deconnectez-vous et
             ↪  reconnectez-vous pour l'activer"
        else
            echo -e "     ${CYAN}sudo usermod -aG vboxusers ${u}${NC}  puis rouvrir la session"
            echo -e "     (ou relancez : ${CYAN}sudo $0 check --fix${NC})"
        fi
        rc=1
    fi

    echo -e "  ${GREEN}✓${NC} son : pilote ${CYAN}${AUDIO_DRIVER}${NC}"

    if find_iso >/dev/null 2>&1; then
        echo -e "  ${GREEN}✓${NC} ISO : $(find_iso)"
    else
        echo -e "  ${YELLOW}⚠${NC} aucune ISO dans le depot - ${CYAN}$0 create --build-iso${NC}"
    fi

    echo ""
    echo -e "${BOLD}Peripheriques USB branches${NC}"
    printf '─%.0s' $(seq 1 60); echo ""
    do_usb_list
    echo ""
    [ $rc -eq 0 ] && echo -e "  ${GREEN}Hote pret.${NC}" || echo -e "  ${YELLOW}Corrigez les points ci-dessus
     ↪  avant de creer le banc.${NC}"
    return $rc
}

# Liste les peripheriques de l'hote et dit lesquels un filtre du banc capterait.
do_usb_list() {
    local out vid pid name known f kv kp
    out="$("$VBM" list usbhost 2>/dev/null)"
    if [ -z "$out" ]; then echo -e "  ${YELLOW}aucun peripherique USB visible${NC}"; return 0; fi
    echo "$out" | awk -v RS='' '{print}' | while IFS= read -r line; do :; done
    printf '%s\n' "$out" | awk '
        /^VendorId:/  { v=$0; sub(/.*0x/,"",v); sub(/ .*/,"",v); vid=v }
        /^ProductId:/ { p=$0; sub(/.*0x/,"",p); sub(/ .*/,"",p); pid=p }
        /^Product:/   { pr=$0; sub(/^Product: */,"",pr); prod=pr }
        /^Manufacturer:/ { m=$0; sub(/^Manufacturer: */,"",m); man=m }
        /^Current State:/ { st=$0; sub(/^Current State: */,"",st)
                            if (vid != "") printf "%s:%s|%s|%s|%s\n", vid, pid, man, prod, st
                            vid=""; pid=""; prod=""; man="" }
    ' | while IFS='|' read -r ids man prod st; do
        known=""
        for f in "${USB_KNOWN[@]}" ${EXTRA_FILTERS[@]+"${EXTRA_FILTERS[@]}"}; do
            kv="${f%%:*}"; kp="${f#*:}"; kp="${kp%%:*}"
            [ "${kv}:${kp}" = "$ids" ] && { known="${f#*:*:}"; break; }
        done
        if [ -n "$known" ]; then
            printf '  ✓ %-12s %-34s → capte (%s)\n' "$ids" "${prod:-$man}" "$known"
        else
            printf '    %-12s %-34s %s\n' "$ids" "${prod:-$man}" "$st"
        fi
    done
    echo ""
    echo -e "  ${CYAN}Pour capter un peripherique absent de la liste :${NC}"
    echo -e "    $0 usb --vm 1 --add ${CYAN}vid:pid:nom${NC}"
}

do_usb() {
    if [ -n "$USB_ADD" ]; then
        [ -n "$USB_VM" ] || { echo -e "${RED}--add exige --vm N${NC}" >&2; exit 2; }
        local vm vid pid name idx
        vm="$(vm_name "$USB_VM")"
       "$VBM" list vms 2>/dev/null | qgrep "\"${vm}\"" || { echo -e "${RED}${vm} n'existe pas${NC}" >&2; exit 1; }
        vid="${USB_ADD%%:*}"; name="${USB_ADD#*:}"; pid="${name%%:*}"; name="${name#*:}"
        [ "$name" = "$pid" ] && name="usb-${vid}-${pid}"
```

```
        idx="$("$VBM" showvminfo "$vm" --machinereadable | grep -c '^USBFilterName')"
        vbm usbfilter add "$idx" --target "$vm" --name "$name" --vendorid "$vid" --productid "$pid"
        echo -e "  ${GREEN}✓${NC} ${vm} captera ${CYAN}${vid}:${pid}${NC} (${name})"
        echo -e "  ${CYAN}Actif au prochain demarrage de la VM, ou au prochain branchement.${NC}"
    else
        do_usb_list
    fi
}

case "$ACTION" in
    check)   do_check ;;
    usb)     do_usb ;;
    create)  do_create ;;
    start)   echo -e "${GREEN}Demarrage...${NC}"; do_each start ;;
    stop)    echo -e "${YELLOW}Arret...${NC}";    do_each stop ;;
    destroy) echo -e "${RED}Suppression...${NC}"; do_each destroy
             vbm dhcpserver remove --network="$NETNAME" >/dev/null 2>&1 || true ;;
    status)  echo -e "${BOLD}VM du banc WAN :${NC}"; do_each status ;;
    table)   build_table; wan_nodes_summary
             echo ""; echo "  --wan-nodes \"$(wan_nodes_spec)\"" ;;
    ""|-h|--help) sed -n '2,33p' "$0" | sed 's/^# \{0,1\}//' ;;
    *) echo -e "${RED}action inconnue : $ACTION${NC}" >&2; exit 2 ;;
esac
```

3.449 /opt/GSM/osmo-operator/tools/vty-connect.exp

756 octets, 30 lignes → 30 lignes

```
#!/usr/bin/env expect

set container [lindex $argv 0]
set port [lindex $argv 1]
set vty_type [lindex $argv 2]
# On récupère l'IP du groupe, par défaut 127.0.0.1
set target_ip [lindex $argv 3]
if {$target_ip == ""} { set target_ip "127.0.0.1" }

# On se connecte à l'IP spécifique du groupe Baseband
spawn sudo docker exec -ti $container telnet $target_ip $port

expect {
    "Connection refused" { exit 1 }
    "Connected to" { }
    ">"
}

# La suite du script reste identique pour activer les logs automatiques
send "enable\r"
expect "#"
send "logging enable\r"
expect "#"
send "logging level set-all notice\r"
expect "#"
send "logging filter all 1\r"
expect "#"

puts "\n--- CONSOLE VTY ACTIVE : $vty_type ($target_ip:$port) ---"
interact
```

3.450 /opt/GSM/osmo-operator/tools/vty-menu.sh

6162 octets, 156 lignes → 156 lignes

```
#!/bin/bash
# [2026-10-12] le port VTY du mobile : 4247, ou 4347 en --dsp (publie par start-native.sh)
BB_VTY="$(tr -dc 0-9 < /run/osmo-banc/ms1-vty 2>/dev/null)"; BB_VTY="${BB_VTY:-4247}"

# Sauvegarde de la configuration actuelle du terminal
OLD_SETTINGS=$(stty -g)
# Sauvegarde des couleurs initiales (si supporte)
INITIAL_BG=$(printf '\e]11;?\a')

# Palette de couleurs "Nord Theme" (Plus elegant)
set_kernel_profile() {
    # On change le fond et le texte
    echo -ne "\e]11;#2E3440\a" # Fond bleu-gris fonce
    echo -ne "\e]10;#D8DEE9\a" # Texte blanc casse
    clear
}

reset_terminal_profile() {
    # 1. Restauration des couleurs d'origine du profil utilisateur
    echo -ne "\e]110\a" # Reset Foreground
    echo -ne "\e]111\a" # Reset Background
    echo -ne "\e]112\a" # Reset cursor color

    # 2. Commande systeme pour reset le terminal
    tput reset
    tput sgr0

    # 3. Restauration des parametres stty
    stty "$OLD_SETTINGS"

    clear
    echo "--- Session SDR terminee, profil restaure ---"
}

# Theme Whiptail harmonise
export NEWT_COLORS='
  root=,rgb:2e/34/40
  window=rgb:d8/de/e9,rgb:3b/42/52
  shadow=,rgb:1a/1c/23
  title=rgb:88/c0/d0,rgb:3b/42/52
  button=rgb:2e/34/40,rgb:81/a1/c1
  actbutton=rgb:d8/de/e9,rgb:5e/81/ac
  listbox=rgb:d8/de/e9,rgb:3b/42/52
  actlistbox=rgb:88/c0/d0,rgb:43/4c/5e
'

set_kernel_profile
trap reset_terminal_profile EXIT

connect_vty() {
    tput cnorm # S'assure que le curseur est visible
    expect ./tools/vty-connect.exp "$1" "$3" "$4"
    [[ $? -ne 0 ]] && { echo -e "\n[!] Service hors ligne."; sleep 1; }
}

# ... (garder le debut du fichier identique) ...

menu_services() {
    local OP=$1
    while true; do
        # Ajout de l'option 7 pour le Baseband
        CHOICE=$(whiptail --title " OPERATOR CONTROL : $OP " --menu "" 20 62 9 \
            "1" "MSC (4254)" "2" "BSC (4242)" "3" "HLR (4258)" \
            "4" "MGW (4243)" "5" "GGSN (4260)" "6" "SGSN (4245)" \
            "7" "STP (4239)" "8" "BASEBAND ($BB_VTY)" \
            "T" "TMUX - la pile en direct (session osmo)" \
```

```
            "R" "<< BACK" 3>&1 1>&2 2>&3)

        [[ -z "$CHOICE" || "$CHOICE" == "R" ]] && break

        case $CHOICE in
            1) connect_vty "$OP" "MSC" 4254 "msc" ;;
            2) connect_vty "$OP" "BSC" 4242 "bsc" ;;
            3) connect_vty "$OP" "HLR" 4258 "hlr" ;;
            4) connect_vty "$OP" "MGW" 4243 "mgw" ;;
            5) connect_vty "$OP" "GGSN" 4260 "gg" ;;
            6) connect_vty "$OP" "SGSN" 4245 "sg" ;;
            7) connect_vty "$OP" "STP" 4239 "stp" ;;
            T|t)
                # La session que start-docker.sh ouvre dans chaque conteneur operateur.
                # Le terminal de lancement est pris par osmo-operator-1 ; c'est
                # par ici qu'on rejoint les autres, sans avoir a retenir la
                # commande docker.
                tput cnorm
                # Trois conventions cohabitent selon le conteneur :
                #   osmocom sur /tmp/osmocom_tmux  la pile lancee par run.sh
                #   calypso / gapk sur le socket par defaut  le pipeline QEMU
                # On les cherche dans cet ordre plutot que d'en imposer une :
                # une session annoncee mais introuvable ("no sessions") laisse
                # croire que la pile est morte alors qu'elle tourne.
                _sock=""; _sess=""
                if docker exec "$OP" tmux -S /tmp/osmocom_tmux has-session -t osmocom 2>/dev/null; then
                    _sock="-S /tmp/osmocom_tmux"; _sess="osmocom"
                elif docker exec "$OP" tmux has-session -t calypso 2>/dev/null; then
                    _sess="calypso"
                elif docker exec "$OP" tmux has-session -t osmo 2>/dev/null; then
                    _sess="osmo"
                fi
                if [ -n "$_sess" ]; then
                    echo "  Session ${_sess} - detacher : Ctrl-b puis d"
                    # shellcheck disable=SC2086
                    docker exec -ti "$OP" tmux $_sock attach -t "$_sess"
                else
                   whiptail --msgbox "Pas de session tmux 'osmo' dans $OP.\n\nLe journal reste lisible :\n  docker
                    ↪  exec $OP tail -f /var/log/osmocom/run.sh.log" 12 66
                fi
                ;;
            8)
                # Gestion multi-groupes Baseband (127.0.0.1, 127.0.0.2, etc.)
                local GRP_IP
                GRP_IP=$(whiptail --title " BASEBAND GROUP SELECT " --inputbox "Target IP (127.0.0.X) :" 10 40
                ↪  "127.0.0.1" 3>&1 1>&2 2>&3)
                if [[ -n "$GRP_IP" ]]; then
                    # On passe l'IP du groupe au script expect
                    tput cnorm
                    expect ./tools/vty-connect.exp "$OP" "$BB_VTY" "bb" "$GRP_IP"
                fi
                ;;
        esac
    done
}

# --- BOUCLE PRINCIPALE SECURISEE ---
while true; do
    MAIN_TYPE=$(whiptail --title " OSMO-SDR STACK MANAGER " --menu "Select Category :" 15 60 3 \
        "CORE" "Global Infrastructure (Inter-STP)" \
        "OPS"  "Network Operators (MSC/BSC/STP...)" \
        "EXIT" "Quit Manager" 3>&1 1>&2 2>&3)

    [[ -z "$MAIN_TYPE" || "$MAIN_TYPE" == "EXIT" ]] && break

    if [[ "$MAIN_TYPE" == "CORE" ]]; then
```

```
        # Verification si l'inter-stp est en vie
        if docker ps --format '{{.Names}}' | grep -q "osmo-inter-stp"; then
            docker exec -ti osmo-inter-stp telnet 127.0.0.1 4239
        else
            whiptail --msgbox "Erreur : osmo-inter-stp n'est pas lance." 10 50
        fi
    else
        # On recupere les operateurs actifs
        OPS=$(docker ps --format "{{.Names}}" | grep "osmo-operator" | sort)

        if [[ -z "$OPS" ]]; then
            whiptail --msgbox "Aucun operateur detecte ! Verifiez start-docker.sh" 10 50
            continue
        fi

        MENU_LIST=()
        while read -r line; do MENU_LIST+=("$line" "Status: Active"); done <<< "$OPS"

        OP_CHOICE=$(whiptail --title " OPERATOR SELECTION " --menu "Select Operator Node :" 20 70 10 \
            "${MENU_LIST[@]}" 3>&1 1>&2 2>&3)

        [[ -n "$OP_CHOICE" ]] && menu_services "$OP_CHOICE"
    fi
done

reset_terminal_profile
```

3.451 /opt/GSM/osmo-operator/tools/wallpaper-render.py

26625 octets, 499 lignes → 499 lignes

```
#!/usr/bin/env python3
# wallpaper-render.py - compose le fond d ecran du banc (1920x1080).
#
#   python3 wallpaper-render.py --tower configs/wallpaper/tower.png \
#           [--strip calvin.gif] [--date 2026-09-04] --out gsm-lab-wallpaper.png
#
# Disposition, pensee pour ne rien faire chevaucher sur l ecran :
#
#   +--------+-----------------------------------------+------------------+
#   | dock + | carte LAB GSM (x 320..1430, y 60..560)  | conky (400 px,   |
#   | icones |                                         | gap 24, y 40..660)|
#   | bureau | encart (x 510..1410, CENTRE, y 220..1010) |                  |
#   | (haut  |   haut : BD geek au hasard / spectre 4G  |                  |
#   | gauche)|   bas  : Calvin & Hobbes  / spectre 2G   |                  |
#   +--------+-----------------------------------------+------------------+
#
# Le pylone de la photo est au MILIEU (voir background/TOWER_X) : les deux
# panneaux sont translucides et le laissent passer.
# Les deux colonnes de cote ne recoivent que la photo. A gauche (x < 320) : le
# dock GNOME (~70 px) puis les icones du bureau, en haut a gauche (DING,
# start-corner top-left, cases de ~100 px). A droite (x >= 1440) : conky
# (top_right, gap_x 24, 400 px de large, cf. configs/conky/osmo-conky.conf).
#
# Sans --strip (pas de reseau au build, ou au boot), la zone du strip reste
# vide : le fond est complet sans lui. Le strip du jour est pose par
# tools/osmo-wallpaper.sh (service + timer osmo-wallpaper).
#
# Ne depend que de Pillow (python3-pil) et des polices DejaVu (fonts-dejavu),
# presents sur l image comme sur l hote.

# [2026-09-04] L ENCART EST AU MILIEU BAS. Sa boite etait (320, 600, 1430, 1010) :
# calee a gauche, sous la carte LAB GSM et sur la meme verticale qu elle. Depuis
# que le pylone de la photo est recentre (tools/wallpaper-render.py, TOWER_X),
```

```
# cet encart decale n avait plus de raison d etre - et il n a jamais ete au
# milieu de l ecran, seulement au milieu de la bande laissee libre.
# La nouvelle boite (510, 600, 1410, 1010) est CENTREE sur les 1920 px
# (510 + 900/2 = 960) et s arrete avant x=1440, ou commence la colonne du conky.
# [2026-09-08] L ENCART A GRANDI VERS LE HAUT ET S EST COUPE EN DEUX. La 4G du
# banc (srsRAN) n avait pas de place a l ecran : la boite passe a
# (510, 220, 1410, 1010), soit 900x790, en deux cadres de 391 px separes de
# 8 px. Le haut porte une bande dessinee geek tiree au sort (et le spectre
# du UE srsRAN quand il tourne), le bas Calvin & Hobbes (et le spectre 2G).
# Elle empiete sur la zone haute LAB GSM (osmo-topzone.py, y 60..~400) :
# c est assume, la banniere reste un decor derriere.
# LES QUATRE FICHIERS QUI LA CONNAISSENT DOIVENT RESTER D ACCORD :
#   tools/wallpaper-render.py  (les deux cadres dessines dans le fond)
#   tools/osmo-fft-snap.py     (l image 900x790 qu on y peint)
#   tools/osmo-panel.py        (la fenetre GTK cliquable posee dessus)
#   tools/osmo-launcher.py     (ou se cale une appli lancee « dans le panel »)
BOX_HAUT = (510, 220, 1410, 611)
BOX_BAS = (510, 619, 1410, 1010)
import argparse
import datetime
import os
import sys

from PIL import Image, ImageChops, ImageDraw, ImageEnhance, ImageFilter, ImageFont

W, H = 1920, 1080
FONT_DIR = "/usr/share/fonts/truetype/dejavu"


def font(name, size):
    try:
        return ImageFont.truetype(os.path.join(FONT_DIR, name), size)
    except OSError:
        return ImageFont.load_default()


def lerp(a, b, t):
    return tuple(int(a[i] + (b[i] - a[i]) * t) for i in range(3))


def gradient_h(w, h, stops):
    """Bande horizontale w x h, degrade entre les couleurs `stops`."""
    img = Image.new("RGB", (w, h))
    px = img.load()
    n = len(stops) - 1
    for x in range(w):
        t = x / max(1, w - 1) * n
        i = min(int(t), n - 1)
        c = lerp(stops[i], stops[i + 1], t - i)
        for y in range(h):
            px[x, y] = c
    return img


def rounded(draw, box, r, fill=None, outline=None, width=1):
    draw.rounded_rectangle(box, radius=r, fill=fill, outline=outline, width=width)


# ── GOMMER LES DEFAUTS DE LA PHOTO ──────────────────────────────────
# [2026-09-04] La photo est prise EN CONTRE-JOUR, au telephone, et elle en porte
# les trois marques. Elles ne se voyaient pas tant que le pylone etait un
# lointain decor a gauche ; au milieu et agrandi x1.9, elles sautent aux yeux :
#
#   1. UN VOILE MAGENTA. Le soleil dans l objectif renvoie un halo diagonal
#      violace en travers du ciel - un fantome de diaphragme. Sur un fond bleu
#      nuit, il ressort comme une trainee sale que personne ne sait nommer.
```

```
#   2. DU FRANGEAGE CHROMATIQUE. Les montants sombres sur ciel brule prennent
#      un lisere violet (aberration laterale). L accentuation qui suit le
#      RENFORCAIT : on accentuait le defaut avec le detail.
#   3. DU BLOCKING JPEG. Le ciel est un degrade quasi uni : c est exactement ce
#      que le JPEG rend le plus mal (carres de 8x8 et bandes). L agrandissement
#      les etale en plaques visibles.
#
# Trois corrections, ciblees, dans cet ordre - et AVANT l accentuation, sinon
# on accentue ce qu on s apprete a gommer.
#
# OSMO_WP_CLEAN=0 les neutralise (pour comparer, ou pour une autre photo).
def nettoyer_photo(im):
    """Defrangeage magenta + lissage du ciel, sans toucher au treillis."""
    # [2026-09-04] PAR DEFAUT : NE RIEN FAIRE. Ce nettoyage a ete ecrit pour la
    # photo PRECEDENTE - un contre-jour au telephone, avec son voile de flare,
    # son frangeage et son blocking JPEG. La photo actuelle est propre, et
    # surtout elle est ROSE : un ciel de couchant, ou le vert est legitimement
    # sous la moyenne du rouge et du bleu sur la moitie de l image. La
    # correction magenta y verrait un defaut partout et REPEINDRAIT LE CIEL en
    # gris - elle detruirait exactement ce qui fait l image.
    # Le code reste, documente : OSMO_WP_CLEAN=1 le rallume pour une photo qui
    # en aurait besoin.
    if os.environ.get("OSMO_WP_CLEAN", "0") != "1":
        return im

    # ── 1 et 2. LE MAGENTA, VOILE ET LISERES, D UN SEUL GESTE ──────────────
    # Un ciel a R < G < B. Le halo de flare comme le frangeage ont la meme
    # signature : le VERT tombe sous la moyenne du rouge et du bleu (c est la
    # definition du magenta). On remonte donc le vert jusqu a cette moyenne, la
    # ou il est dessous. Ce qui est reellement magenta dans la scene - rien,
    # ici : du ciel et de l acier - n existe pas, donc rien de legitime n est
    # touche. Une correction en une passe, sans masque a regler.
    r, g, b = im.split()
    moy = ImageChops.add(r.point(lambda v: v // 2), b.point(lambda v: v // 2))
    g = ImageChops.lighter(g, moy)
    im = Image.merge("RGB", (r, g, b))

    # ── 3. LE CIEL LISSE, LE TREILLIS INTACT ───────────────────────────────
    # On ne peut pas flouter toute l image : le pylone y perdrait ce que
    # l accentuation lui rend juste apres. On construit donc un masque de
    # DETAIL - l ecart entre l image et sa version floue, c est-a-dire les
    # arretes - qu on elargit (MaxFilter) pour couvrir les bords, puis qu on
    # adoucit pour que la transition ne se voie pas. Le flou n est applique que
    # LA OU IL N Y A PAS DE DETAIL : le ciel, et lui seul.
    flou = im.filter(ImageFilter.GaussianBlur(2.2))
    gris = im.convert("L")
    ecart = ImageChops.difference(gris, gris.filter(ImageFilter.GaussianBlur(2.2)))
    masque = ecart.point(lambda v: 255 if v > 4 else 0)
    masque = masque.filter(ImageFilter.MaxFilter(5)).filter(ImageFilter.GaussianBlur(1.6))
    return Image.composite(im, flou, masque)


# ── OU TOMBE LE PYLONE SUR L ECRAN ─────────────────────────────────────────
# [2026-09-04] IL EST REVENU AU MILIEU. Il etait cale a GAUCHE : la photo etait
# decalee pour poser le SOLEIL en bas a gauche du strip, et le pylone suivait -
# il finissait derriere la colonne des icones du bureau, a moitie mange par le
# dock. Sur l ecran, le sujet de l image passait derriere les objets qui ne
# demandaient qu a etre lisibles.
#
# On ancre donc sur le PYLONE et non sur le soleil. Le mat traverse la photo
# d origine (1170x1123) autour de x = 658 : c est ce point qu on amene a
# TOWER_X (0.62 : entre la carte LAB GSM et la colonne du conky - le plein
# centre, 0.5, agrandissait trop la photo et noyait le soleil hors cadre),
# qui sont translucides - le pylone se lit a travers).
#
# L ECHELLE SE DEDUIT DE LA POSITION VOULUE, elle n est plus le simple "couvrir
```

```
# l ecran". Recadrer sans agrandir laissait un vide sur un cote, que la version
# precedente comblait par un miroir du ciel. On calcule la plus petite echelle
# qui met le pylone a sa place SANS vide d aucun cote - le miroir devient un
# filet de securite au lieu d etre la regle.
# x du mat, et taille de reference de configs/wallpaper/tower.jpg.
# [2026-09-04] Nouvelle photo (pylone hertzien au couchant, 3840x2160) : le mat
# y est a 34,2 % de la largeur, mesure en cherchant la colonne la plus
# STRUCTUREE de la bande centrale (somme du detail image-flou, colonne par
# colonne) - le pic est franc, sans ambiguite. L ancienne photo avait le sien a
# 658/1170 ; le rapport est ce qui compte, background() le remet a l echelle du
# fichier reel, donc remplacer l image ne demande que cette ligne.
# [2026-09-04] La photo est prise TELLE QUELLE, en PNG, a sa definition
# d origine (5824x3264) : la reduire ou la passer en JPEG pour alleger, c est
# perdre de la qualite sur la seule image que l ecran montre en grand. Le rendu
# la RE-DUIT de toute facon (echelle ~0.57 pour caler le mat), et reduire depuis
# la pleine definition donne un meilleur resultat que reduire deux fois.
TOWER_SRC = (1992, 5824, 3264)
TOWER_X = float(os.environ.get("OSMO_WP_TOWER_X", "0.62"))


def background(tower_path, tower_x=None, sun_y_at=984):
    """La photo en fond : le pylone AU MILIEU (tower_x, fraction de la largeur
    d ecran ; 0.5 par defaut). Assombrie au centre (les panneaux sont poses
    dessus) et teintee bleu nuit."""
    tower = nettoyer_photo(Image.open(tower_path).convert("RGB"))
    tw, th = tower.size
    src_x, ref_w, _ref_h = TOWER_SRC
    # Le repere du mat suit la taille reelle du fichier : une photo remplacee
    # par une autre du meme cadrage reste correcte.
    src_x = src_x * tw / ref_w
    target = (TOWER_X if tower_x is None else tower_x) * W
    # Trois contraintes : couvrir en hauteur, et ne deborder ni a gauche ni a
    # droite une fois le mat amene sur `target`.
    scale = max(H / th,
                target / max(src_x, 1),
                (W - target) / max(tw - src_x, 1))
    tower = tower.resize((int(tw * scale) + 1, int(th * scale) + 1), Image.LANCZOS)
    # ── ACCENTUER LE PYLONE ──────────────────────────────────────────────
    # [2026-09-04] La photo est un CONTRE-JOUR : le soleil est derriere le mat,
    # qui sort donc en silhouette sombre sur un ciel brule, et l agrandissement
    # (x1.9 pour amener le pylone au milieu) etale ce peu de detail sur deux
    # fois plus de pixels. A l ecran, le treillis devenait une bouillie grise -
    # on voyait qu il y avait « quelque chose », pas une antenne.
    #
    # Trois gestes, dans cet ordre, et pas un de plus :
    #   1. un masque flou (unsharp) large et doux : il redonne l arete des
    #      montants et des croix sans creuser le grain du ciel ;
    #   2. le contraste, legerement : la silhouette se detache du halo ;
    #   3. la nettete, un cheveu : les cables et les panneaux d antenne.
    # Applique AVANT la teinte et l assombrissement : accentuer apres, ce
    # serait accentuer un voile bleu, pas la photo.
    #
    # Les trois valeurs sont reglables (OSMO_WP_SHARPEN=0 les neutralise) :
    # sur une autre photo, ou un ecran tres dense, le bon dosage n est pas le
    # meme, et une image sur-accentuee est pire qu une image molle.
    # [2026-09-04] 1 -> 0.45. L ancienne photo faisait 1170 px de large et etait
    # AGRANDIE x1.9 : il fallait beaucoup pour recoller les details etales. La
    # nouvelle fait 3840 px et se retrouve REDUITE (echelle ~0.9) - or reduire
    # affute deja. Garder 1 donnait des halos sur les haubans et un ciel
    # granuleux. Un cheveu suffit maintenant, juste pour compenser le
    # reechantillonnage.
    _shp = float(os.environ.get("OSMO_WP_SHARPEN", "0.45"))
    if _shp > 0:
        tower = tower.filter(ImageFilter.UnsharpMask(radius=2.4 * _shp,
                                                     percent=int(135 * _shp), threshold=3))
        tower = ImageEnhance.Contrast(tower).enhance(1 + 0.16 * _shp)
```

```
        tower = ImageEnhance.Sharpness(tower).enhance(1 + 0.35 * _shp)
    x0 = int(round(src_x * scale - target))
    x0 = max(0, min(x0, tower.width - W))
    # Verticalement, le soleil reste ou il etait : en bas a gauche du strip.
    sun_y = int(600 * scale * th / _ref_h)
    y0 = max(0, min(sun_y - sun_y_at, tower.height - H))
    bg = Image.new("RGB", (W, H))
    part = tower.crop((x0, y0, min(tower.width, x0 + W), y0 + H))
    bg.paste(part, (0, 0))
    gap = W - part.width
    if gap > 0:
        # Ne devrait plus arriver (l echelle est calculee pour), mais une photo
        # d un autre cadrage pourrait l imposer : miroir du bord, ciel uni.
        edge = tower.crop((tower.width - gap, y0, tower.width, y0 + H)).transpose(Image.FLIP_LEFT_RIGHT)
        bg.paste(edge, (part.width, 0))
    # Teinte bleu nuit + assombrissement : le pylone est au centre, donc SOUS
    # les deux panneaux - ils sont translucides, il faut qu il transparaisse
    # sans rendre le texte gris. D ou un assombrissement plus doux qu avant au
    # centre (85 -> 70) et symetrique.
    tint = Image.new("RGB", (W, H), (10, 18, 40))
    bg = Image.blend(bg, tint, 0.18)
    mask = gradient_h(W, 1, [(0, 0, 0), (70, 70, 70), (70, 70, 70), (0, 0, 0)]).resize((W, H))
    dark = Image.new("RGB", (W, H), (6, 10, 24))
    bg = Image.composite(dark, bg, mask.convert("L"))
    return bg


def glass_panel(base, box, radius=26, alpha=185, fill=(13, 22, 44), border=(90, 120, 200), blur=14):
    """Panneau translucide, fond legerement flou dessous (verre depoli)."""
    x0, y0, x1, y1 = box
    if blur > 0:
        region = base.crop(box).filter(ImageFilter.GaussianBlur(blur))
        base.paste(region, (x0, y0))
    overlay = Image.new("RGBA", base.size, (0, 0, 0, 0))
    d = ImageDraw.Draw(overlay)
    rounded(d, box, radius, fill=fill + (alpha,), outline=border + (110,), width=2)
    base.paste(Image.alpha_composite(base.convert("RGBA"), overlay).convert("RGB"))


def gradient_text(base, xy, text, fnt, stops):
    """Texte rempli d un degrade horizontal."""
    x, y = xy
    l, t, r, b = fnt.getbbox(text)
    w, h = r - l, b - t
    mask = Image.new("L", (w + 4, h + 4), 0)
    ImageDraw.Draw(mask).text((-l + 2, -t + 2), text, font=fnt, fill=255)
    grad = gradient_h(w + 4, h + 4, stops)
    base.paste(grad, (x, y), mask)
    return w, h


def card(base, box, arfcn="514", band="DCS 1800"):
    x0, y0, x1, y1 = box
    # Carte bien transparente : le pylone et le ciel restent visibles derriere
    # le texte (le strip, lui, garde un fond opaque pour rester lisible).
    glass_panel(base, box, alpha=38, blur=0)
    d = ImageDraw.Draw(base)
    pad = 44
    x = x0 + pad
    y = y0 + 34

    # En-tete
    d.ellipse((x, y + 4, x + 14, y + 18), fill=(63, 185, 160))
    d.text((x + 28, y), "SOFTWARE DEFINED RADIO  ·  UM INTERFACE  ·  GR-GSM",
           font=font("DejaVuSansMono.ttf", 15), fill=(150, 170, 210))
    tag = f"ARFCN {arfcn}  ·  {band}"
```

```
tf = font("DejaVuSansMono-Bold.ttf", 15)
tw = d.textlength(tag, font=tf)
rounded(d, (x1 - pad - tw - 28, y - 8, x1 - pad, y + 28), 10, outline=(63, 185, 160), width=2)
d.text((x1 - pad - tw - 14, y), tag, font=tf, fill=(120, 230, 210))

# Titre
y += 46
# Titre en violet uni (pas de degrade).
tw, th = gradient_text(base, (x - 4, y), "LAB GSM", font("DejaVuSans-Bold.ttf", 86),
                       [(150, 100, 245), (150, 100, 245)])
d = ImageDraw.Draw(base)
y += th + 20
sub = font("DejaVuSans.ttf", 22)
subb = font("DejaVuSans-Bold.ttf", 22)
line1 = [("Un réseau 2G complet, ", sub), ("sans matériel", subb),
         (" — baseband ", sub), ("Calypso", subb), (" émulé, cœur ", sub), ("Osmocom", subb), (",", sub)]
line2 = [("de la synchro radio à l'appel voix chiffré en ", sub), ("A5/1", subb),
         ("  ·  couche 1 ", sub), ("gr-gsm", subb), (" (SDR)", sub)]
for line in (line1, line2):
    cx = x
    for txt, f in line:
        d.text((cx, y), txt, font=f, fill=(215, 225, 245))
        cx += d.textlength(txt, font=f)
    y += 32

# Trame TDMA
y += 14
d.text((x, y), "TRAME TDMA  ·  8 TIMESLOTS  ·  4,615 MS", font=font("DejaVuSansMono.ttf", 14),
       fill=(150, 170, 210))
y += 26
slots = [("TS0", "FCCH·SCH·BCCH"), ("TS1", "SDCCH"), ("TS2", "TCH/F"), ("TS3", "TCH/F"),
         ("TS4", "TCH/F"), ("TS5", "TCH/F"), ("TS6", "PDCH"), ("TS7", "PDCH")]
inner = (x1 - pad) - x
gap = 10
sw = (inner - gap * 7) // 8
# [2026-09-10] EN HAUTEUR, PAS EN LARGEUR - meme geste que dans la banniere
# vivante (configs/conky/labgsm.html) : le chronogramme du normal burst est
# parti, et ce sont les huit cases qui reprennent ses ~100 px. La largeur
# (sw, calculee sur la carte) ne bouge pas ; seul sh grandit, et le texte
# garde sa taille, centre verticalement dans la case.
sh = 148
for i, (name, use) in enumerate(slots):
    sx = x + i * (sw + gap)
    active = i == 0
    rounded(d, (sx, y, sx + sw, y + sh), 10,
            fill=(58, 62, 150) if active else (24, 34, 70),
            outline=(120, 130, 240) if active else (60, 75, 130), width=2)
    nf = font("DejaVuSansMono-Bold.ttf", 16)
    uf = font("DejaVuSansMono.ttf", 11)
    cy = y + sh / 2
    d.text((sx + (sw - d.textlength(name, font=nf)) / 2, cy - 24), name, font=nf, fill=(235, 240, 255))
    d.text((sx + (sw - d.textlength(use, font=uf)) / 2, cy + 2), use, font=uf,
           fill=(120, 230, 210) if active else (140, 160, 200))
y += sh + 8
d.text((x, y), "0 µs", font=font("DejaVuSansMono.ttf", 12), fill=(120, 140, 180))
mid = "1 burst = 577 µs"
d.text((x + inner / 2 - d.textlength(mid, font=font("DejaVuSansMono.ttf", 12)) / 2, y), mid,
       font=font("DejaVuSansMono.ttf", 12), fill=(120, 140, 180))
end = "4615 µs"
d.text((x1 - pad - d.textlength(end, font=font("DejaVuSansMono.ttf", 12)), y), end,
       font=font("DejaVuSansMono.ttf", 12), fill=(120, 140, 180))

# [2026-09-10] Le chronogramme du normal burst (3 tail | 57 data | 26
# training | 57 data | 8,25 guard) etait dessine ici. Retire : c est un
# schema de cours, fixe, qui ne dit rien de l etat du banc - alors que les
# timeslots au-dessus, eux, montrent le plan de canaux reel. Sa hauteur est
```

```
    # passee aux cases des timeslots (sh, plus haut).
    y += 22

    # Puces en bas
    chips = [("IMSI 001-01", (150, 170, 210)), ("A5/1 ✓", (120, 230, 210)), ("COMP128v1 ✓", (120, 230, 210)),
             ("TCH/F · GAPK fr", (120, 230, 210)), ("gr-gsm · trxcon · osmo-bts-trx", (150, 170, 210))]
    cf = font("DejaVuSansMono-Bold.ttf", 13)
    cx = x
    for txt, col in chips:
        cw = d.textlength(txt, font=cf) + 22
        rounded(d, (cx, y, cx + cw, y + 28), 8, outline=col, width=1)
        d.text((cx + 11, y + 6), txt, font=cf, fill=col)
        cx += cw + 10
    site = "bbaranoff.github.io"
    sf = font("DejaVuSansMono-Bold.ttf", 14)
    d.text((x1 - pad - d.textlength(site, font=sf), y + 6), site, font=sf, fill=(180, 200, 240))


def strip_panel(base, box, strip_path, date_str, credit=None):
    """L image du jour, dans un cadre blanc arrondi.

    `credit` est la ligne de pied : elle NOMME la source. Elle etait ecrite en
    dur (« Calvin & Hobbes ... gocomics.com ») du temps ou il n y avait qu une
    source ; depuis que osmo-wallpaper.sh en tire une au sort, une legende figee
    aurait attribue a Bill Watterson un xkcd ou une photo de la NASA.
    """
    x0, y0, x1, y1 = box
    glass_panel(base, box, fill=(20, 24, 36), alpha=210, border=(200, 200, 210))
    d = ImageDraw.Draw(base)
    pad = 18
    # ── PLUS DE LIGNE DE SOURCE DANS LE CADRE ──────────────────────────
    # [2026-09-04] Ce pied portait « Bing · image du jour · Phare de
    # Westerheversand a Westerhever, Schleswig-Holstein, Allemagne (© ...) ».
    # Deux problemes, et le second suffit :
    #   - il passait SOUS la barre de boutons de l encart (Dashboard / tmux /
    #     VTY, plus les fleches), qui vit exactement la : on lisait une moitie
    #     de phrase derriere « Dashboard op2 » ;
    #   - il n a rien a faire la. Ce cadre est l ENCART DU BANC : des que la
    #     pile monte, il porte un spectre et un journal, et la provenance d une
    #     photo n y a aucun sens.
    # La place rendue va a l image, qui remplit maintenant tout le cadre.
    # OSMO_WP_CREDIT=1 remet la ligne pour qui la veut.
    foot = 30 if os.environ.get("OSMO_WP_CREDIT") == "1" else 0
    strip = Image.open(strip_path)
    strip.seek(0)
    strip = strip.convert("RGB")
    sw, sh = strip.size
    aw, ah = (x1 - x0) - 2 * pad, (y1 - y0) - 2 * pad - foot
    scale = min(aw / sw, ah / sh)
    strip = strip.resize((int(sw * scale), int(sh * scale)), Image.LANCZOS)
    sx = x0 + pad + (aw - strip.width) // 2
    sy = y0 + pad
    # Cadre blanc derriere le strip (les GIF ont des marges irregulieres).
    d.rounded_rectangle((sx - 6, sy - 6, sx + strip.width + 6, sy + strip.height + 6), radius=8,
                        fill=(255, 255, 255))
    base.paste(strip, (sx, sy))
    if foot == 0:
        return
    d = ImageDraw.Draw(base)
    cap = credit or f"Calvin & Hobbes  ·  Bill Watterson  ·  {date_str}  ·  gocomics.com"
    cf = font("DejaVuSansMono.ttf", 13)
    # ── LA LEGENDE NE DOIT PAS SORTIR DU CADRE ─────────────────────────
    # [2026-09-04] Elle etait ecrite sans aucune limite de largeur. Tant que le
    # cadre faisait 1110 px, les legendes tenaient ; a 900 px (l encart recentre
    # au milieu bas), le credit Bing - « Phare de Westerheversand a Westerhever,
    # Schleswig-Holstein, Allemagne (© bluejayphoto/Getty Images) » - DEBORDAIT
```

```
    # sur le bureau, a droite du panneau. Et comme l encart vivant
    # (osmo-fft-snap.py) recouvre exactement le cadre et rien de plus, ce qui
    # depassait restait visible SOUS le spectre et le mobile.log : un bout de
    # phrase orpheline en travers du fond, sans rien pour l expliquer.
    #
    # On coupe donc a la largeur utile, avec une ellipse. Mesurer puis tronquer,
    # et non deviner un nombre de caracteres : la police est a chasse fixe ici,
    # mais la legende peut contenir des accents et des symboles.
    avail = (x1 - x0) - 2 * pad
    if d.textlength(cap, font=cf) > avail:
        while cap and d.textlength(cap + "...", font=cf) > avail:
            cap = cap[:-1]
        cap = cap.rstrip(" ·") + "..."
    d.text((x0 + pad, y1 - foot + 6), cap, font=cf, fill=(160, 170, 190))


def main():
    ap = argparse.ArgumentParser()
    ap.add_argument("--tower", required=True)
    ap.add_argument("--strip", default=None, help="l image du cadre du BAS (Calvin & Hobbes)")
    ap.add_argument("--strip-haut", default=None, help="l image du cadre du HAUT (la BD geek)")
    ap.add_argument("--credit-haut", default=None)
    ap.add_argument("--date", default=datetime.date.today().isoformat())
    ap.add_argument("--credit", default=None,
                    help="ligne de pied sous l image (source, auteur). "
                         "Defaut : la legende Calvin & Hobbes.")
    ap.add_argument("--out", required=True)
    ap.add_argument("--arfcn", default="514")
    ap.add_argument("--band", default="DCS 1800")
    ap.add_argument("--tower-x", type=float, default=None,
                    help="ou poser le pylone, en fraction de la largeur d ecran "
                         "(0.5 = milieu, le defaut ; 0.17 = l ancien calage a gauche)")
    a = ap.parse_args()

    base = background(a.tower, a.tower_x)
    # [2026-09-06] LA CARTE LAB GSM N EST PLUS CUITE DANS LE FOND, POINT.
    # C est tools/osmo-topzone.py (widget WebKit vivant) qui la rend, avec les
    # timeslots qui clignotent et les valeurs ARFCN/IMSI/A5 live. La version
    # dessinee ici etait la MEME carte, en plus grand et fixe : sur un bureau ou
    # le widget tourne, on voyait les deux, la peinte en fond derriere la
    # vivante. C etait pilote par OSMO_LIVE_BANNER=1, que seul le wrapper du
    # bureau exportait - tout autre chemin de rendu (osmo-wallpaper.sh a la
    # main, un service, une regeneration apres coup) ramenait le doublon. On
    # inverse donc le defaut : plus rien de cuit, et OSMO_LEGACY_BANNER=1 pour
    # qui voudrait l ancien fond autonome (un banc sans bureau vivant).
    if os.environ.get("OSMO_LEGACY_BANNER", "0") == "1":
        card(base, (320, 60, 1430, 560), a.arfcn, a.band)
    # Deux cadres, deux images. Un cadre sans image reste dessine (vide) : le
    # spectre viendra le couvrir, et vide il dit « rien a montrer ici ».
    poses = {}
    for nom, box, chemin, credit in (("bas", BOX_BAS, a.strip, a.credit),
                                     ("haut", BOX_HAUT, a.strip_haut, a.credit_haut)):
        poses[nom] = False
        if chemin and os.path.isfile(chemin):
            try:
                strip_panel(base, box, chemin, a.date, credit)
                poses[nom] = True
            except Exception as e:  # GIF corrompu, page HTML au lieu d une image...
                print(f"[wallpaper] image du {nom} ignoree : {e}", file=sys.stderr)
        else:
            glass_panel(base, box, fill=(20, 24, 36), alpha=210, border=(200, 200, 210))
    tmp = a.out + ".tmp.png"
    base.save(tmp, "PNG", optimize=True)
    os.replace(tmp, a.out)
    print(f"[wallpaper] {a.out} ({W}x{H}, bas={'oui' if poses['bas'] else 'non'}, "
          f"haut={'oui' if poses['haut'] else 'non'})")
```

```
if __name__ == "__main__":
    main()
```

3.452 /opt/GSM/osmo-operator/tools/softsim/0001-simos-comp128v1-a38.patch

3323 octets, 103 lignes → 103 lignes

```
diff --git a/src/demo_server.rb b/src/demo_server.rb
index e89b69d..4bbe12d 100755
--- a/src/demo_server.rb
+++ b/src/demo_server.rb
@@ -107,23 +107,30 @@ when "unix"
 else
   raise "please defined which socket to use"
 end
-# wait for a client to connect
-io = socket.accept
+# serve clients one after the other: a new connection (mobile "sim sap") must
+# not need a server restart after the previous one disconnected
+loop do
+  # wait for a client to connect
+  puts "[sap] en attente d une connexion sur #{@unix || @port}"
+  io = socket.accept
+  puts "[sap] client connecte"

-case @type
-when "pcsc"
-  require_relative 'pcsc_server'
-  server = PCSCServer.new(io)
-when "sim"
-  require_relative 'simos_server'
-  server = SIMServer.new(io)
-else
-  raise "unkown server type"
-end
+  case @type
+  when "pcsc"
+    require_relative 'pcsc_server'
+    server = PCSCServer.new(io)
+  when "sim"
+    require_relative 'simos_server'
+    server = SIMServer.new(io, @file)
+  else
+    raise "unkown server type"
+  end

-# starting the server
-server.start
+  # starting the server
+  server.start

-# close IO when finished
-io.close
-server.close
+  # close IO when finished
+  io.close
+  server.close
+  puts "[sap] device disconnected : pret pour une reconnexion"
+end
diff --git a/src/simos_server.rb b/src/simos_server.rb
index 3c7a52a..80f6691 100644
--- a/src/simos_server.rb
```

```
+++ b/src/simos_server.rb
@@ -165,6 +165,31 @@ class SIMServer < Server
           sw = [0x90,0x00]
         end
       end
+    when 0xb2 # READ RECORD
+      # P1 = numero d enregistrement, P2 = 0x04 (absolu, P1 valide), P3 = longueur
+      # (Le) de l enregistrement, pour un EF lineaire fixe (ou cyclique) seulement.
+      type = file_info
+      if type[:type]!="EF" then
+        # no EF selected
+        sw = [0x94,0x00]
+      elsif !["linear fixed","cyclic"].include?(type[:structure]) then
+        # file is inconsitent with the command
+        sw = [0x94,0x08]
+      elsif request[3]!=0x04 then
+        # incorrect parameter P2 (seul le mode absolu est gere)
+        sw = [0x6b,0x00]
+      else
+        body = @selected.find_first("./body").content.hex2arr
+        record = request[2]
+        length = request[4]
+        if length==0 or record==0 or (record-1)*length+length>body.length then
+          # record not found / out of range
+          sw = [0x94,0x02]
+        else
+          data = body[(record-1)*length,length]
+          sw = [0x90,0x00]
+        end
+      end
     when 0xd6 # UPDATE BINARY
       # is an transparent ef selected ?
       type = file_info
@@ -205,6 +230,14 @@ class SIMServer < Server
         if tuple then
           @response = tuple["sres"].hex2arr+tuple["kc"].hex2arr
           sw = [0x9f,@response.length]
+        elsif ENV["SOFTSIM_KI"] && ENV["SOFTSIM_KI"].length == 32
+          # COMP128v1 avec le Ki de la SIM (patch local, lib/comp128.rb) : repond a
+          # n importe quel RAND du HLR, pas seulement aux triplets enregistres.
+          require_relative 'lib/comp128'
+          ki = [ENV["SOFTSIM_KI"]].pack("H*")
+          sres, kc = Comp128.v1(ki, request[5,16].pack("C*"))
+          @response = sres.unpack("C*") + kc.unpack("C*")
+          sw = [0x9f,@response.length]
         else
           # memory problem
           sw = [0x92,0x40]
```

3.453 /opt/GSM/osmo-operator/tools/softsim/0002-sap-serve-next-client.patch

500 octets, 15 lignes → 15 lignes

```
diff --git a/src/sap/common.rb b/src/sap/common.rb
index 9054e9f..a1d3862 100644
--- a/src/sap/common.rb
+++ b/src/sap/common.rb
@@ -183,7 +183,9 @@ class SAP
       rescue EOFError
         $stderr.puts "device disconnected"
         @io.close
-        exit 0
+        # return, not exit: demo_server.rb serves the next client (the mobile
+        # reconnects after every baseband restart)
```

```
+        return
       end
       # parse the message from @buffer_in to @messages_in
       parse_messages
```

3.454 /opt/GSM/osmo-operator/tools/softsim/0003-sap-log-timestamps.patch

1929 octets, 40 lignes → 40 lignes

```
diff --git a/src/sap/common.rb b/src/sap/common.rb
--- a/src/sap/common.rb
+++ b/src/sap/common.rb
@@ -454,7 +454,10 @@
   def log (group,message,level)
     if $verbosity and $verbosity>=level then
       color = 95-level
-      puts "\e[1m\e[#{color}m[#{group}]\e[0m #{message}"
+      # timestamp (ms) : lets one measure the time between two APDUs
+      ts = Time.now.strftime("%H:%M:%S.%L")
+      puts "#{ts} \e[1m\e[#{color}m[#{group}]\e[0m #{message}"
+      $stdout.flush
     end
   end

diff --git a/src/sap/server.rb b/src/sap/server.rb
--- a/src/sap/server.rb
+++ b/src/sap/server.rb
@@ -138,9 +138,18 @@
         set_state :processing_apdu_request
         # apdu should return APDU response byte array, or error result code
         raise "no APDU request in message" unless message[:payload].size==1
-        log("APDU","> #{hex(message[:payload][0][:value])}",1)
-        apdu_result = apdu(message[:payload][0][:value])
-        log("APDU","< #{hex(apdu_result)}",1)
+        ins_names = {0xa4=>"SELECT",0xc0=>"GET RESPONSE",0xb0=>"READ BINARY",
+                     0xb2=>"READ RECORD",0xd6=>"UPDATE BINARY",0xdc=>"UPDATE RECORD",
+                     0x88=>"RUN GSM ALGORITHM",0xf2=>"STATUS",0x20=>"VERIFY CHV",
+                     0x26=>"DISABLE CHV",0x28=>"ENABLE CHV",0x24=>"CHANGE CHV",
+                     0x2c=>"UNBLOCK CHV",0x04=>"INVALIDATE",0x44=>"REHABILITATE",
+                     0xfa=>"SLEEP",0x12=>"FETCH",0x14=>"TERMINAL RESPONSE"}
+        apdu_in = message[:payload][0][:value]
+        ins = ins_names[apdu_in[1]] || "INS 0x%02x" % apdu_in[1].to_i
+        log("APDU","> #{ins} #{hex(apdu_in)}",1)
+        t0 = Time.now
+        apdu_result = apdu(apdu_in)
+        log("APDU","< #{hex(apdu_result)} (#{ins}, #{((Time.now-t0)*1000).round(2)} ms)",1)
         payload = []
         if apdu_result.kind_of?(Array) then
           # ["ResultCode",["OK, request processed correctly"]]
```

3.455 /opt/GSM/osmo-operator/tools/softsim/0004-pcsc-disconnect-idempotent.patch

905 octets, 29 lignes → 29 lignes

```
diff --git a/src/pcsc_server.rb b/src/pcsc_server.rb
index ab3820d..082381d 100644
--- a/src/pcsc_server.rb
+++ b/src/pcsc_server.rb
@@ -104,9 +104,22 @@ class PCSCServer < Server
   end
```

```
   # release card and context
+  # Called twice per client: on DISCONNECT_REQ (sap/server.rb) and again by
+  # demo_server.rb (server.close) once the socket is gone. The second call hit
+  # the smartcard gem with a nil handle (TypeError) and killed the server, so the
+  # mobile could never reconnect. Idempotent now, and a lost reader is not fatal.
   def disconnect
-    @card.disconnect
-    @context.release
+    begin
+      @card.disconnect(:leave) if @card
+    rescue Exception => e
+      $stderr.puts "pcsc: disconnect ignored (#{e.class}: #{e.message})"
+    end
+    @card = nil
+    begin
+      @context.release if @context
+    rescue Exception
+    end
+    @context = nil
   end

   # get ATR
```

3.456 /opt/GSM/osmo-operator/tools/softsim/comp128.rb

7307 octets, 138 lignes → 138 lignes

```ruby
# frozen_string_literal: true
# tools/softsim/comp128.rb - COMP128 v1 (A3/A8 de GSM) en Ruby.
#
# Port de libosmocore src/gsm/comp128.c (GPL-2.0+, (C) 2009 Sylvain Munaut),
# lui-meme derive du document GSM 10-1617-01 (1988) et corrige sur trois
# coquilles connues. Valide octet pour octet contre le C (voir tests).
#
#   sres, kc = Comp128.v1(ki_bytes, rand_bytes)   # String binaire, 16 octets chacun
module Comp128
  TABLE_0 = [
    102, 177, 186, 162,   2, 156, 112,  75,  55,  25,   8,  12, 251, 193, 246, 188,
    109, 213, 151,  53,  42,  79, 191, 115, 233, 242, 164, 223, 209, 148, 108, 161,
    252,  37, 244,  47,  64, 211,   6, 237, 185, 160, 139, 113,  76, 138,  59,  70,
     67,  26,  13, 157,  63, 179, 221,  30, 214,  36, 166,  69, 152, 124, 207, 116,
    247, 194,  41,  84,  71,   1,  49,  14,  95,  35, 169,  21,  96,  78, 215, 225,
    182, 243,  28,  92, 201, 118,   4,  74, 248, 128,  17,  11, 146, 132, 245,  48,
    149,  90, 120,  39,  87, 230, 106, 232, 175,  19, 126, 190, 202, 141, 137, 176,
    250,  27, 101,  40, 219, 227,  58,  20,  51, 178,  98, 216, 140,  22,  32, 121,
     61, 103, 203,  72,  29, 110,  85, 212, 180, 204, 150, 183,  15,  66, 172, 196,
     56, 197, 158,   0, 100,  45, 153,   7, 144, 222, 163, 167,  60, 135, 210, 231,
    174, 165,  38, 249, 224,  34, 220, 229, 217, 208, 241,  68, 206, 189, 125, 255,
    239,  54, 168,  89, 123, 122,  73, 145, 117, 234, 143,  99, 129, 200, 192,  82,
    104, 170, 136, 235,  93,  81, 205, 173, 236,  94, 105,  52,  46, 228, 198,   5,
     57, 254,  97, 155, 142, 133, 199, 171, 187,  50,  65, 181, 127, 107, 147, 226,
    184, 218, 131,  33,  77,  86,  31,  44,  88,  62, 238,  18,  24,  43, 154,  23,
     80, 159, 134, 111,   9, 114,   3,  91,  16, 130,  83,  10, 195, 240, 253, 119,
    177, 102, 162, 186, 156,   2,  75, 112,  25,  55,  12,   8, 193, 251, 188, 246,
    213, 109,  53, 151,  79,  42, 115, 191, 242, 233, 223, 164, 148, 209, 161, 108,
     37, 252,  47, 244, 211,  64, 237,   6, 160, 185, 113, 139, 138,  76,  70,  59,
     26,  67, 157,  13, 179,  63,  30, 221,  36, 214,  69, 166, 124, 152, 116, 207,
    194, 247,  84,  41,   1,  71,  14,  49,  35,  95,  21, 169,  78,  96, 225, 215,
    243, 182,  92,  28, 118, 201,  74,   4, 128, 248,  11,  17, 132, 146,  48, 245,
     90, 149,  39, 120, 230,  87, 232, 106,  19, 175, 190, 126, 141, 202, 176, 137,
     27, 250,  40, 101, 227, 219,  20,  58, 178,  51, 216,  98,  22, 140, 121,  32,
    103,  61,  72, 203, 110,  29, 212,  85, 204, 180, 183, 150,  66,  15, 196, 172,
    197,  56,   0, 158,  45, 100,   7, 153, 222, 144, 167, 163, 135,  60, 231, 210,
```

```
  165, 174, 249,  38,  34, 224, 229, 220, 208, 217,  68, 241, 189, 206, 255, 125,
   54, 239,  89, 168, 122, 123, 145,  73, 234, 117,  99, 143, 200, 129,  82, 192,
  170, 104, 235, 136,  81,  93, 173, 205,  94, 236,  52, 105, 228,  46,   5, 198,
  254,  57, 155,  97, 133, 142, 171, 199,  50, 187, 181,  65, 107, 127, 226, 147,
  218, 184,  33, 131,  86,  77,  44,  31,  62,  88,  18, 238,  43,  24,  23, 154,
  159,  80, 111, 134, 114,   9,  91,   3, 130,  16,  10,  83, 240, 195, 119, 253
].freeze
TABLE_1 = [
   19,  11,  80, 114,  43,   1,  69,  94,  39,  18, 127, 117,  97,   3,  85,  43,
   27, 124,  70,  83,  47,  71,  63,  10,  47,  89,  79,   4,  14,  59,  11,   5,
   35, 107, 103,  68,  21,  86,  36,  91,  85, 126,  32,  50, 109,  94, 120,   6,
   53,  79,  28,  45,  99,  95,  41,  34,  88,  68,  93,  55, 110, 125, 105,  20,
   90,  80,  76,  96,  23,  60,  89,  64, 121,  56,  14,  74, 101,   8,  19,  78,
   76,  66, 104,  46, 111,  50,  32,   3,  39,   0,  58,  25,  92,  22,  18,  51,
   57,  65, 119, 116,  22, 109,   7,  86,  59,  93,  62, 110,  78,  99,  77,  67,
   12, 113,  87,  98, 102,   5,  88,  33,  38,  56,  23,   8,  75,  45,  13,  75,
   95,  63,  28,  49, 123, 120,  20, 112,  44,  30,  15,  98, 106,   2, 103,  29,
   82, 107,  42, 124,  24,  30,  41,  16, 108, 100, 117,  40,  73,  40,   7, 114,
   82, 115,  36, 112,  12, 102, 100,  84,  92,  48,  72,  97,   9,  54,  55,  74,
  113, 123,  17,  26,  53,  58,   4,   9,  69, 122,  21, 118,  42,  60,  27,  73,
  118, 125,  34,  15,  65, 115,  84,  64,  62,  81,  70,   1,  24, 111, 121,  83,
  104,  81,  49, 127,  48, 105,  31,  10,   6,  91,  87,  37,  16,  54, 116, 126,
   31,  38,  13,   0,  72, 106,  77,  61,  26,  67,  46,  29,  96,  37,  61,  52,
  101,  17,  44, 108,  71,  52,  66,  57,  33,  51,  25,  90,   2, 119, 122,  35
].freeze
TABLE_2 = [
   52,  50,  44,   6,  21,  49,  41,  59,  39,  51,  25,  32,  51,  47,  52,  43,
   37,   4,  40,  34,  61,  12,  28,   4,  58,  23,   8,  15,  12,  22,   9,  18,
   55,  10,  33,  35,  50,   1,  43,   3,  57,  13,  62,  14,   7,  42,  44,  59,
   62,  57,  27,   6,   8,  31,  26,  54,  41,  22,  45,  20,  39,   3,  16,  56,
   48,   2,  21,  28,  36,  42,  60,  33,  34,  18,   0,  11,  24,  10,  17,  61,
   29,  14,  45,  26,  55,  46,  11,  17,  54,  46,   9,  24,  30,  60,  32,   0,
   20,  38,   2,  30,  58,  35,   1,  16,  56,  40,  23,  48,  13,  19,  19,  27,
   31,  53,  47,  38,  63,  15,  49,   5,  37,  53,  25,  36,  63,  29,   5,   7
].freeze
TABLE_3 = [
    1,   5,  29,   6,  25,   1,  18,  23,  17,  19,   0,   9,  24,  25,   6,  31,
   28,  20,  24,  30,   4,  27,   3,  13,  15,  16,  14,  18,   4,   3,   8,   9,
   20,   0,  12,  26,  21,   8,  28,   2,  29,   2,  15,   7,  11,  22,  14,  10,
   17,  21,  12,  30,  26,  27,  16,  31,  11,   7,  13,  23,  10,   5,  22,  19
].freeze
TABLE_4 = [
   15,  12,  10,   4,   1,  14,  11,   7,   5,   0,  14,   7,   1,   2,  13,   8,
   10,   3,   4,   9,   6,   0,   3,   2,   5,   6,   8,   9,  11,  13,  15,  12
].freeze
TABLES = [TABLE_0, TABLE_1, TABLE_2, TABLE_3, TABLE_4].freeze

module_function

# x: Array de 32 entiers (octets), modifie en place
def compression_round(x, n, tbl)
  m = 4 - n
  mask = (32 << m) - 1
  (1 << n).times do |i|
    (1 << m).times do |j|
      a = j + i * (2 << m)
      b = a + (1 << m)
      y = (x[a] + (x[b] << 1)) & mask
      z = ((x[a] << 1) + x[b]) & mask
      x[a] = tbl[y]
      x[b] = tbl[z]
    end
  end
end

def compression(x)
```

```
    5.times { |n| compression_round(x, n, TABLES[n]) }
  end

  def bits_from_bytes(x)
    Array.new(128) { |i| (x[i >> 2] >> (3 - (i & 3))) & 1 }
  end

  def permutation(x, bits)
    16.upto(31) { |k| x[k] = 0 }
    128.times do |i|
      x[(i >> 3) + 16] |= bits[(i * 17) & 127] << (7 - (i & 7))
    end
  end

  # Retourne [sres (4 octets), kc (8 octets)], deux String binaires
  def v1(ki, rand)
    raise ArgumentError, 'ki/rand: 16 octets attendus' unless ki.bytesize == 16 && rand.bytesize == 16
    k = ki.bytes
    x = Array.new(32, 0)
    rand.bytes.each_with_index { |byte, i| x[16 + i] = byte }
    7.times do
      x[0, 16] = k
      compression(x)
      bits = bits_from_bytes(x)
      permutation(x, bits)
    end
    x[0, 16] = k
    compression(x)
    sres = Array.new(4) { |i| (x[2 * i] << 4) | x[2 * i + 1] }
    kc = Array.new(8, 0)
    (0...12).step(2) { |i| kc[i >> 1] = ((x[i + 18] << 6) | (x[i + 19] << 2) | (x[i + 20] >> 2)) & 0xFF }
    kc[6] = ((x[30] << 6) | (x[31] << 2)) & 0xFF
    kc[7] = 0
    [sres.pack('C*'), kc.pack('C*')]
  end
end
```

3.457 /opt/GSM/osmo-operator/tools/softsim/make_sim_op1.py

3903 octets, 97 lignes → 97 lignes

```
#!/usr/bin/env python3
# tools/softsim/make_sim_op1.py - genere le sim.xml de la SIM de l abonne du HLR
# a partir du sim.xml de demonstration de softSIM.
#
#   make_sim_op1.py <sim.xml de base> <sortie> [--imsi 001010001000001]
#
# Remplace l EF_IMSI (6f07) des deux DF par l IMSI donne, encode en BCD comme
# TS 51.011 (octet 1 : chiffre 1 + indicateur impair + type IMSI). Le Ki n est
# PAS ecrit ici : il vient de SOFTSIM_KI (lu par le serveur patche).
import re
import sys

DEMO_IMSI_BODY = "084906202771401469"  # IMSI de demonstration 460027217044196


def imsi_body(imsi):
    d = [int(c) for c in imsi]
    if len(d) != 15:
        raise SystemExit("IMSI a 15 chiffres attendu : %r" % imsi)
    out = [(d[0] << 4) | 0x09]  # chiffre 1, indicateur impair, type IMSI (1)
    for k in range(1, 15, 2):
        lo = d[k]
        hi = d[k + 1] if k + 1 < 15 else 0xF
```

```
        out.append((hi << 4) | lo)
    return "08" + "".join("%02x" % b for b in out)


def iccid_body(imsi, mcc, mnc):
    """ICCID de 20 chiffres (89 + MCC + MNC + fin de l IMSI + 00 + cle de Luhn),
    en BCD a quartets inverses comme TS 51.011 : le sim.xml de demonstration porte
    un ICCID aux quartets invalides (e, a) que le mobile affichait en > et :."""
    d = [int(c) for c in "89" + mcc + mnc + imsi[5:] + "00"]
    tot = 0
    for k, v in enumerate(reversed(d)):  # le chiffre de controle s ajoute apres
        if k % 2 == 0:
            v *= 2
            v = v - 9 if v > 9 else v
        tot += v
    d.append((10 - tot % 10) % 10)
    return "".join("%x%x" % (d[k + 1], d[k]) for k in range(0, 20, 2))


def plmn_bcd(mcc, mnc):
    """PLMN sur 3 octets (TS 31.102) : MCC1 MCC2 / MNC3 MCC3 / MNC2 MNC1."""
    m = [int(c) for c in mcc]
    n = [int(c) for c in mnc]
    n3 = n[2] if len(n) == 3 else 0xF
    return "%02x%02x%02x" % ((m[1] << 4) | m[0], (n3 << 4) | m[2], (n[1] << 4) | n[0])


def set_file_body(s, fid, fn):
    """Remplace le <body> du fichier fid (DF et tous les endroits) par fn(old_hex)."""
    pat = re.compile(r'(<file id="%s"[^>]*>.*?<body>)([0-9a-fA-F]*)(</body>)' % fid, re.S)
    return pat.sub(lambda m: m.group(1) + fn(m.group(2)) + m.group(3), s)


def set_plmn(s, mcc, mnc):
    """Remplace le PLMN des fichiers qui le portent (PLMNsel, FPLMN, LOCI, SPN)."""
    p = plmn_bcd(mcc, mnc)
    # EF_PLMNsel : premiere entree = PLMN, le reste en ff (meme longueur)
    s = set_file_body(s, "6f30", lambda b: p + "ff" * (len(b) // 2 - 3))
    # EF_FPLMN : aucune PLMN interdite
    s = set_file_body(s, "6f7b", lambda b: "ff" * (len(b) // 2))
    # EF_LOCI : LAI = PLMN (octets 4..6 du body, apres le TMSI)
    def loci(b):
        raw = bytearray.fromhex(b)
        raw[4:7] = bytes.fromhex(p)
        return raw.hex()
    s = set_file_body(s, "6f7e", loci)
    # EF_SPN : nom de l operateur, indicateur d affichage 00
    s = set_file_body(s, "6f46", lambda b: "00" + "Test".encode().hex() + "ff" * (len(b) // 2 - 5))
    return s


def main(argv):
    if len(argv) < 3:
        raise SystemExit(__doc__)
    src, dst = argv[1], argv[2]
    imsi = argv[argv.index("--imsi") + 1] if "--imsi" in argv else "001010001000001"
    # verification : l encodeur reproduit l IMSI de demonstration
    assert imsi_body("460027217044196") == DEMO_IMSI_BODY
    body = imsi_body(imsi)
    s = open(src, encoding="utf-8").read()
    s, n = re.subn(r"<body>%s</body>" % DEMO_IMSI_BODY, "<body>%s</body>" % body, s)
    if n == 0:
        raise SystemExit("EF_IMSI de demonstration introuvable dans %s" % src)
    mcc = imsi[:3]
    mnc = imsi[3:5]  # MNC a 2 chiffres, comme l IMSI du plan
    s = set_plmn(s, mcc, mnc)
```

```
    # EF_ICCID (2fe2) : 10 octets, valide en BCD
    s = set_file_body(s, "2fe2", lambda b: iccid_body(imsi, mcc, mnc))
    open(dst, "w", encoding="utf-8").write(s)
    print("EF_IMSI %s -> %s (%d occurrence(s))" % (imsi, body, n))


if __name__ == "__main__":
    main(sys.argv)
```

3.458 /opt/GSM/osmo-operator/tools/softsim/ruby_compat.rb

446 octets, 11 lignes → 11 lignes

```
# tools/softsim/ruby_compat.rb - shim pour softSIM (gitea.osmocom.org/sim-card/softsim)
# sous Ruby >= 3.2 : File.exists? et Fixnum/Bignum ont ete supprimes. Charge avec
#   ruby -r/usr/local/share/softsim/ruby_compat.rb demo_server.rb ...
# Ne modifie pas le depot softSIM.
class File
  def self.exists?(p) = exist?(p)
end
Fixnum = Integer unless defined?(Fixnum)
Bignum = Integer unless defined?(Bignum)
$stdout.sync = true
$stderr.sync = true
```

3.459 /opt/GSM/osmo-operator/web/app.js

15332 octets, 307 lignes → 307 lignes

```
// Reglages Osmocom - web/index.html, servi par tools/osmo-reglages.py (API /api/...)
'use strict';

const $ = s => document.querySelector(s);
const esc = s => String(s).replace(/[&<>"]/g, c => ({ '&': '&amp;', '<': '&lt;', '>': '&gt;', '"': '&quot;' }[c]));

function msg(t, ok = true) {
  const m = $('#msg');
  m.textContent = t;
  m.className = ok ? 'ok' : 'err';
  clearTimeout(msg.t);
  msg.t = setTimeout(() => (m.textContent = ''), 6000);
}

async function api(u, corps) {
  const r = await fetch(u, corps === undefined ? {} :
    { method: 'POST', headers: { 'Content-Type': 'application/json' }, body: JSON.stringify(corps) });
  const j = await r.json();
  if (!r.ok) throw new Error(j.err || r.status);
  return j;
}

/* ── onglets ── */
document.querySelectorAll('nav button').forEach(b => b.onclick = () => {
  document.querySelectorAll('nav button, .onglet').forEach(x => x.classList.remove('actif'));
  b.classList.add('actif');
  $('#' + b.dataset.o).classList.add('actif');
  try { localStorage.setItem('onglet', b.dataset.o); } catch (e) { }
  if (b.dataset.o === 'etat') chargerEtat();
});

/* ── formulaires globals.conf / instance ──
 * Un champ = {cle, val, section, doc, defaut, choix[]}. Vide = defaut du script (grise). Quand les scripts
```

```
 * comparent la variable a des valeurs fixes (choix), un menu deroulant : « (defaut) », les choix, et la
 * valeur actuelle si elle n'en fait pas partie. "autre…" rend la saisie libre. */
function dessinerForm(champs, conteneur, filtre, posees) {
  const f = filtre.trim().toLowerCase();
  const parSec = new Map();
  for (const c of champs) {
    if (f && !(c.cle + ' ' + c.doc + ' ' + c.section).toLowerCase().includes(f)) continue;
    if (posees && c.val === '') continue;
    if (!parSec.has(c.section)) parSec.set(c.section, []);
    parSec.get(c.section).push(c);
  }
  const champ = c => {
    const id = conteneur.id + '_' + c.cle;
    let ctrl;
    if (c.choix && c.choix.length) {
      const opts = ['<option value="">(défaut' + (c.defaut ? ' : ' + esc(c.defaut) : '') + ')</option>'];
      const vals = c.choix.slice();
      if (c.val !== '' && !vals.includes(c.val)) vals.unshift(c.val);
      for (const v of vals) opts.push(`<option value="${esc(v)}"${v === c.val ? ' selected' :
       ↪  ''}>${esc(v)}</option>`);
      opts.push('<option value="\u0000autre">autre…</option>');
      ctrl = `<select id="${id}" data-cle="${c.cle}" data-orig="${esc(c.val)}">${opts.join('')}</select>`;
    } else {
      ctrl = `<input id="${id}" data-cle="${c.cle}" data-orig="${esc(c.val)}" value="${esc(c.val)}"
↪  placeholder="${esc(c.defaut || '')}">`;
    }
    return `<div class="champ${c.val !== '' ? ' pose' : ''}"><label
     ↪  for="${id}">${c.cle}<small>${esc(c.doc)}</small></label>${ctrl}</div>`;
  };
  conteneur.innerHTML = [...parSec].map(([s, l]) =>
    `<div class="carte"><h2>${esc(s || 'Divers')} <small>(${l.length})</small></h2><div
     ↪  class="grille">${l.map(champ).join('')}</div></div>`
  ).join('') || '<p class="aide">Aucun réglage ne correspond.</p>';
  conteneur.querySelectorAll('input, select').forEach(x => {
    const maj = () => x.classList.toggle('modifie', x.value !== x.dataset.orig);
    x.oninput = maj;
    x.onchange = () => {
      if (x.tagName === 'SELECT' && x.value === '\u0000autre') {
        // bascule en saisie libre, meme id/cle
        const i = document.createElement('input');
        i.id = x.id; i.dataset.cle = x.dataset.cle; i.dataset.orig = x.dataset.orig; i.value = x.dataset.orig;
        i.oninput = () => i.classList.toggle('modifie', i.value !== i.dataset.orig);
        x.replaceWith(i); i.focus();
        return;
      }
      maj();
    };
  });
}
const modifies = conteneur => {
  const v = {};
  conteneur.querySelectorAll('input.modifie, select.modifie').forEach(x => (v[x.dataset.cle] = x.value));
  return v;
};

/* ── globals.conf ── */
let champs = [];
async function chargerGlobals() { champs = await api('/api/globals'); dessinerGlobals(); }
const dessinerGlobals = () => dessinerForm(champs, $('#g-form'), $('#g-filtre').value, $('#g-modifiees').checked);
$('#g-filtre').oninput = dessinerGlobals;
$('#g-modifiees').onchange = dessinerGlobals;
$('#g-enr').onclick = async () => {
  const v = modifies($('#g-form'));
  if (!Object.keys(v).length) return msg('Rien de modifié');
  try { await api('/api/globals', v); msg('globals.conf réécrit : ' + Object.keys(v).join(', '));
   ↪  chargerGlobals(); }
```

```
  catch (e) { msg(e.message, false); }
};

/* ── instances Calypso (instances/instance-N.env) ── */
let champsInst = [];
async function chargerInstances() {
  const l = await api('/api/instances');
  $('#i-sel').innerHTML = l.map(i => `<option value="${i.n}">${esc(i.role)}${i.existe ? '' : ' — (pas encore de
   ↪  fichier)'}</option>`).join('');
  try { const n = localStorage.getItem('instance'); if (n) $('#i-sel').value = n; } catch (e) { }
  chargerInstance();
}
async function chargerInstance() {
  champsInst = await api('/api/instance?n=' + $('#i-sel').value);
  try { localStorage.setItem('instance', $('#i-sel').value); } catch (e) { }
  dessinerInstance();
}
const dessinerInstance = () => dessinerForm(champsInst, $('#i-form'), $('#i-filtre').value,
 ↪  $('#i-modifiees').checked);
$('#i-sel').onchange = chargerInstance;
$('#i-filtre').oninput = dessinerInstance;
$('#i-modifiees').onchange = dessinerInstance;
$('#i-enr').onclick = async () => {
  const v = modifies($('#i-form'));
  if (!Object.keys(v).length) return msg('Rien de modifié');
  try {
    const r = await api('/api/instance', { n: +$('#i-sel').value, valeurs: v });
    msg(r.fichier + ' écrit : ' + Object.keys(v).join(', '));
    chargerInstances();
  } catch (e) { msg(e.message, false); }
};

/* ── demons (VTY) ── */
let demons = [], cour = null;

async function chargerDemons(force) {
  demons = await api('/api/demons' + (force ? '?rafraichir=1' : ''));
  const avant = $('#d-sel').value;
  $('#d-sel').innerHTML = demons.map((d, i) =>
    `<option value="${i}">${esc(d.nom)} — ${d.hote}:${d.port} (${esc(d.invite)})</option>`).join('');
  if (avant && demons[avant]) $('#d-sel').value = avant;
  if (demons.length) chargerCfg();
}
const demon = () => demons[$('#d-sel').value];

async function chargerCfg() {
  const d = demon();
  if (!d) return;
  $('#d-cfg').textContent = 'lecture…';
  try { cour = await api(`/api/config?hote=${d.hote}&port=${d.port}`); dessinerCfg(); }
  catch (e) { $('#d-cfg').textContent = e.message; }
}

function dessinerCfg() {
  const f = $('#filtre').value.trim().toLowerCase();
  $('#d-cfg').innerHTML = cour.lignes.map((l, i) => {
    if (f && !(l.txt + ' ' + l.chemin.join(' ')).toLowerCase().includes(f)) return '';
    return `<div class="l${l.noeud ? ' noeud' : ''}" style="padding-left:${l.prof * 14}px">
      <input class="t" data-i="${i}" value="${esc(l.txt)}">
      <button class="b2" data-a="ok" data-i="${i}" title="appliquer">✓</button>
      ${l.noeud ? `<button class="b2" data-a="aj" data-i="${i}" title="ajouter une ligne dans ce
 ↪  noeud">+</button>` : ''}
      <button class="b2" data-a="no" data-i="${i}" title="envoyer « no … »">no</button></div>`;
  }).join('');
  $('#d-cfg').querySelectorAll('input.t').forEach(x => {
    x.oninput = () => x.classList.toggle('modifie', x.value !== cour.lignes[x.dataset.i].txt);
```

```
    x.onkeydown = e => {
      if (e.key === 'Enter') envoyer(+x.dataset.i, [x.value]);
      if (e.key === 'Escape') { x.value = cour.lignes[x.dataset.i].txt; x.classList.remove('modifie'); }
    };
  });
  $('#d-cfg').querySelectorAll('button').forEach(b => b.onclick = () => {
    const i = +b.dataset.i, l = cour.lignes[i], x = $(`#d-cfg input[data-i="${i}"]`);
    if (b.dataset.a === 'ok') envoyer(i, [x.value]);
    if (b.dataset.a === 'no' && confirm('Envoyer « no ' + l.txt + ' » ?')) envoyer(i, ['no ' + l.txt]);
    if (b.dataset.a === 'aj') { const n = prompt('Ligne à ajouter dans « ' + l.txt + ' » :'); if (n) envoyer(i,
     ↪  [n], true); }
  });
}

async function envoyer(i, lignes, dedans) {
  const d = demon(), l = cour.lignes[i], chemin = dedans ? [...l.chemin, l.txt] : l.chemin;
  try {
    const r = await api('/api/appliquer', { hote: d.hote, port: d.port, chemin, lignes });
    $('#d-jrn').textContent = `[${new Date().toLocaleTimeString()}] ${d.nom}
     ↪  ${d.hote}:${d.port}\n${r.journal}\n\n` + $('#d-jrn').textContent;
    msg(r.ok ? 'Appliqué' : 'Erreur VTY — voir le journal', r.ok);
    if (r.ok) chargerCfg();
  } catch (e) { msg(e.message, false); }
}

async function consoleVty(lignes) {
  const d = demon();
  if (!d) return;
  $('#c-out').textContent = '…';
  try { $('#c-out').textContent = (await api('/api/console', { hote: d.hote, port: d.port, lignes })).sortie; }
  catch (e) { $('#c-out').textContent = e.message; }
}

$('#d-sel').onchange = chargerCfg;
$('#d-ref').onclick = chargerCfg;
$('#d-scan').onclick = () => chargerDemons(true);
$('#filtre').oninput = () => cour && dessinerCfg();
$('#c-go').onclick = () => consoleVty($('#c-in').value.split('\n'));
$('#c-in').onkeydown = e => { if (e.key === 'Enter' && (e.ctrlKey || e.metaKey)) $('#c-go').click(); };
document.querySelectorAll('[data-c]').forEach(b => b.onclick = () => consoleVty([b.dataset.c]));
$('#d-write').onclick = () => {
  const d = demon();
  if (d && confirm('write memory sur ' + d.nom + ' ? (réécrit son fichier de config)')) consoleVty(['write
   ↪  memory']);
};

/* ── fichiers ── */
async function chargerFichiers() {
  const l = await api('/api/fichiers');
  $('#f-sel').innerHTML = l.map(p => `<option>${esc(p)}</option>`).join('');
  chargerFichier();
}
async function chargerFichier() {
  const t = $('#f-txt');
  t.value = (await api('/api/fichier?p=' + encodeURIComponent($('#f-sel').value))).texte;
  t.dataset.orig = t.value;
  t.classList.remove('modifie');
}
$('#f-sel').onchange = chargerFichier;
$('#f-txt').oninput = e => e.target.classList.toggle('modifie', e.target.value !== e.target.dataset.orig);
$('#f-enr').onclick = async () => {
  try { await api('/api/fichier', { p: $('#f-sel').value, texte: $('#f-txt').value }); msg('Enregistré');
   ↪  chargerFichier(); }
  catch (e) { msg(e.message, false); }
};
```

```
/* ── configs osmo : /etc/osmocom/osmo-*.cfg, fichier actif ou gabarit configs/ ── */
const osmo = { liste: [], courant: null, vue: 'etc' };
function dessinerOsmo() {
  const f = $('#o-filtre').value.trim().toLowerCase();
  $('#o-liste').innerHTML = osmo.liste.filter(c => !f || c.nom.includes(f)).map(c =>
    `<li class="${osmo.courant && osmo.courant.nom === c.nom ? 'actif' : ''}" data-n="${esc(c.nom)}">` +
    `<span class="pt ${c.pid ? 'on' : ''}" title="${c.pid ? 'démon en marche (pid ' + c.pid + ')' : 'aucun démon
     ↪  ne lit ce fichier'}"></span>` +
    `${esc(c.nom)}${c.protege ? ' <span title="protégé : start.sh ne le réécrit pas">🔒</span>' : ''}` +
    `${c.gabarit ? '' : ' <small>sans gabarit</small>'}</li>`).join('');
  document.querySelectorAll('#o-liste li').forEach(li => li.onclick = () => ouvrirOsmo(li.dataset.n));
}
async function chargerOsmo() {
  osmo.liste = await api('/api/configs-osmo');
  dessinerOsmo();
  if (!osmo.courant && osmo.liste.length) {
    let n = null;
    try { n = localStorage.getItem('osmo'); } catch (e) { }
    ouvrirOsmo(osmo.liste.some(c => c.nom === n) ? n : osmo.liste[0].nom);
  }
}
async function ouvrirOsmo(nom, vue) {
  const t = $('#o-txt');
  if (t.classList.contains('modifie') && !confirm('Modifications non enregistrées : les abandonner ?')) return;
  osmo.courant = osmo.liste.find(c => c.nom === nom);
  osmo.vue = vue || (osmo.vue === 'gabarit' && osmo.courant.gabarit ? 'gabarit' : 'etc');
  try { localStorage.setItem('osmo', nom); } catch (e) { }
  const c = osmo.courant, p = osmo.vue === 'gabarit' ? c.gabarit : c.etc;
  $('#o-titre').textContent = c.nom + '.cfg';
  $('#o-v-etc').classList.toggle('actif', osmo.vue === 'etc');
  $('#o-v-gab').classList.toggle('actif', osmo.vue === 'gabarit');
  $('#o-v-gab').disabled = !c.gabarit;
  $('#o-prot').textContent = c.protege ? '🔒 Protégé' : '🔓 Non protégé';
  $('#o-prot').classList.toggle('actif', c.protege);
  $('#o-aide').textContent = osmo.vue === 'gabarit'
    ? c.gabarit + (c.protege ? ' — ' + c.etc + ' est protégé : ce gabarit ne s\'applique plus tant que la
     ↪  protection reste.'
                             : ' — start.sh en régénère ' + c.etc + ' (marqueurs __X__ remplacés).')
    : c.etc + (c.protege ? ' — protégé : start.sh ne le réécrit pas. ' : c.gabarit ? ' — réécrit depuis le gabarit
     ↪  au prochain run ; Enregistrer le protège. ' : ' — ') +
      (c.pid ? 'Lu par le démon pid ' + c.pid + ' : effet à son redémarrage.' : 'Aucun démon ne le lit en ce
↪  moment.');
  t.value = (await api('/api/fichier?p=' + encodeURIComponent(p))).texte;
  t.dataset.orig = t.value;
  t.dataset.p = p;
  t.classList.remove('modifie');
  dessinerOsmo();
}
$('#o-filtre').oninput = dessinerOsmo;
$('#o-txt').oninput = e => e.target.classList.toggle('modifie', e.target.value !== e.target.dataset.orig);
$('#o-v-etc').onclick = () => osmo.courant && ouvrirOsmo(osmo.courant.nom, 'etc');
$('#o-v-gab').onclick = () => osmo.courant && ouvrirOsmo(osmo.courant.nom, 'gabarit');
$('#o-rech').onclick = () => { $('#o-txt').classList.remove('modifie'); chargerOsmo().then(() => osmo.courant &&
↪  ouvrirOsmo(osmo.courant.nom, osmo.vue)); };
$('#o-enr').onclick = async () => {
  const t = $('#o-txt');
  try {
    const r = await api('/api/fichier', { p: t.dataset.p, texte: t.value });
    msg(t.dataset.p + ' enregistré (.bak horodaté)' + (r.protege ? ', protégé contre la régénération' : ''));
    t.classList.remove('modifie');
    await chargerOsmo();
    ouvrirOsmo(osmo.courant.nom, osmo.vue);
  } catch (e) { msg(e.message, false); }
};
$('#o-prot').onclick = async () => {
  const c = osmo.courant;
```

```
  if (!c) return;
  if (c.protege && !confirm(c.nom + '.cfg redeviendra réécrit depuis le gabarit au prochain run. Continuer ?'))
   ↪  return;
  try {
    await api('/api/protege', { nom: c.nom + '.cfg', on: !c.protege });
    msg(c.nom + '.cfg ' + (c.protege ? 'rendu au gabarit' : 'protégé'));
    await chargerOsmo();
    ouvrirOsmo(c.nom, osmo.vue);
  } catch (e) { msg(e.message, false); }
};

/* ── etat ── */
const duree = s => s > 3600 ? (s / 3600 | 0) + 'h' + String((s % 3600) / 60 | 0).padStart(2, '0') : s > 60 ? (s /
 ↪  60 | 0) + 'm' : s + 's';
async function chargerEtat() {
  const e = await api('/api/etat');
  $('#e-ps').innerHTML = '<tr><th>PID</th><th>Âge</th><th>RSS Mo</th><th>CPU %</th><th>Commande</th></tr>' +
    e.procs.map(p => `<tr><td>${p.pid}</td><td>${duree(p.age)}</td><td>${p.rss}</td><td>${p.cpu}</td><td><code>
 ↪  ${esc(p.cmd)}</code></td></tr>`).join('');
  $('#e-vty').innerHTML = '<tr><th>Démon</th><th>Adresse</th><th>Invite</th><th>PID</th></tr>' +
    e.vty.map(d => `<tr><td>${esc(d.nom)}</td><td>${d.hote}:${d.port}</td><td>${esc(d.invite)}</td><td>${d.pid}
 ↪  </td></tr>`).join('');
}

/* ── demarrage ── */
chargerGlobals();
chargerInstances().catch(e => msg(e.message, false));
chargerDemons().catch(e => msg(e.message, false));
chargerFichiers();
chargerOsmo().catch(e => msg(e.message, false));
try { const o = localStorage.getItem('onglet'); if (o) document.querySelector(`nav
 ↪  button[data-o="${o}"]`)?.click(); } catch (e) { }
```

3.460 /opt/GSM/osmo-operator/web/index.html

5148 octets, 119 lignes → 119 lignes

```
<!doctype html>
<html lang="fr">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width,initial-scale=1">
<title>Réglages Osmocom</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
<header>
  <h1>Réglages Osmocom</h1>
  <nav>
    <button data-o="reseau" class="actif">globals.conf</button>
    <button data-o="osmo">Configs osmo</button>
    <button data-o="instances">Instances Calypso</button>
    <button data-o="demons">Démons (VTY)</button>
    <button data-o="fichiers">Fichiers</button>
    <button data-o="etat">État</button>
  </nav>
  <span id="msg"></span>
</header>

<main>
  <!-- globals.conf (reseau + lanceur), sur le modele de globals.conf.example -->
  <section id="reseau" class="onglet actif">
    <div class="barre">
      <button class="b" id="g-enr">Enregistrer globals.conf</button>
```

```
    <input id="g-filtre" placeholder="filtrer (ex. ARFCN, SAP, WAN)">
    <label class="case"><input type="checkbox" id="g-modifiees"> seulement les valeurs posées</label>
    <span class="aide">Vide = défaut du script (grisé). Pris en compte au prochain run ; sauvegarde .bak
     ↪  horodatée.</span>
  </div>
  <div id="g-form"></div>
</section>

<!-- chaque /etc/osmocom/osmo-*.cfg : menu a gauche, fichier actif ou gabarit configs/ a droite -->
<section id="osmo" class="onglet">
  <div class="osmo">
    <div class="carte o-menu">
      <h2>/etc/osmocom</h2>
      <input id="o-filtre" placeholder="filtrer (bsc, bts, pcu…)">
      <ul id="o-liste"></ul>
    </div>
    <div class="carte o-edit">
      <div class="barre serre">
        <strong id="o-titre">—</strong>
        <span class="bascule">
          <button class="b2" id="o-v-etc" data-v="etc">Fichier actif</button>
          <button class="b2" id="o-v-gab" data-v="gabarit">Gabarit configs/</button>
        </span>
        <button class="b" id="o-enr">Enregistrer</button>
        <button class="b2" id="o-prot" title="un fichier protégé n'est plus réécrit par start.sh depuis
         ↪  configs/"> Protégé</button>
        <button class="b2" id="o-rech">↻ Recharger</button>
      </div>
      <p class="aide" id="o-aide"></p>
      <textarea id="o-txt" rows="34" spellcheck="false"></textarea>
    </div>
  </div>
</section>

<!-- un fichier par instance Calypso (c54x_exe/run.sh) : instances/instance-N.env -->
<section id="instances" class="onglet">
  <div class="barre">
    <select id="i-sel"></select>
    <button class="b" id="i-enr">Enregistrer l'instance</button>
    <input id="i-filtre" placeholder="filtrer (ex. CALYPSO_BSP, INSNS, AIR)">
    <label class="case"><input type="checkbox" id="i-modifiees"> seulement les valeurs posées</label>
    <span class="aide">Réglages du banc DSP de <em>cette</em> instance (pas globaux) : la ligne de commande
     ↪  garde la priorité, puis ce fichier, puis le défaut de run.sh.</span>
  </div>
  <div id="i-form"></div>
</section>

<!-- running-config de chaque demon, par la VTY -->
<section id="demons" class="onglet">
  <div class="barre">
    <select id="d-sel"></select>
    <button class="b2" id="d-ref">↻ Recharger</button>
    <button class="b2" id="d-scan">Rechercher les démons</button>
    <input id="filtre" placeholder="filtrer (ex. arfcn, bts 0)">
    <button class="b2" id="d-write" title="écrit le fichier de config du démon">write memory</button>
  </div>
  <div class="flex2">
    <div class="carte"><h2>running-config — clic pour modifier, Entrée pour appliquer</h2><div id="d-cfg"
     ↪  class="cfg"></div></div>
    <div>
      <div class="carte">
        <h2>Console VTY</h2>
        <textarea id="c-in" rows="5" placeholder="show bts 0&#10;configure terminal&#10; network&#10;
         ↪  ..."></textarea>
        <div class="barre serre">
          <button class="b" id="c-go">Exécuter</button>
```

```
            <button class="b2" data-c="show running-config">running-config</button>
            <button class="b2" data-c="show rate-counters">rate-counters</button>
            <button class="b2" data-c="show bts 0">bts 0</button>
            <button class="b2" data-c="show subscriber cache">abonnés</button>
            <button class="b2" data-c="show talloc-context application brief">talloc</button>
          </div>
          <pre id="c-out"></pre>
        </div>
        <div class="carte"><h2>Journal des modifications</h2><pre id="d-jrn"></pre></div>
      </div>
    </div>
  </section>

  <!-- edition brute des fichiers de config -->
  <section id="fichiers" class="onglet">
    <div class="barre">
      <select id="f-sel"></select>
      <button class="b" id="f-enr">Enregistrer</button>
      <span class="aide">Sauvegarde .bak horodatée ; effet au prochain démarrage du démon.</span>
    </div>
    <textarea id="f-txt" rows="34" spellcheck="false"></textarea>
  </section>

  <section id="etat" class="onglet">
    <div class="carte"><h2>Processus</h2><table id="e-ps"></table></div>
    <div class="carte"><h2>VTY détectées</h2><table id="e-vty"></table></div>
  </section>
</main>
<script src="app.js"></script>
</body>
</html>
```

3.461 /opt/GSM/osmo-operator/web/style.css

5048 octets, 81 lignes → 81 lignes

```
/* Reglages Osmocom - web/index.html */
:root {
  --fond: #f6f7f9; --carte: #fff; --texte: #1d2330; --doux: #667085; --bord: #d9dde5;
  --accent: #2f6fde; --ok: #1f8a4c; --err: #c0362c; --modif: #fff4d6; --pose: #e8f1ff; --code: #f0f2f6;
}
@media (prefers-color-scheme: dark) {
  :root {
    --fond: #14171c; --carte: #1c2027; --texte: #e3e6eb; --doux: #8d96a5; --bord: #2e3440;
    --accent: #6b9cff; --ok: #4cc27e; --err: #ff6b5e; --modif: #3a3220; --pose: #1d2a40; --code: #232833;
  }
}
* { box-sizing: border-box; }
body { margin: 0; background: var(--fond); color: var(--texte); font: 14px/1.45 system-ui, sans-serif; }

header {
  display: flex; gap: 4px; align-items: center; flex-wrap: wrap; padding: 10px 16px;
  border-bottom: 1px solid var(--bord); background: var(--carte); position: sticky; top: 0; z-index: 2;
}
header h1 { font-size: 15px; margin: 0 16px 0 0; }
nav button { background: none; border: 0; padding: 6px 12px; border-radius: 6px; color: var(--doux); cursor:
 ↪  pointer; font: inherit; }
nav button.actif { background: var(--code); color: var(--texte); font-weight: 600; }
#msg { margin-left: auto; font-size: 13px; }
.ok { color: var(--ok); } .err { color: var(--err); }

main { padding: 16px; max-width: 1300px; margin: auto; }
.onglet { display: none; } .onglet.actif { display: block; }
```

```
.carte { background: var(--carte); border: 1px solid var(--bord); border-radius: 8px; padding: 14px;
 ↪  margin-bottom: 14px; }
h2 { font-size: 13px; text-transform: uppercase; letter-spacing: .04em; color: var(--doux); margin: 0 0 10px; }
.aide { color: var(--doux); font-size: 13px; }

.barre { display: flex; gap: 8px; align-items: center; flex-wrap: wrap; margin-bottom: 12px; }
.barre.serre { margin: 8px 0 0; }
.barre select { width: auto; min-width: 240px; }
#g-filtre, #filtre { max-width: 300px; }
.case { display: inline-flex; gap: 6px; align-items: center; font: 13px system-ui; color: var(--doux); }
.case input { width: auto; }

/* formulaire globals.conf */
.grille { display: grid; grid-template-columns: repeat(auto-fill, minmax(300px, 1fr)); gap: 10px 18px; }
.champ label { display: block; font: 600 12.5px ui-monospace, monospace; }
.champ label small { display: block; font: 12px system-ui; color: var(--doux); font-weight: 400; }
.champ.pose input { background: var(--pose); }
input, select, textarea {
  font: 13px ui-monospace, monospace; background: var(--fond); color: var(--texte);
  border: 1px solid var(--bord); border-radius: 5px; padding: 5px 7px; width: 100%;
}
input::placeholder { color: var(--doux); opacity: .7; }
input.modifie, textarea.modifie { background: var(--modif) !important; }

button.b { background: var(--accent); color: #fff; border: 0; border-radius: 6px; padding: 6px 12px; cursor:
 ↪  pointer; font: inherit; }
button.b2 { background: var(--code); color: var(--texte); border: 1px solid var(--bord); border-radius: 6px;
 ↪  padding: 4px 10px; cursor: pointer; font: inherit; }

/* running-config */
.cfg { font: 13px ui-monospace, monospace; }
.cfg .l { display: flex; align-items: center; gap: 6px; padding: 1px 0; }
.cfg .l .t { flex: 1; padding: 2px 6px; border: 1px solid transparent; background: none; width: auto; }
.cfg .l .t:hover { border-color: var(--bord); }
.cfg .l.noeud > .t { font-weight: 700; color: var(--accent); }
.cfg .l button { visibility: hidden; }
.cfg .l:hover button, .cfg .l .t.modifie ~ button { visibility: visible; }

pre { background: var(--code); padding: 10px; border-radius: 6px; overflow: auto; max-height: 420px; margin: 8px 0
 ↪  0; white-space: pre-wrap; }
table { border-collapse: collapse; width: 100%; font-size: 13px; }
td, th { border-bottom: 1px solid var(--bord); padding: 4px 8px; text-align: left; vertical-align: top; }
td code { font-size: 12px; word-break: break-all; }
.flex2 { display: grid; grid-template-columns: 1fr 1fr; gap: 14px; }
@media (max-width: 800px) { .flex2 { grid-template-columns: 1fr; } main { padding: 16px 12px; } }
/* [2026-10-22] onglet Configs osmo : menu des /etc/osmocom/osmo-*.cfg a gauche, editeur a droite */
.osmo { display: grid; grid-template-columns: 240px 1fr; gap: 14px; align-items: start; }
.o-menu { position: sticky; top: 64px; }
.o-menu ul { list-style: none; margin: 8px 0 0; padding: 0; max-height: 70vh; overflow: auto; }
.o-menu li { padding: 5px 8px; border-radius: 5px; cursor: pointer; font: 13px ui-monospace, monospace; display:
 ↪  flex; gap: 8px; align-items: center; }
.o-menu li:hover { background: var(--code); }
.o-menu li.actif { background: var(--pose); font-weight: 600; }
.o-menu li small { color: var(--doux); font: 11px system-ui; }
.pt { width: 8px; height: 8px; border-radius: 50%; background: var(--bord); flex: none; }
.pt.on { background: var(--ok); }
.bascule button.actif { background: var(--accent); color: #fff; border-color: var(--accent); }
.bascule button:disabled { opacity: .45; cursor: default; }
@media (max-width: 720px) { .osmo { grid-template-columns: 1fr; } .o-menu { position: static; } }
#o-prot.actif { background: var(--ok); color: #fff; border-color: var(--ok); }
```

3.462 /opt/GSM/osmo-operator/wiki/installer-osmo-jusquau-dsp.ipynb

50257 octets, 1132 lignes → 1132 lignes

```
{
 "cells": [
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# Native install on Ubuntu 24.04, up to a call on the Calypso DSP\n",
    "\n",
    "This notebook builds the whole bench on a **fresh Ubuntu 24.04** (desktop or server, x86_64), without\n",
    "Docker. It ends with a verified call in DSP mode (`c54x_exe` + TI mask ROM). It replays the build steps of\n",
    "the `Dockerfile` (stages `base`, `osmo-core`, `bb`, `qemu`, `l1`, and the runtime setup at the end of\n",
    "`osmocom-nitb`), with `sudo make install` in place of the `.deb` cache (`osmo-deb pack`): same result in\n",
    "`/usr/local`, without the package.\n",
    "\n",
    "**Why a notebook and not `install.sh`?** `sudo ./install.sh` installs the apt dependencies, osmocom-bb,\n",
    "gapk and the configs, but **not** the Osmocom core (osmo-stp, msc, hlr, bsc, bts…), qosmo, c54x_exe or the\n",
    "firmware: its last step fails with \"demons absents\" on a stock 24.04. This notebook reuses `install.sh`\n",
    "for the parts it does well (the apt list, extracted from the `Dockerfile`, and the service account) and\n",
    "builds the rest.\n",
    "\n",
    "Rules: **one step = one cell**, followed by its **check** cell. Cells are idempotent, so you can re-run\n",
    "them. `%%bash` cells do not share their environment: each one sources `/opt/GSM/.notebook.env`.\n",
    "\n",
    "| | |\n",
    "|---|---|\n",
    "| OS | Ubuntu 24.04 LTS (noble), x86_64 |\n",
    "| Disk | ~12 GB free under `/opt` |\n",
    "| Time | ~1 h on 8 cores (core 20–30 min, QEMU 10–15 min) |\n",
    "| Network | gitea.osmocom.org, github.com, ftp.freecalypso.org, archive.ubuntu.com |\n",
    "\n",
    "### Before opening the notebook\n",
    "\n",
    "Ubuntu 24.04 refuses `pip install` outside a venv (PEP 668). Install Jupyter in a venv, and **unlock sudo\n",
    "in the same terminal** before starting it: a notebook cannot answer a sudo password prompt, so a cell that\n",
    "asks for one would hang.\n",
    "\n",
    "```bash\n",
    "sudo apt-get install -y python3-venv git\n",
    "python3 -m venv ~/.venv-jupyter\n",
    "~/.venv-jupyter/bin/pip install jupyterlab\n",
    "sudo -v                      # caches your sudo credentials (15 min by default)\n",
    "~/.venv-jupyter/bin/jupyter lab\n",
    "```\n",
    "\n",
    "For a long run, give your user a temporary `NOPASSWD` rule (`sudo visudo -f /etc/sudoers.d/osmo-nb`) or\n",
    "run the kernel as root. The cells work in both cases.\n",
    "\n",
    "### Recommended: a disposable container\n",
    "\n",
    "Every `make install` writes into `/usr/local`, and the Osmocom libraries installed there shadow any other\n",
    "version on the machine. On a machine that is not a **fresh** Ubuntu 24.04 (an earlier install, other\n",
    "Osmocom builds, packages already half there), prefer a throw-away container: nothing leaks onto the host,\n",
    "and a failed run is deleted in one command.\n",
    "\n",
    "```bash\n",
    "docker run -it --name nb-ubuntu --privileged --shm-size=2g -p 8888:8888 ubuntu:24.04 bash\n",
    "```\n",
    "\n",
    "The full procedure (in the container, stop and resume, testing local changes, unattended run) is in\n",
    "[README-notebook](README-notebook.md).\n",
    "\n",
    "`--privileged` gives the stack what it needs at run time: `/dev/net/tun` (the APN interface) and\n",
    "`NET_ADMIN` (the addressing plan). Step 0 warns when the system does not look fresh.\n",
    "\n",
```

```
   "> [2026-10-20] The same installation, generated step by step from the Dockerfile (every `RUN` as a bash
↪  block): [doc/install-from-scratch.md](../doc/install-from-scratch.md) (French, with the Dockerfile's
↪  rationale) and [doc/install-from-scratch.en.md](../doc/install-from-scratch.en.md) (English notes per stage).
↪  Regenerate with `tools/dockerfile-to-install-md.sh`.\n",
   "\n",
   "> [2026-10-23] The launcher is now `./start.sh`. `start-native.sh` is kept as a symlink to it, so the old\n",
   "> name and every command written with it still work.\n"
  ],
  "id": "c00"
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "## 0. Root, environment, sudo\n",
   "\n",
   "`Dockerfile` stage `base`: `ARG ROOT=/opt/GSM`, `PKG_CONFIG_PATH=/usr/local/lib/pkgconfig`,\n",
   "`LD_LIBRARY_PATH=/usr/local/lib`, `GIT_TERMINAL_PROMPT=0`. They go into a file that every later cell\n",
   "sources, together with `SUDO` (empty when the kernel already runs as root)."
  ],
  "id": "c01"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": [
   "%%bash\n",
   "set -e\n",
   ". /etc/os-release; echo \"$PRETTY_NAME\"\n",
   "[ \"${ID:-}\" = ubuntu ] && [ \"${VERSION_ID:-}\" = \"24.04\" ] \\\n",
   "  || echo \"WARNING: this notebook targets Ubuntu 24.04, found ${ID:-?} ${VERSION_ID:-?}: prefer a disposable
↪  ubuntu:24.04 container (see above)\"\n",
   "# Not a fresh system? make install writes into /usr/local: an earlier Osmocom there gets overwritten or mixed
↪  in\n",
   "prev=\"$(ls -d /usr/local/lib/libosmo* /usr/local/bin/osmo-* 2>/dev/null | head -5)\"\n",
   "[ -z \"$prev\" ] || { echo \"WARNING: Osmocom already present in /usr/local (this is not a fresh Ubuntu
↪  24.04):\"\n",
   "                     echo \"$prev\" | sed 's/^/           /'\n",
   "                     echo \"         it will be overwritten by this notebook; prefer a disposable ubuntu:24.04
↪  container (see above)\"; }\n",
   "if [ \"$(id -u)\" = 0 ]; then SUDO=\"\"; else SUDO=\"sudo -n\"; fi\n",
   "$SUDO true || { echo \"sudo needs a password: run 'sudo -v' in the terminal that started Jupyter, then
↪  re-run\"; exit 1; }\n",
   "$SUDO mkdir -p /opt/GSM\n",
   "$SUDO chown \"$(id -un):$(id -gn)\" /opt/GSM\n",
   "cat > /opt/GSM/.notebook.env <<EOF\n",
   "export ROOT=/opt/GSM\n",
   "export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig\n",
   "export LD_LIBRARY_PATH=/usr/local/lib\n",
   "export GIT_TERMINAL_PROMPT=0\n",
   "export DEBIAN_FRONTEND=noninteractive\n",
   "SUDO=\"$SUDO\"\n",
   "EOF\n",
   "cat /opt/GSM/.notebook.env"
  ],
  "id": "c02"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": [
   "%%bash\n",
```

```
   ". /opt/GSM/.notebook.env\n",
   "# Check\n",
   "test -w \"$ROOT\" && echo \"OK: $ROOT writable\" && $SUDO true && echo \"OK: sudo without prompt\"\n",
   "df -h /opt | tail -1"
  ],
  "id": "c03"
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "## 1. osmo-operator, then the apt dependencies\n",
   "\n",
   "The repository is cloned first: it carries the patches, the configs, the bridge (`pont/`) and\n",
   "`install.sh`. `install.sh --only deps` installs the apt list **extracted from the `Dockerfile`**, so the\n",
   "two can never diverge. The `debconf` line answers the wireshark-common question in advance (`Dockerfile`,\n",
   "stage `base`), and `DEBIAN_FRONTEND=noninteractive` keeps apt from waiting on a prompt nobody can see."
  ],
  "id": "c04"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": [
   "%%bash\n",
   ". /opt/GSM/.notebook.env\n",
   "set -e\n",
   "$SUDO apt-get update -qq\n",
   "$SUDO apt-get install -y --no-install-recommends git ca-certificates netcat-openbsd   # nc: VTY checks of
↪  step 13\n",
   "cd \"$ROOT\"; [ -d osmo-operator ] || git clone https://github.com/bbaranoff/osmo-operator\n",
   "echo 'wireshark-common wireshark-common/install-setuid boolean true' | $SUDO debconf-set-selections\n",
   "cd \"$ROOT/osmo-operator\"\n",
   "$SUDO env DEBIAN_FRONTEND=noninteractive GSM_ROOT=\"$ROOT\" ./install.sh --only deps\n",
   "$SUDO env GSM_ROOT=\"$ROOT\" ./install.sh --only user"
  ],
  "id": "c05"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": [
   "%%bash\n",
   ". /opt/GSM/.notebook.env\n",
   "# Check: witness packages (sqlite/dbi: osmo-hlr's configure needs them), and the service account.\n",
   "# Fails if anything is missing, so the run stops here and not three steps later.\n",
   "miss=0\n",
   "for p in build-essential libtalloc-dev libsctp-dev libdbi-dev libdbd-sqlite3 libsqlite3-dev libglib2.0-dev
↪  \\\n",
   "        libpixman-1-dev ninja-build python3-numpy socat asterisk tmux tshark liburing-dev libgsm1-dev; do\n",
   "  if dpkg-query -W -f='${db:Status-Status}\\n' \"$p\" 2>/dev/null | grep -qx installed; then echo \"ok
↪  $p\"\n",
   "  else echo \"MISSING $p\"; miss=1; fi\n",
   "done\n",
   "getent passwd osmocom || { echo \"MISSING account osmocom\"; miss=1; }\n",
   "exit $miss"
  ],
  "id": "c06"
 },
 {
  "cell_type": "markdown",
  "metadata": {},
```

```
 "source": [
  "### 1b. (optional) GNU Radio build dependencies\n",
  "\n",
  "Only for the gr-gsm chain (`--grgsm`), `si_bridge.py` and the FFT views. **Not needed for the DSP chain**:\n",
  "the bridge loads libosmocoding through ctypes and does not import GNU Radio."
 ],
 "id": "c07"
},
{
 "cell_type": "code",
 "metadata": {},
 "execution_count": null,
 "outputs": [],
 "source": [
  "%%bash\n",
  ". /opt/GSM/.notebook.env\n",
  "# Dockerfile, end of the apt block: apt-fast build-dep -y gnuradio (needs deb-src)\n",
  "# $SUDO sed -i 's/^Types: deb$/Types: deb deb-src/' /etc/apt/sources.list.d/ubuntu.sources\n",
  "# $SUDO apt-get update && $SUDO apt-get build-dep -y gnuradio\n",
  "echo \"left commented: GNU Radio is not required for the DSP chain\""
 ],
 "id": "c08"
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "## 2. Non-root capture\n",
  "\n",
  "`Dockerfile` stage `base`: `dumpcap` gets the capture capabilities, so Wireshark works without root."
 ],
 "id": "c09"
},
{
 "cell_type": "code",
 "metadata": {},
 "execution_count": null,
 "outputs": [],
 "source": [
  "%%bash\n",
  ". /opt/GSM/.notebook.env\n",
  "$SUDO setcap cap_net_raw,cap_net_admin+eip \"$(command -v dumpcap)\"\n",
  "getcap \"$(command -v dumpcap)\""
 ],
 "id": "c10"
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "## 3. The Osmocom core (stage `osmo-core`)\n",
  "\n",
  "Same repositories, versions and configure options as the `Dockerfile` loop. The three\n",
  "`force-rand-toy` patches (osmo-bts, osmo-hlr, osmo-msc) are applied **before** `configure`, as in the\n",
  "image: RAND becomes deterministic, so authentication is reproducible from one start to the next, which the\n",
  "test suite relies on.\n",
  "\n",
  "`osmo-trx` (IPC patches) is skipped: the DSP and gr-gsm chains use `osmo-bts-trx` with the bridge, not\n",
  "`osmo-trx`. Long: 20–30 min.\n",
  "\n",
  "Each repository's full output (clone, autoreconf, configure, make, install) goes to\n",
  "`$ROOT/logs/<name>.log`; the cell prints one line per repository, or the end of the log on failure."
 ],
 "id": "c11"
},
{
```

```
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": [
   "%%bash\n",
   ". /opt/GSM/.notebook.env\n",
   "P=\"$ROOT/osmo-operator/patches\"\n",
   "mkdir -p \"$ROOT/logs\"\n",
   "# A patch is applied if it applies in reverse; applicable if it applies forward; otherwise the tree does\n",
   "# not match it, which is an error, never \"already applied\".\n",
   "apply_patch() {   # $1 = patch file\n",
   "  if git apply --reverse --check \"$1\" 2>/dev/null; then echo \"  patch already applied: $(basename
↪  \"$1\")\"\n",
   "  elif git apply --check \"$1\" 2>/dev/null; then git apply \"$1\" && echo \"  patched: $(basename
↪  \"$1\")\"\n",
   "  else echo \"  PATCH DOES NOT APPLY: $(basename \"$1\") (tree $(git describe --tags --always) differs from
↪  what the patch expects)\"; return 1; fi\n",
   "}\n",
   "build() {         # $1 = name (already cloned, checked out and patched)\n",
   "  local name=$1 flags=\"\"\n",
   "  cd \"$ROOT/$name\" || return 1\n",
   "  case \"$name\" in\n",
   "    libosmo-abis) flags=\"--disable-dahdi\" ;;\n",
   "    osmo-msc)     flags=\"--enable-smpp\" ;;\n",
   "    osmo-mgw)     flags=\"--enable-alsa\" ;;\n",
   "    osmo-bts)     flags=\"--enable-virtual --enable-trx\" ;;\n",
   "    osmo-ggsn)    flags=\"--enable-gtp-linux\" ;;\n",
   "  esac\n",
   "  autoreconf -fi && ./configure $flags && make -j\"$(nproc)\" \\\n",
   "    && $SUDO make install && $SUDO ldconfig      # Dockerfile: osmo-deb pack \"$name\" \"$version\" make
↪  install\n",
   "}\n",
   "for repo in \\\n",
   "    libosmocore:1.12.1 libosmo-netif:1.7.0 libosmo-abis:2.1.0 libosmo-sigtran:2.2.1 \\\n",
   "    libsmpp34:1.14.5 libgtpnl:1.3.3 osmo-hlr:1.9.2 osmo-mgw:1.15.0 osmo-ggsn:1.14.0 \\\n",
   "    osmo-sgsn:1.13.1 osmo-msc:1.15.0 osmo-bsc:1.14.0 osmo-bts:1.10.0 osmo-pcu:1.5.2 \\\n",
   "    osmo-sip-connector:1.7.2 libosmo-gprs:0.2.1; do\n",
   "  name=${repo%%:*}; version=${repo##*:}\n",
   "  case \"$name\" in libosmo*) url=\"https://gitea.osmocom.org/osmocom/$name\" ;;\n",
   "                  *)        url=\"https://gitea.osmocom.org/cellular-infrastructure/$name\" ;; esac\n",
   "  log=\"$ROOT/logs/$name.log\"\n",
   "  # clone first (outside the log redirection) so the patch can be checked against the right version\n",
   "  ( cd \"$ROOT\"; [ -d \"$name\" ] || git clone -q \"$url\"; cd \"$name\" && git checkout -q \"$version\" ) >
↪  \"$log\" 2>&1 \\\n",
   "    || { echo \"FAILED $name (clone/checkout $version)\"; tail -20 \"$log\"; exit 1; }\n",
   "  if [ -f \"$P/$name-force-rand-toy.patch\" ]; then\n",
   "    ( cd \"$ROOT/$name\" && apply_patch \"$P/$name-force-rand-toy.patch\" ) || exit 1\n",
   "  fi\n",
   "  if build \"$name\" >> \"$log\" 2>&1; then\n",
   "    echo \"ok     $name $version\"\n",
   "  else\n",
   "    echo \"FAILED $name $version  (full log: $log)\"; tail -20 \"$log\"; exit 1\n",
   "  fi\n",
   "done"
  ],
  "id": "c12"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": [
   "%%bash\n",
   ". /opt/GSM/.notebook.env\n",
```

```
   "# Check: libraries for c54x_exe (pkg-config libosmocoding) and the daemons install.sh's verify step looks
↪  for.\n",
   "# Fails if anything is missing.\n",
   "miss=0\n",
   "pkg-config --modversion libosmocore libosmocoding || miss=1\n",
   "for b in osmo-stp osmo-hlr osmo-msc osmo-mgw osmo-bsc osmo-bts-trx osmo-pcu osmo-sgsn osmo-ggsn
↪  osmo-sip-connector; do\n",
   "  if command -v \"$b\" >/dev/null; then echo \"ok      $b\"; else echo \"MISSING $b\"; miss=1; fi\n",
   "done\n",
   "exit $miss"
  ],
  "id": "c13"
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "## 4. osmo-gapk (before osmocom-bb)\n",
   "\n",
   "The bench's mobile uses `tch-voice io-handler gapk`: osmocom-bb must find libosmogapk at configure time,\n",
   "so gapk comes **first**."
  ],
  "id": "c14"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": [
   "%%bash\n",
   ". /opt/GSM/.notebook.env\n",
   "set -e\n",
   "cd \"$ROOT\"; [ -d osmo-gapk ] || git clone -q https://gitea.osmocom.org/osmocom/gapk osmo-gapk\n",
   "cd osmo-gapk && autoreconf -fi >/dev/null && ./configure -q --enable-alsa && make -s -j\"$(nproc)\"\n",
   "$SUDO make -s install && $SUDO ldconfig"
  ],
  "id": "c15"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": [
   "%%bash\n",
   ". /opt/GSM/.notebook.env\n",
   "# Check\n",
   "pkg-config --modversion libosmogapk && command -v osmo-gapk"
  ],
  "id": "c16"
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "## 5. The SMSC: gsup-smsc-proto and sms-coding-utils\n",
   "\n",
   "`proto-smsc-daemon` receives MO SMS from OsmoHLR over GSUP, and `proto-smsc-sendmt` injects MT SMS.\n",
   "`run_modules/22-smsc.sh` starts them: **without them there is no SMS**. Its top-level Makefile forces\n",
   "`DESTDIR=`, so the two sub-directories are installed one by one, as in the `Dockerfile`.\n",
   "`sms-coding-utils` (FreeCalypso) adds the PDU encode/decode tools."
  ],
  "id": "c17"
 },
 {
```

```
 "cell_type": "code",
 "metadata": {},
 "execution_count": null,
 "outputs": [],
 "source": [
  "%%bash\n",
  ". /opt/GSM/.notebook.env\n",
  "set -e\n",
  "cd \"$ROOT\"; [ -d gsup-smsc-proto ] || git clone -q https://gitea.osmocom.org/themwi/gsup-smsc-proto\n",
  "cd gsup-smsc-proto && ./configure --with-osmo=/usr/local && make -s -j\"$(nproc)\"\n",
  "for d in daemon sendmt; do $SUDO make -s -C \"$d\" install; done\n",
  "cd \"$ROOT\"\n",
  "if [ ! -d sms-coding-utils-r1 ]; then\n",
  "  wget -q https://www.freecalypso.org/pub/GSM/FreeCalypso/sms-coding-utils-latest.tar.bz2\n",
  "  tar xf sms-coding-utils-latest.tar.bz2\n",
  "fi\n",
  "cd sms-coding-utils-r1 && ./configure && make -s -j\"$(nproc)\" && $SUDO make -s install\n",
  "$SUDO ldconfig"
 ],
 "id": "c18"
},
{
 "cell_type": "code",
 "metadata": {},
 "execution_count": null,
 "outputs": [],
 "source": [
  "%%bash\n",
  ". /opt/GSM/.notebook.env\n",
  "# Check\n",
  "command -v proto-smsc-daemon proto-smsc-sendmt sms-encode-text"
 ],
 "id": "c19"
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "## 6. osmocom-bb (stage `bb`): osmocon, mobile, trxcon\n",
  "\n",
  "Clone, the **three** patches the image applies (`clck-gen-frame-tolerance` for fake_trx,\n",
  "`plmn-a6-reinsert` and `sim-plmn-from-bcd` for the mobile's PLMN selection), then `make nofirmware`\n",
  "(host tools only: the firmware comes from step 7). `osmocon` stays in the tree, where `run.sh` and\n",
  "`c54x_exe/run.sh` look for it."
 ],
 "id": "c20"
},
{
 "cell_type": "code",
 "metadata": {},
 "execution_count": null,
 "outputs": [],
 "source": [
  "%%bash\n",
  ". /opt/GSM/.notebook.env\n",
  "set -e\n",
  "P=\"$ROOT/osmo-operator/patches\"\n",
  "# [2026-10-23] pinned like the Dockerfile (OSMOCOM_BB_REF): upstream a0a315e2 moved trxcon to libosmo-trx and
↪  out of\n",
  "# `make nofirmware`, so a clone of master no longer builds src/host/trxcon/src/trxcon\n",
  "cd \"$ROOT\"; [ -d osmocom-bb ] || { git clone -q https://gitea.osmocom.org/phone-side/osmocom-bb \\\n",
  "                                   && git -C osmocom-bb checkout -q \"${OSMOCOM_BB_REF:-eac4eed1}\"; }\n",
  "# Same three cases as step 3: applied (reverse applies) / applicable / does not match = error\n",
  "for p in clck-gen-frame-tolerance plmn-a6-reinsert sim-plmn-from-bcd; do\n",
  "  f=\"$P/osmocom-bb-$p.patch\"\n",
  "  if git -C osmocom-bb apply --reverse --check \"$f\" 2>/dev/null; then echo \"patch already applied: $p\"\n",
```

```
   "  elif git -C osmocom-bb apply --check \"$f\" 2>/dev/null; then git -C osmocom-bb apply \"$f\" && echo
↪  \"patched: $p\"\n",
   "  else echo \"PATCH DOES NOT APPLY: osmocom-bb-$p.patch (tree $(git -C osmocom-bb describe --tags --always)
↪  differs from what the patch expects)\"; exit 1; fi\n",
   "done\n",
   "cd osmocom-bb/src && make -s nofirmware\n",
   "for b in trxcon:host/trxcon/src/trxcon mobile:host/layer23/src/mobile/mobile \\\n",
   "         virtphy:host/virt_phy/src/virtphy ccch_scan:host/layer23/src/misc/ccch_scan; do\n",
   "  $SUDO install -m755 \"$ROOT/osmocom-bb/src/${b#*:}\" \"/usr/local/bin/${b%%:*}\"\n",
   "done"
  ],
  "id": "c21"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": [
   "%%bash\n",
   ". /opt/GSM/.notebook.env\n",
   "# Check: osmocon in the tree, mobile in the PATH and linked to gapk\n",
   "ls -la \"$ROOT/osmocom-bb/src/host/osmocon/osmocon\" && command -v mobile\n",
   "ldd \"$(command -v mobile)\" | grep -i gapk || echo \"WARNING: mobile is not linked to libosmogapk (gapk
↪  built after osmocom-bb?)\""
  ],
  "id": "c22"
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "## 7. Prebuilt Calypso firmware\n",
   "\n",
   "`make nofirmware` does not build the firmware: the bench takes the binaries from `bbaranoff/firmware`,\n",
   "the only path QEMU (`-kernel .elf`) and osmocon (`.bin`) read (`qosmo/environnement/bench.env`)."
  ],
  "id": "c23"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": [
   "%%bash\n",
   ". /opt/GSM/.notebook.env\n",
   "cd \"$ROOT\"; [ -d firmware ] || git clone -q --depth 1 https://github.com/bbaranoff/firmware\n",
   "ls -la \"$ROOT\"/firmware/board/compal_e88/layer1.highram.{elf,bin}"
  ],
  "id": "c24"
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "### 7b. The transceiver tree and the `trx` firmware (`--osmo-trx`)\n",
   "\n",
   "[2026-10-20] A second osmocom-bb tree, Sylvain Munaut's `jolly/testing` branch, provides the `trx`\n",
   "firmware application (the Calypso as a BTS transceiver) and the host `transceiver` program that speaks\n",
   "TRXD (ports 5700-5702) to osmo-bts-trx. The bench uses them with `./start.sh --bts` (alias `--osmo-trx`)\n",
   "(`c54x_exe/run.sh` with `OSMO_TRX=1`, alias `banc-trx`). The project patch\n",
   "`patches/osmocom-bb-transceiver-qemu-bench.patch` covers both sides: the host program (`--no-sync`,\n",
  "`--ts-mask`/`--tx-mask`, `--l1ctl`, `--base-port`, SETTSC/SETFORMAT/NOMTXPOWER accepted) and the firmware\n",
   "(build with `arm-none-eabi-gcc`, `prim_bts.c`: RX offset reset and TX slot/timeslot mapping)."
  ],
```

```
  "id": "x01"
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "metadata": {},
  "outputs": [],
  "source": [
   "%%bash\n",
   ". /opt/GSM/.notebook.env\n",
   "set -e\n",
   "P=\"$ROOT/osmo-operator/patches\"; T=\"$ROOT/osmocom-bb-transceiver\"\n",
   "[ -d \"$T\" ] || git clone -q --branch jolly/testing https://gitea.osmocom.org/phone-side/osmocom-bb.git
↪  \"$T\"\n",
   "f=\"$P/osmocom-bb-transceiver-qemu-bench.patch\"\n",
   "if git -C \"$T\" apply --reverse --check \"$f\" 2>/dev/null; then echo \"patch already applied\"\n",
   "elif git -C \"$T\" apply --check \"$f\" 2>/dev/null; then git -C \"$T\" apply \"$f\" && echo \"patched\"\n",
   "else echo \"PATCH DOES NOT APPLY (tree $(git -C \"$T\" describe --always) differs)\"; exit 1; fi\n",
   "# host side: the transceiver program (layer23 configured with --enable-transceiver)\n",
   "cd \"$T/src\" && make -s HOST_layer23_CONFARGS=--enable-transceiver nofirmware\n",
   "$SUDO install -m755 \"$T/src/host/layer23/src/transceiver/transceiver\" /usr/local/bin/transceiver\n",
   "transceiver -h 2>&1 | grep -E 'no-sync|ts-mask|tx-mask'"
  ],
  "id": "x02"
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "metadata": {},
  "outputs": [],
  "source": [
   "%%bash\n",
   ". /opt/GSM/.notebook.env\n",
   "set -e\n",
   "T=\"$ROOT/osmocom-bb-transceiver/src/target/firmware\"\n",
   "# firmware side: trx.highram.{elf,bin} with the system arm-none-eabi toolchain (libosmocore for the target
↪  first)\n",
   "command -v arm-none-eabi-gcc >/dev/null || $SUDO apt-get install -y gcc-arm-none-eabi\n",
   "cd \"$ROOT/osmocom-bb-transceiver/src\"\n",
   "[ -f shared/libosmocore/build-target/src/.libs/libosmocore.a ] || { rm -rf shared/libosmocore/build-target &&
↪  mkdir shared/libosmocore/build-target \\\n",
   "  && cd shared/libosmocore/build-target && ../configure --host=arm-none-eabi --enable-embedded
↪  --disable-shared --disable-tests \\\n",
   "     ac_cv_header_sys_select_h=no ac_cv_header_sys_socket_h=no \\\n",
   "     CFLAGS=\"-Os -ffunction-sections -fcommon -I$T/include -nostartfiles -nodefaultlibs\" >/dev/null && make
↪  -s; }\n",
   "cd \"$T\" && make -s BOARDS=compal_e88 APPLICATIONS=trx ENVIRONMENTS=highram CROSS_COMPILE=arm-none-eabi-
↪  2>&1 | grep -v '^make' | tail -3 || true\n",
   "ls -la board/compal_e88/trx.highram.elf board/compal_e88/trx.highram.bin\n",
   "$SUDO install -m755 board/compal_e88/trx.highram.elf board/compal_e88/trx.highram.bin
↪  \"$ROOT/firmware/board/compal_e88/\""
  ],
  "id": "x03"
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "Optional: the historical `arm-elf` toolchain (gcc 4.5.2, binutils 2.21.1, newlib 1.19) from the\n",
   "[gnu-arm-installer](https://github.com/axilirator/gnu-arm-installer), the one osmocom-bb documents for
↪  its\n",
   "firmware, installed in `/root/gnuarm/install`. On Ubuntu 24.04 its `cc1` wants `libmpfr.so.4`; a symlink
↪  to\n",
   "the system's `libmpfr.so.6` is enough."
  ],
  "id": "x04"
```

```
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "metadata": {},
  "outputs": [],
  "source": [
   "%%bash\n",
   ". /opt/GSM/.notebook.env\n",
   "[ -x /root/gnuarm/install/bin/arm-elf-gcc ] || { echo \"gnuarm not installed (optional): git clone
↪  https://github.com/axilirator/gnu-arm-installer /root/gnuarm && cd /root/gnuarm && ./download.sh &&
↪  ./build.sh\"; exit 0; }\n",
   "[ -e /usr/local/lib/libmpfr.so.4 ] || { $SUDO ln -sf /usr/lib/x86_64-linux-gnu/libmpfr.so.6
↪  /usr/local/lib/libmpfr.so.4 && $SUDO ldconfig; }\n",
   "/root/gnuarm/install/bin/arm-elf-gcc --version | head -1"
  ],
  "id": "x05"
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "## 8. qosmo: the Calypso QEMU (stage `qemu`)\n",
   "\n",
   "One binary for both chains: built with `--enable-l1-grgsm`, it runs the DSP chain with\n",
   "`CALYPSO_DSP_EXTERN=1` set at launch. `ninja install` also installs the `qosmo` launcher next to\n",
   "`qemu-system-arm`. Long: 10–15 min."
  ],
  "id": "c25"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": [
   "%%bash\n",
   ". /opt/GSM/.notebook.env\n",
   "set -e\n",
   "cd \"$ROOT\"; [ -d qosmo ] || git clone -q https://github.com/bbaranoff/qosmO qosmo\n",
   "cd qosmo\n",
   "python3 -m venv ~/.venv-qemu && . ~/.venv-qemu/bin/activate\n",
   "pip install -q --no-cache-dir tomli\n",
   "mkdir -p build && cd build\n",
   "[ -f build.ninja ] || ../configure --target-list=arm-softmmu --enable-l1-grgsm \\\n",
   "    --prefix=\"$ROOT/qemu-install\" --disable-werror --disable-docs\n",
   "make -s -j\"$(nproc)\"\n",
   "make -s install\n",
   "$SUDO cp \"$ROOT/qemu-install/bin/qemu-system-arm\" /usr/local/bin/qemu-system-arm\n",
   "$SUDO cp \"$ROOT/qemu-install/bin/qosmo\" /usr/local/bin/qosmo"
  ],
  "id": "c26"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": [
   "%%bash\n",
   ". /opt/GSM/.notebook.env\n",
   "# Check: the binary run.sh expects (QEMU_BIN in qosmo/environnement/bench.env), the launcher, the linked
↪  layer 1\n",
   "ls -la \"$ROOT/qosmo/build/qemu-system-arm\"\n",
   "qosmo -V\n",
   "\"$ROOT/qosmo/build/qemu-system-arm\" -M help | grep -iE \"calypso|couche\""
```

```
 ],
 "id": "c27"
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "## 9. c54x_exe and the DSP ROM (stage `l1`)\n",
  "\n",
  "`c54x_exe` compiles qosmo's sources and links against libosmocoding. The CFLAGS are the Makefile's\n",
  "**without `-march=native`**. The ROM is not in the repository: `rom/fetch-rom.sh` downloads FreeCalypso's\n",
  "dump 3606, converts it and checks its checksum. The raw dump `calypso_dsp.txt` is downloaded separately,\n",
  "as in the image."
 ],
 "id": "c28"
},
{
 "cell_type": "code",
 "metadata": {},
 "execution_count": null,
 "outputs": [],
 "source": [
  "%%bash\n",
  ". /opt/GSM/.notebook.env\n",
  "set -e\n",
  "F=ftp://ftp.freecalypso.org/pub/GSM/Calypso\n",
  "cd \"$ROOT\"; [ -d c54x_exe ] || git clone -q https://github.com/bbaranoff/c54x_exe\n",
  "cd c54x_exe\n",
  "make -s QOSMO=\"$ROOT/qosmo\" \\\n",
  "  CFLAGS=\"-O3 -g -Wall -Werror=format -Werror=format-extra-args -Wno-unused-function -Wno-unused-variable
↪  -Wno-unused-but-set-variable -Wno-sign-compare\"\n",
  "FREECALYPSO_URL=\"$F\" bash rom/fetch-rom.sh --dest \"$ROOT\" --dest \"$ROOT/c54x_exe/rom\"\n",
  "[ -s \"$ROOT/calypso_dsp.txt\" ] || curl -fsS --retry 2 -o \"$ROOT/calypso_dsp.txt\"
↪  \"$F/dsp-rom-3606-dump.txt\"\n",
  "[ -e rom/calypso_dsp.txt ] || cp \"$ROOT/calypso_dsp.txt\" rom/calypso_dsp.txt"
 ],
 "id": "c29"
},
{
 "cell_type": "code",
 "metadata": {},
 "execution_count": null,
 "outputs": [],
 "source": [
  "%%bash\n",
  ". /opt/GSM/.notebook.env\n",
  "# Check: the ROM (fetch-rom.sh re-verifies it) and the DSP alone, no ARM (c54x_exe/README.md)\n",
  "cd \"$ROOT/c54x_exe\"\n",
  "ls \"$ROOT\"/calypso_dsp.{PROM0,PROM1,PROM2,PROM3,DROM,PDROM}.bin\n",
  "./c54x_exe --trames 200 | tail -5"
 ],
 "id": "c30"
},
{
 "cell_type": "markdown",
 "id": "m01",
 "metadata": {},
 "source": [
  "### 9b. (optional) FreeCalypso Magnetite, the real TI firmware for the C139\n",
  "\n",
  "[2026-10-21] With `--ms-magnetite` the emulated mobile does not run the osmocom-bb layer 1 but the
↪  original\n",
  "TI TCS211 firmware, rebuilt by FreeCalypso Magnetite (target `c139`, configuration `hybrid-ui-vo`):
↪  Nucleus,\n",
  "TI's L2/L3 stack and the phone UI all run on the emulated ARM, with the DSP of step 9 under it. There is no\n",
  "osmocon and no `mobile`: `rvinterf` (FreeCalypso host tools) decodes the firmware's traces, and the C139's\n",
```

```
   "colour screen and keypad are in `tools/osmo-c123-gtk.py --modele c139`.\n",
   "\n",
   "Magnetite does **not** build with gcc: it needs TI's TMS470 compiler, a Windows binary run under Wine\n",
   "(environment `installed-env.tar.xz` from ftp.freecalypso.org). The 4 MiB flash image the bench loads is\n",
   "the FreeCalypso Compal bootloader (`compal-flash-boot-for-fc.bin`, 8 KiB, at 0) plus `fwimage.bin` at\n",
   "0x10000: `/opt/GSM/firmware/magnetite/c139-flash.img` (`MAGNETITE_FLASH` overrides it). Full notes, and\n",
   "the qosmo changes it relies on: [doc/magnetite.md](../doc/magnetite.md). Not needed for the call of step
↪  13.\n"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "id": "m02",
  "metadata": {},
  "outputs": [],
  "source": [
   "%%bash\n",
   ". /opt/GSM/.notebook.env\n",
   "[ \"${NB_MAGNETITE:-0}\" = 1 ] || { echo \"Magnetite skipped (optional): NB_MAGNETITE=1 in the environment of
↪  Jupyter to build it\"; exit 0; }\n",
   "set -e\n",
   "cd \"$ROOT\"\n",
   "# Mercurial (the two FreeCalypso repositories) in a venv, Wine with its 32-bit side\n",
   "[ -x ~/.venv-hg/bin/hg ] || { python3 -m venv ~/.venv-hg && ~/.venv-hg/bin/pip install -q mercurial; }\n",
   "[ -d fc-magnetite ]  || ~/.venv-hg/bin/hg clone -q https://hg.themyscira.org/fc-magnetite fc-magnetite\n",
   "[ -d fc-host-tools ] || ~/.venv-hg/bin/hg clone -q https://hg.themyscira.org/freecalypso-tools
↪  fc-host-tools\n",
   "make -s -C fc-host-tools                     # rvinterf, fc-shell, fc-fsio (run.sh:
↪  fc-host-tools/rvinterf/lowlevel/rvinterf)\n",
   "command -v wine >/dev/null || { $SUDO dpkg --add-architecture i386 && $SUDO apt-get update -qq \\\n",
   "                               && $SUDO apt-get install -y wine64 wine32:i386; }\n",
   "# the TI environment (compiler, cygwin, perl, java) and the nowhine wrapper, same FTP directory\n",
   "W=ftp://ftp.freecalypso.org/pub/GSM/TI_src/wine; mkdir -p fc-wine-dl\n",
   "[ -s fc-wine-dl/installed-env.tar.xz ] || curl -fsS -o fc-wine-dl/installed-env.tar.xz
↪  \"$W/installed-env.tar.xz\"\n",
   "[ -s fc-wine-dl/nowhine.c ]            || curl -fsS -o fc-wine-dl/nowhine.c \"$W/nowhine.c\"\n",
   "[ -d ~/.wine/drive_c/Dvpt_tools ] || { mkdir -p ~/.wine/drive_c && tar xJf fc-wine-dl/installed-env.tar.xz -C
↪  ~/.wine/drive_c && wineboot -u; }\n",
   "command -v nowhine >/dev/null || $SUDO gcc -O2 -o /usr/local/bin/nowhine fc-wine-dl/nowhine.c\n",
   "cd fc-magnetite && (cd helpers && make -s)\n",
   "[ -d build-c139-hybrid-ui-vo ] || ./configure.sh c139 hybrid-ui-vo\n",
   "make -s -C build-c139-hybrid-ui-vo -j4        # ~1 min, 4 cl470 under Wine\n",
   "ls -la build-c139-hybrid-ui-vo/fwimage.bin\n"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "id": "m03",
  "metadata": {},
  "outputs": [],
  "source": [
   "%%bash\n",
   ". /opt/GSM/.notebook.env\n",
   "[ \"${NB_MAGNETITE:-0}\" = 1 ] || { echo \"Magnetite skipped (optional, NB_MAGNETITE=1)\"; exit 0; }\n",
   "# The flash image: bootloader at 0, fwimage.bin at 0x10000, 0xFF elsewhere (doc/magnetite.md, section 2).\n",
   "# The bootloader is not built here: put compal-flash-boot-for-fc.bin (FreeCalypso) in $ROOT/fc-dl first.\n",
   "B=\"$ROOT/fc-dl/compal-flash-boot-for-fc.bin\";
↪  F=\"$ROOT/fc-magnetite/build-c139-hybrid-ui-vo/fwimage.bin\"\n",
   "[ -s \"$B\" ] || { echo \"MISSING $B (FreeCalypso Compal bootloader, 8 KiB)\"; exit 1; }\n",
   "mkdir -p \"$ROOT/firmware/magnetite\"\n",
   "python3 - \"$B\" \"$F\" \"$ROOT/firmware/magnetite/c139-flash.img\" <<'EOF'\n",
   "import sys\n",
   "img = bytearray(b\"\\xff\" * (4 << 20))\n",
   "boot = open(sys.argv[1], \"rb\").read(); fw = open(sys.argv[2], \"rb\").read()\n",
```

```
  "img[0:len(boot)] = boot; img[0x10000:0x10000 + len(fw)] = fw\n",
  "open(sys.argv[3], \"wb\").write(img)\n",
  "EOF\n",
  "# Check: the image and the tools the bench uses with it\n",
  "ls -la \"$ROOT/firmware/magnetite/c139-flash.img\" \"$ROOT/fc-host-tools/rvinterf/lowlevel/rvinterf\"\n",
  "\"$ROOT/osmo-operator/scripts/magnetite-c139.sh\" status 2>/dev/null || true\n"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "## 10. (optional) grgsm_exe\n",
  "\n",
  "Bench tool (gr-gsm layer 1 outside QEMU, no firmware): it does **not** make the mobile camp. Not needed\n",
  "for a call on the DSP."
 ],
 "id": "c31"
},
{
 "cell_type": "code",
 "metadata": {},
 "execution_count": null,
 "outputs": [],
 "source": [
  "%%bash\n",
  ". /opt/GSM/.notebook.env\n",
  "cd \"$ROOT\"; [ -d grgsm_exe ] || git clone -q https://github.com/bbaranoff/grgsm_exE grgsm_exe\n",
  "cd grgsm_exe && make -s QOSMO=\"$ROOT/qosmo\" && ./grgsm_exe --trames 5000 | tail -5"
 ],
 "id": "c32"
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "## 11. Runtime setup\n",
  "\n",
  "What the end of the `Dockerfile` does for the running stack:\n",
  "\n",
  "- `install.sh --only configs`: the templates of `configs/` into `/etc/osmocom` (never overwrites an\n",
  "  existing file) and `mobile.cfg`;\n",
  "- the run-time scripts of `scripts/` into `/etc/osmocom`: `run_modules/22-smsc.sh` starts\n",
  "  `/etc/osmocom/smsc-start.sh` and `sms-interop-relay.py`, the audio chain uses `gapk-start.sh`;\n",
  "- the run, log and Asterisk directories;\n",
  "- the cell: the DSP mobile looks for **ARFCN 514, BSIC 7** (`stick 514` in `c54x_exe/mobile_pont.cfg`,\n",
  "  `pont/config.py`), which is the bench's default in `globals.conf`.\n",
  "\n",
  "The `__MCC__`, `__ARFCN__`… markers are filled at launch by `./start.sh --regen` from `globals.conf`.\n",
  "\n",
  "[2026-10-23] The bench's shell helpers (`banc-trx`, `banc-mobile`, `banc-rssi`, `banc-stop`, `c123`…)\n",
  "come from `scripts/osmo-banc-aliases.sh`, installed as `/etc/profile.d/98-osmo-banc-aliases.sh` (login\n",
  "shells source it)."
 ],
 "id": "c33"
},
{
 "cell_type": "code",
 "metadata": {},
 "execution_count": null,
 "outputs": [],
 "source": [
  "%%bash\n",
  ". /opt/GSM/.notebook.env\n",
  "set -e\n",
  "O=\"$ROOT/osmo-operator\"\n",
```

```
   "cd \"$O\" && $SUDO env GSM_ROOT=\"$ROOT\" OSMOCOM_HOME=/root/.osmocom ./install.sh --only configs\n",
   "$SUDO mkdir -p /var/log/osmocom /var/run/smsc /var/lib/gapk /root/.osmocom/bb \\\n",
   "               /etc/asterisk /var/lib/asterisk /var/log/asterisk /var/run/asterisk\n",
   "$SUDO install -d -o osmocom -g osmocom /var/lib/osmocom    # osmo-hlr keeps its database there\n",
   "for f in \"$O\"/scripts/*.sh \"$O\"/scripts/*.py; do\n",
   "  [ \"$(basename \"$f\")\" = run.sh ] && continue        # the image moves it to /root/run.sh; start.sh uses
↪  qosmo/run.sh\n",
   "  $SUDO install -m755 \"$f\" /etc/osmocom/\n",
   "done\n",
   "$SUDO ln -sf /etc/osmocom/gapk-start.sh /usr/local/bin/gapk-start.sh\n",
   "$SUDO install -m644 \"$O/scripts/osmo-banc-aliases.sh\" /etc/profile.d/98-osmo-banc-aliases.sh   #
↪  banc-rssi, banc-stop, c123...\n",
   "$SUDO test -e /root/.osmocom/bb/mobile.cfg || $SUDO cp \"$O/configs/mobile.cfg\"
↪  /root/.osmocom/bb/mobile.cfg"
  ],
  "id": "c34"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": [
   "%%bash\n",
   ". /opt/GSM/.notebook.env\n",
   "# Check\n",
   "ls /etc/osmocom/osmo-bsc.cfg /etc/osmocom/smsc-start.sh /etc/osmocom/sms-interop-relay.py
↪  /etc/osmocom/gapk-start.sh\n",
   "ls -ld /var/lib/osmocom\n",
   "# effective radio/PLMN settings (an empty value in globals.conf means the built-in default)\n",
   "cd \"$ROOT/osmo-operator\" && ./generate_configs.sh --show | grep -E \"MCC|MNC|BAND|ARFCN|BSIC|LAC\" | head"
  ],
  "id": "c35"
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "### 11b. (optional) audio sinks\n",
   "\n",
   "Only for **hearing** the call (the `voix` rung of the tests). The core, the camp, the location update,\n",
   "the SMS and the call set-up do not need audio. Ubuntu 24.04 desktop runs PipeWire with its PulseAudio\n",
   "server, which `pactl` drives. `configs/asound.conf` routes ALSA to Pulse (`type pulse`), which needs\n",
   "`libasound2-plugins`, installed at step 1."
  ],
  "id": "c36"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": [
   "%%bash\n",
   "# run as YOUR user (the desktop session's audio server), not with sudo\n",
   "for s in gsm_audio gsm_mic; do\n",
   "  pactl list short sinks | grep -q \"$s\" || pactl load-module module-null-sink sink_name=$s format=s16le
↪  rate=8000 channels=1\n",
   "done\n",
   "pactl list short sinks | grep gsm_ || echo \"no audio server reachable (server install): skip, the call still
↪  sets up\""
  ],
  "id": "c37"
 },
 {
  "cell_type": "markdown",
```

```
  "metadata": {},
  "source": [
   "## 12. Start: the network, the BTS, then the DSP bench\n",
   "\n",
   "`./start.sh` (DSP by default since 2026-10-01; `start-native.sh` is the same script under its old name)\n",
   "plays `qosmo/run.sh`'s plan **without** `qemu,pty,osmocon,l2` (Osmocom core, HLR subscribers,\n",
   "osmo-bts-trx), then hands over to `c54x_exe/run.sh` for the phone chain. First the dependency check and a\n",
   "dry run, then the real start. `./start.sh --help` lists every option; the ones that pick the set-up:\n",
   "\n",
   "| option | what runs |\n",
   "|---|---|\n",
   "| `--bts-classic` / `--bts-classique` | the **classic BTS**: osmo-bts-trx talks to `pont/pont_dsp.py` (TRXD
↪  5700-5702), the emulated phone (QEMU + c54x_exe) is plugged on the bridge. The default set-up of `--dsp`,
↪  named; exclusive with `--bts` / `--ts` |\n",
   "| `--ms` | the emulated phone is an osmocom-bb **mobile** (the default) |\n",
   "| `--ms-magnetite` | like `--ms`, but the mobile runs the real TI firmware, FreeCalypso Magnetite for the
↪  C139 (step 9b) |\n",
   "| `--bts` (alias `--osmo-trx`) | the Calypso of QEMU **is the TRX** of BTS#0 (`trx.highram.elf` + host
↪  `transceiver`); with `--ms` / `--ms-magnetite` a second Calypso camps on it over the air |\n",
   "| `--ts N` | timeslots of that Calypso TRX (implies `--bts`): `1` TS0 only, `2` one Calypso per timeslot,
↪  `2/1` TS0 + TS1 on one Calypso (the default) |\n",
   "| `--vocoder-rom` / `--gapk` | the speech codec: the DSP mask ROM's FR codec (default), or the host
↪  (osmo-gapk in the mobile; with Magnetite, `pont/voix_hote.py` in the bridge). The emulated DSP keeps real time
↪  only with `--gapk` |\n",
   "| `--no-attach` | do not attach to tmux at the end (what a notebook needs) |\n",
   "\n",
   "[2026-10-20] With `--bts`, the emulated phone boots `trx.highram.elf` and becomes the **transceiver of\n",
   "BTS#0** (host `transceiver` on TRXD 5700-5702, no bridge, no mobile). A second emulated phone
↪  (`INSTANCE=2`,\n",
   "alias `banc-mobile`) then camps on it through `pont/air.py`. See the run notebook, step 12b, and\n",
   "`wiki/Reference.md` section 2.7.\n"
  ],
  "id": "c38"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": [
   "%%bash\n",
   ". /opt/GSM/.notebook.env\n",
   "# --check-paths only reads; --dry-run walks the plan with no side effects\n",
   "cd \"$ROOT/osmo-operator\" && $SUDO ./start.sh --check-paths && $SUDO ./start.sh --dry-run"
  ],
  "id": "c39"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": [
   "%%bash\n",
   ". /opt/GSM/.notebook.env\n",
   "# --regen fills the templates of /etc/osmocom ; --no-attach: do not take the terminal (no tmux attach)\n",
   "# classic BTS (osmo-bts-trx + pont_dsp.py) and the osmocom-bb mobile on the bridge\n",
   "cd \"$ROOT/osmo-operator\" && $SUDO ./start.sh --regen --bts-classique --ms --no-attach\n"
  ],
  "id": "c40"
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "id": "m04",
  "metadata": {},
```

```
  "outputs": [],
  "source": [
   "%%bash\n",
   ". /opt/GSM/.notebook.env\n",
   "[ \"${NB_MAGNETITE:-0}\" = 1 ] && [ -s \"$ROOT/firmware/magnetite/c139-flash.img\" ] || { echo \"Magnetite
↪  variant skipped (NB_MAGNETITE=1 and step 9b)\"; exit 0; }\n",
   "# Variant (instead of the previous cell, needs step 9b): the C139 Magnetite on the classic BTS, voice by the
↪  host.\n",
   "# ENCRYPTION=\"a5 0\": calls on Magnetite need it (with A5/1 the FACCH fails and the call never sets up);\n",
   "# --regen carries it into osmo-bsc.cfg / osmo-msc.cfg (encryption a5 0). Screen: c123, or\n",
   "#   /usr/bin/python3 $ROOT/osmo-operator/tools/osmo-c123-gtk.py --modele c139 --instance 1\n",
   "cd \"$ROOT/osmo-operator\" && $SUDO env ENCRYPTION=\"a5 0\" ./start.sh --regen --bts-classique --ms-magnetite
↪  --gapk --no-attach\n",
   "grep -E '^ *encryption' /etc/osmocom/osmo-bsc.cfg /etc/osmocom/osmo-msc.cfg\n"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "metadata": {},
  "outputs": [],
  "source": [
   "%%bash\n",
   ". /opt/GSM/.notebook.env\n",
   "# Variant: the phone as the BTS transceiver (instead of the previous cells). Expected afterwards in\n",
   "# /tmp/c54x-pont/dsp.log: \"[rom-tx] ... SB n, FB n\" with SB = FB = 1/10 of the bursts; no \"STALE BURST\"
↪  in\n",
   "# osmocon.log once osmo-bts-trx has \"osmotrx fn-advance 20\" (configs/osmo-bts-trx.cfg).\n",
   "# Add --ms for a second phone over the air (instance 2), or --ms-magnetite for the C139.\n",
   "cd \"$ROOT/osmo-operator\" && $SUDO env NO_MENU=1 ./start.sh --bts --no-attach\n",
   "grep -a 'rom-tx\\] [0-9]' /tmp/c54x-pont/dsp.log | tail -1\n"
  ],
  "id": "x06"
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "Manual alternative, process by process: the notebook [lancer-pas-a-pas.ipynb](lancer-pas-a-pas.ipynb)\n",
   "(network daemons one by one, then `c54x_exe --arm`, QEMU, osmocon, mobile, `pont_dsp.py`), or the page\n",
   "[Lancer-a-la-main](Lancer-a-la-main.md)."
  ],
  "id": "c41"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": [
   "%%bash\n",
   "# Check the phone chain (c54x_exe/run.sh --status) and its key log lines\n",
   "# (Magnetite: no osmocon, the third line is then rvinterf's trace log, /tmp/c54x-pont/rvinterf.log)\n",
   "/opt/GSM/c54x_exe/run.sh --status\n",
   "grep -a \"en attente de l'ARM\"          /tmp/c54x-pont/dsp.log     | tail -1\n",
   "grep -a \"pont DSP : API RAM partagee\"  /tmp/c54x-pont/qemu.log    | tail -1\n",
   "grep -a \"your code is running now\"     /tmp/c54x-pont/osmocon.log | tail -1\n",
   "grep -a \"pont TRX : ports\"             /tmp/c54x-pont/pont.log    | tail -1"
  ],
  "id": "c42"
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "## 13. Camp, location update, then a call\n",
```

```
   "\n",
   "The same checks as the test suite ([bbaranoff/tests](https://github.com/bbaranoff/tests),\n",
   "`modules/_lib.sh`, `60-camp.sh`, `70-attache.sh`, `90-appel.sh`). The DSP mobile's VTY is **4247**\n",
   "(`c54x_exe/mobile_pont.cfg`, `bind 127.0.0.1 4247`), the MSC's is 4254. `600` is the echo test.\n",
   "Reference: [`tout-en-un-20261016-164539.qmd`](https://github.com/bbaranoff/tests/blob/main/tout-en-un-
↪  20261016-164539.qmd)\n",
   "reaches camp after 1 s and CC ACTIVE after 3 s."
  ],
  "id": "c43"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": [
   "%%bash\n",
   "# Camp: wait for \"C3 camped normally\"\n",
   "vty() { { printf 'enable\\n%s\\n' \"$2\"; sleep 1; } | timeout --foreground 6 nc -N 127.0.0.1 \"$1\"
↪  2>/dev/null | tr -d '\\r'; }\n",
   "for i in $(seq 1 60); do vty 4247 \"show ms\" | grep -q \"C3 camped normally\" && { echo \"camp OK after
↪  $((i*2)) s\"; break; }; sleep 2; done\n",
   "vty 4247 \"show ms\" | grep -E \"camped|MM|IMSI\""
  ],
  "id": "c44"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": [
   "%%bash\n",
   "# Location update: in the mobile's log, and the subscriber in the MSC's VLR\n",
   "sed 's/\\x1b\\[[0-9;]*m//g' /tmp/c54x-pont/mobile.log | grep -ac 'LOCATION UPDATING ACCEPT'\n",
   "sed 's/\\x1b\\[[0-9;]*m//g' /tmp/c54x-pont/mobile.log | grep -aoE 'lai=[0-9]+-[0-9]+-[0-9]+' | tail -1\n",
   "{ printf 'enable\\nshow subscriber cache\\n'; sleep 1; } | timeout 6 nc -N 127.0.0.1 4254 | grep -E
↪  'IMSI|MSISDN'"
  ],
  "id": "c45"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": [
   "%%bash\n",
   "# Call 600 (echo test), wait for ACTIVE, then hang up\n",
   "vty() { { printf 'enable\\n%s\\n' \"$2\"; sleep 2; } | timeout --foreground 6 nc -N 127.0.0.1 \"$1\"
↪  2>/dev/null | tr -d '\\r'; }\n",
   "vty 4247 \"call 1 600\"\n",
   "for i in $(seq 1 15); do st=\"$(vty 4247 'show ms' | sed -n 's/^ *call control state: //p' | head -1)\"; echo
↪  \"$st\"; [ \"$st\" = ACTIVE ] && break; sleep 2; done\n",
   "grep -a \"\\[montant\\] TCH\" /tmp/c54x-pont/dsp.log | tail -2\n",
   "vty 4247 \"call 1 hangup\""
  ],
  "id": "c46"
 },
 {
  "cell_type": "markdown",
  "id": "m05",
  "metadata": {},
  "source": [
   "With `--ms-magnetite` there is no `mobile` and no VTY 4247: the C139's own UI is the phone. Watch the\n",
   "camp and the location update on its screen (`c123`, or `tools/osmo-c123-gtk.py --modele c139`) and in the\n",
```

```
   "MSC (`show subscriber cache` on 4254, as above), and dial 600 on the window's keypad. The firmware's
↪  traces\n",
   "are in `/tmp/c54x-pont/rvinterf.log`. The call needs `encryption a5 0` (step 12 variant); voice on this\n",
   "path is recommended with `--gapk` (state on 2026-10-22: speech both ways, downlink at about 27 frames/s).\n"
  ]
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "## 14. Full test ladder (optional)\n",
   "\n",
   "The end-to-end suite climbs from core to voice and says how far it got. See\n",
   "[README-tests](../doc/README-tests.md)."
  ],
  "id": "c47"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": [
   "%%bash\n",
   ". /opt/GSM/.notebook.env\n",
   "cd \"$ROOT\"; [ -d tests ] || git clone -q https://github.com/bbaranoff/tests\n",
   "cd tests && $SUDO ./banc-max.sh --dsp --continue | tail -40"
  ],
  "id": "c48"
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "## 15. Stop\n",
   "\n",
   "`./start.sh --stop` stops the stack **and** the DSP bench, without repeating `--dsp`.\n",
   "`c54x_exe/run.sh --stop` alone stops only the phone chain and purges `/dev/shm/calypso_*`. The rssi\n",
   "application (instance 4, `banc-rssi`) is left running on purpose: stop it with `banc-rssi --stop`\n",
   "(or `banc-rssi-stop`).\n",
   "\n",
   "[2026-10-23] Two safeguards against losing the network in the middle of a run:\n",
   "\n",
  "- **BTS watchdog.** osmo-bts-trx exits on \"PC clock skew too high\" when the bridge's clock, paced by the\n",
   "  DSP, stalls more than 50 frames. `qosmo/run_modules/60-bts.sh` now runs BTS#0 inside a loop that\n",
   "  restarts it (`BTS_GARDIEN=1`, the default; `BTS_GARDIEN_PAUSE` seconds between restarts, 3 by default;\n",
   "  `BTS_GARDIEN=0` for the old single start). Each restart leaves a `[gardien bts]` line in the BTS log.\n",
   "- **Clock pause cap.** `pont/trx.py` no longer asks the BTS for a pause close to that 50-frame limit:\n",
   "  when the BTS is between `PONT_CLCK_PAUSE_PLAFOND` (38) and `PONT_CLCK_SKEW_ZONE` (60) frames ahead, the\n",
   "  `IND CLOCK` is capped and the rest is absorbed by the next ones (`pause bornee` in `pont.log`;\n",
   "  `PONT_CLCK_PAUSE_PLAFOND=0` restores the old behaviour).\n"
  ],
  "id": "c49"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": [
   "%%bash\n",
   ". /opt/GSM/.notebook.env\n",
   "# Before stopping: did the BTS restart, did the bridge cap a clock pause?\n",
   "sed 's/\\x1b\\[[0-9;]*m//g' /run/user/*/osmo-nitb/logs/bts.log 2>/dev/null | grep -aE 'gardien bts|clock
↪  skew' | tail -3\n",
   "grep -a 'pause bornee' /tmp/c54x-pont/pont.log 2>/dev/null | tail -1\n",
```

```
    "cd \"$ROOT/osmo-operator\" && $SUDO ./start.sh --stop\n"
   ],
   "id": "c50"
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "name": "python"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 5
}
```

3.463 /opt/GSM/osmo-operator/wiki/lancer-les-modes.ipynb

11940 octets, 349 lignes → 349 lignes

```
{
 "cells": [
  {
   "cell_type": "markdown",
   "id": "m00",
   "metadata": {},
   "source": [
    "# Running the bench: one mode per section\n",
    "\n",
    "Companion of the help page (the **?** icon of the dock, `osmo-info`, http://127.0.0.1:8080/info.html) and
↪  of\n",
    "[lancer-pas-a-pas.ipynb](lancer-pas-a-pas.ipynb) (the same bench, process by process). This notebook
↪  installs\n",
    "nothing: it assumes the bench of [installer-osmo-jusquau-dsp.ipynb](installer-osmo-jusquau-dsp.ipynb) or the
↪  ISO.\n",
    "\n",
    "Each mode has three cells: **start**, **check**, **stop**. Run one mode at a time. The phone screens are
↪  shown\n",
    "here, inline (`screen(n)`), and you can press keys (`keys(n, ...)`); the GTK window is `c123 --instance
↪  N`.\n",
    "\n",
    "| Instance | What | Window |\n",
    "|---|---|---|\n",
    "| 1 | the phone of `--bts-classique` (osmocom or Magnetite), or the TRX of `--bts` | `c123` (`c123 --instance
↪  1`) |\n",
    "| 2 | the phone of `--bts --ms` (over the air) | `c123-1` (`c123 --instance 2`) |\n",
    "| 3 | the 2nd TRX phone of `--ts 2` (TS1) | `c123-2` (`c123 --instance 3`) |\n",
    "| 4 | the `rssi` app (`banc-rssi`) | `c123-3` (`c123 --instance 4`) |\n",
    "\n",
    "The commands need root (the container runs as root; on a desktop, start Jupyter after `sudo -v`)."
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "m01",
   "metadata": {},
   "outputs": [],
   "source": [
    "# Helpers: run a command and stream it, show a phone screen, press keys, check the network.\n",
    "import os, socket, struct, subprocess, time\n",
```

```
 "from IPython.display import display\n",
 "from PIL import Image\n",
 "\n",
 "OP = \"/opt/GSM/osmo-operator\"\n",
 "ALIASES = f\"{OP}/scripts/osmo-banc-aliases.sh\"\n",
 "\n",
 "def sh(cmd, timeout=900):\n",
 "    \"\"\"bash -c with the bench aliases (banc-rssi, c123...) loaded; output streamed.\"\"\"\n",
 "    p = subprocess.Popen([\"bash\", \"-c\", f\". {ALIASES} 2>/dev/null; {cmd}\"], cwd=OP,\n",
 "                         stdout=subprocess.PIPE, stderr=subprocess.STDOUT, text=True)\n",
 "    t0 = time.time()\n",
 "    for line in p.stdout:\n",
 "        print(line, end=\"\")\n",
 "        if time.time() - t0 > timeout:\n",
 "            p.kill(); print(\"... timeout\"); break\n",
 "    return p.wait()\n",
 "\n",
 "def _lcd_path(n):\n",
 "    return \"/dev/shm/calypso_lcd\" if n == 1 else f\"/dev/shm/calypso{n}/calypso_lcd\"\n",
 "\n",
 "def screen(n=1, scale=4):\n",
 "    \"\"\"The emulated screen of instance n: C123 (ST7558, 1 bit) or C139 (colour RGB565).\"\"\"\n",
 "    b = open(_lcd_path(n), \"rb\").read()\n",
 "    w, h, pages, flags, _, seq = struct.unpack_from(\"<HHHBBI\", b, 8)\n",
 "    d = b[32:]\n",
 "    if b[:7] == b\"C139LCD\":\n",
 "        im = Image.new(\"RGB\", (w, h))\n",
 "        px = [((p >> 11 & 31) << 3, (p >> 5 & 63) << 2, (p & 31) << 3)\n",
 "              for p in struct.unpack_from(\"<%dH\" % (w * h), d)]\n",
 "        im.putdata(px)\n",
 "    else:\n",
 "        im = Image.new(\"1\", (w, h), 1)\n",
 "        for y in range(h):\n",
 "            for x in range(w):\n",
 "                if d[(y >> 3) * w + x] & (1 << (y & 7)):\n",
 "                    im.putpixel((x, y), 0)\n",
 "    display(im.resize((w * scale, h * scale), Image.NEAREST))\n",
 "\n",
 "def keys(n, *names, wait=0.6):\n",
 "    \"\"\"Press keys on instance n: '0'..'9', 'ok', 'up', 'down', 'left', 'right', 'lsb', 'rsb', 'menu',
↪ 'power'.\"\"\"\n",
 "    path = \"/tmp/calypso_keys.sock\" if n == 1 else f\"/tmp/calypso_keys{n}.sock\"\n",
 "    s = socket.socket(socket.AF_UNIX, socket.SOCK_DGRAM)\n",
 "    for k in names:\n",
 "        s.sendto(f\"press {k}\\n\".encode(), path); time.sleep(0.25)\n",
 "        s.sendto(f\"release {k}\\n\".encode(), path); time.sleep(wait)\n",
 "\n",
 "def vty(port, cmd):\n",
 "    \"\"\"One VTY command (4254 MSC, 4242 BSC, 4241 BTS, 4247 mobile).\"\"\"\n",
 "    with socket.create_connection((\"127.0.0.1\", port), timeout=5) as c:\n",
 "        c.sendall(f\"enable\\n{cmd}\\nexit\\n\".encode()); time.sleep(1.0)\n",
 "        out = b\"\"\n",
 "        c.settimeout(1.0)\n",
 "        try:\n",
 "            while chunk := c.recv(65536): out += chunk\n",
 "        except socket.timeout:\n",
 "            pass\n",
 "    return out.decode(errors=\"replace\")\n",
 "\n",
 "def subscribers():\n",
 "    \"\"\"IMSI and cell of every subscriber attached at the MSC.\"\"\"\n",
 "    lines = [l.strip() for l in vty(4254, \"show subscriber cache\").splitlines()]\n",
 "    for l in lines:\n",
 "        if l.startswith((\"IMSI:\", \"MSISDN:\", \"LAC / cell ID:\")):\n",
 "            print(l)"
]
```

```
 },
 {
  "cell_type": "markdown",
  "id": "m02",
  "metadata": {},
  "source": [
   "## 1. Classic BTS + osmocom phone (the default)\n",
   "\n",
   "`osmo-bts-trx` talks to `pont/pont_dsp.py`; the emulated Calypso runs the osmocom-bb `layer1` firmware and
↪  `mobile`,\n",
   "on the real DSP mask ROM. This is what the red phone icon starts."
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "id": "m03",
  "metadata": {},
  "outputs": [],
  "source": [
   "sh(\"./start.sh --regen --no-attach --bts-classique --ms\")"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "id": "m04",
  "metadata": {},
  "outputs": [],
  "source": [
   "# Check: BTS#0 alive, the phone camped and registered (allow ~1 min after the start)\n",
   "sh(\"pgrep -af 'osmo-bts-trx -c /etc/osmocom/osmo-bts-trx.cfg$' | head -1; tail -2 /dev/shm/pont.log\")\n",
   "subscribers()"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "id": "m05",
  "metadata": {},
  "outputs": [],
  "source": [
   "sh(\"./start.sh --stop\")"
  ]
 },
 {
  "cell_type": "markdown",
  "id": "m06",
  "metadata": {},
  "source": [
   "## 2. Classic BTS + C139 Magnetite (the real TI firmware)\n",
   "\n",
   "Same network; the phone is a Motorola C139 running FreeCalypso Magnetite (its own L2/L3 and UI). Calls need
↪  A5/0\n",
   "(with A5/1 the FACCH fails): `ENCRYPTION=\"a5 0\"` with `--regen`. `--gapk` puts the voice on the host.\n",
   "Building the firmware: `sudo ./addition.sh --magnetite` (Wine + TI TMS470)."
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "id": "m07",
  "metadata": {},
  "outputs": [],
  "source": [
   "sh('ENCRYPTION=\"a5 0\" ./start.sh --regen --no-attach --bts-classique --ms-magnetite --gapk')"
```

```
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "id": "m08",
  "metadata": {},
  "outputs": [],
  "source": [
   "# The C139 screen (colour). Its traces are in /tmp/c54x-pont/rvinterf.log.\n",
   "screen(1, scale=5)\n",
   "sh(\"grep -aE 'L1: (NP_I|RA_C|IAS_C|DDS_R)' /tmp/c54x-pont/rvinterf.log | tail -4\")\n",
   "subscribers()"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "id": "m09",
  "metadata": {},
  "outputs": [],
  "source": [
   "# Call the echo test 600 from the C139 keypad, then look at the screen\n",
   "keys(1, \"6\", \"0\", \"0\", \"ok\")\n",
   "time.sleep(8); screen(1, scale=5)"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "id": "m10",
  "metadata": {},
  "outputs": [],
  "source": [
   "sh(\"./start.sh --stop\")"
  ]
 },
 {
  "cell_type": "markdown",
  "id": "m11",
  "metadata": {},
  "source": [
   "## 3. The Calypso as the BTS radio (`--bts`)\n",
   "\n",
   "The Calypso of QEMU runs the `trx` firmware and the host `transceiver`: it **is** the radio of BTS#0. A
↪  second\n",
   "Calypso (`--ms`, instance 2) is the phone and camps on it over the air (`pont/air.py`).\n",
   "\n",
   "| `--ts` | Timeslots | State |\n",
   "|---|---|---|\n",
   "| `--ts 1` | TS0 only (CCCH + SDCCH/4) | LU, SMS, no call |\n",
   "| `--ts 2/1` | TS0 + TS1 by one Calypso (default of `--bts`) | calls |\n",
   "| `--ts 2` | TS0 and TS1 by one Calypso each (instances 1 and 3) | camps, LU; no call yet (2nd TRX not
↪  frame-aligned) |"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "id": "m12",
  "metadata": {},
  "outputs": [],
  "source": [
   "sh(\"./start.sh --regen --no-attach --bts --ms --ts 2\")"
  ]
 },
```

```
{
 "cell_type": "code",
 "execution_count": null,
 "id": "m13",
 "metadata": {},
 "outputs": [],
 "source": [
  "# The TRX (instance 1), the phone over the air (instance 2), the 2nd TRX (instance 3)\n",
  "for n in (2,):\n",
  "    print(f\"instance {n}\"); screen(n)\n",
  "sh(\"tail -3 /tmp/c54x-pont2/air.log 2>/dev/null; tail -3 /tmp/c54x-pont/transceiver.log 2>/dev/null\")\n",
  "subscribers()"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "id": "m14",
 "metadata": {},
 "outputs": [],
 "source": [
  "sh(\"./start.sh --stop\")"
 ]
},
{
 "cell_type": "markdown",
 "id": "m15",
 "metadata": {},
 "source": [
  "## 4. `banc-rssi`: the rssi app on a 4th Calypso\n",
  "\n",
  "The osmocom-bb `rssi` app (spectrum, sync, per-timeslot levels) runs on instance 4. In `--bts-classique` the
↪  pont\n",
  "copies every downlink (and uplink) burst to it (UDP 6762, `PONT_DSP_COPIES`); with `--bts` it listens to the
↪  air.\n",
  "The realistic air (`CALYPSO_AIR_NIVEAUX=1`) plays the cell only on its ARFCN, at -60 dBm downlink and -75 dBm
↪  uplink,\n",
  "with noise elsewhere. Start a network first (section 1 or 2), then:"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "id": "m16",
 "metadata": {},
 "outputs": [],
 "source": [
  "sh(\"banc-rssi\")"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "id": "m17",
 "metadata": {},
 "outputs": [],
 "source": [
  "# Spectrum view: ARFCN 1 (the cell) should read about -60 dBm, the next ARFCN the noise floor\n",
  "time.sleep(20); screen(4)\n",
  "keys(4, \"right\"); time.sleep(3); screen(4)     # ARFCN 2: noise\n",
  "keys(4, \"left\")"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
```

```
   "id": "m18",
   "metadata": {},
   "outputs": [],
   "source": [
    "# Sync on the cell (OK), then the per-timeslot view (Up), then the uplink (right soft key)\n",
    "keys(4, \"ok\"); time.sleep(40); screen(4)       # BSIC 7/7, MCC/MNC/LAC, cell id\n",
    "keys(4, \"up\"); time.sleep(5); screen(4)        # 8 timeslots, downlink: all at about -60\n",
    "keys(4, \"rsb\"); time.sleep(5); screen(4)       # uplink: the timeslot of the phone (when it transmits)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "m19",
   "metadata": {},
   "outputs": [],
   "source": [
    "sh(\"banc-rssi --stop\")"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "m20",
   "metadata": {},
   "source": [
    "## 5. Windows, consoles, stop\n",
    "\n",
    "- Phone windows: `c123 --instance N` (`c123`, `c123-1`, `c123-2`, `c123-3`, `c123-tous` for all of them).\n",
    "- Consoles: `sudo tmux attach -t calypso`; VTY: MSC 4254, BSC 4242, BTS 4241, mobile 4247.\n",
    "- DSP bench logs: `/tmp/c54x-pont*/*.log`; `cd /opt/GSM/c54x_exe && ./run.sh --logs`.\n",
    "- BTS#0 runs under a watchdog (`BTS_GARDIEN=1`): a \"PC clock skew too high\" exit restarts it after 3 s.\n",
    "- Nothing telecom starts at boot (`services/80-osmo-operator.preset`): `systemctl enable osmo-banc` to change
↪  that."
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "m21",
   "metadata": {},
   "outputs": [],
   "source": [
    "sh(\"./start.sh --stop; banc-rssi --stop\")"
   ]
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "name": "python"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 5
}
```

3.464 /opt/GSM/osmo-operator/wiki/lancer-pas-a-pas.ipynb

35898 octets, 508 lignes → 508 lignes

```
{
 "cells": [
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": "# Launching the bench step by step, up to a call on the Calypso DSP\n\nThis notebook starts the
 ↪  bench **one process at a time** and checks each one before adding the next:\nfirst the Osmocom network (STP,
 ↪  HLR, MGW, MSC, BSC, SIP side, BTS), then the emulated phone (the C54x DSP,\nQEMU with the ARM and the layer 1
 ↪  firmware, osmocon, mobile, the bridge), and finally camp, location\nupdate and a call. It runs the same
 ↪  commands as `start.sh` and `c54x_exe/run.sh`, written out by\nhand, so you can see what each component does
 ↪  and where a fault comes from.\n\n**It installs nothing.** Run it where the bench is already built: the
 ↪  `osmocom-run` image from\n`build.sh`, the ISO, or a machine after the [install
 ↪  notebook](installer-osmo-jusquau-dsp.ipynb).\n\n| | |\n|---|---|\n| Needs | everything under `/opt/GSM`
 ↪  (osmo-operator, c54x_exe, qosmo, osmocom-bb, firmware, DSP ROM) and the Osmocom daemons in the `PATH` |\n|
 ↪  Runs as | root (or a kernel started after `sudo -v`) |\n| Logs | `↩
/tmp/c54x-pont/<name>.log`, one per process, pid in `<name>.pid` |\n| Time | a few minutes |\n\nRules: one process
 ↪  = one cell, followed by its **check**. A check exits with an error if what it waits\nfor does not show up, and
 ↪  prints the end of the log. Do not move on after a failed check: fix it first,\nthe next steps depend on it.
 ↪  Stop everything with the last cell before starting over.\n\n> [2026-10-23] The launcher is now `./start.sh`;
 ↪  `start-native.sh` is a symlink to it and still works.\n\nIn the `osmocom-run` image:\n\n```bash\ndocker run
 ↪  -it --rm --privileged --shm-size=2g -p 8888:8888 --entrypoint bash osmocom-run:latest\npython3 -m venv /venv
 ↪  && /venv/bin/pip install -q jupyterlab\ncd /opt/GSM/osmo-operator/wiki && /venv/bin/jupyter lab --ip=0.0.0.0
 ↪  --allow-root --no-browser \\\n  --LabApp.default_url=/lab/tree/lancer-pas-a-pas.ipynb\n```",
   "id": "c00"
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": "## 0. Helpers, cleanup, configs\n\nThe helpers go into `/tmp/c54x-pont/lib.sh`, sourced by every
 ↪  cell (`%%bash` cells do not share their\nenvironment):\n\n- `start NAME CMD…` starts a process detached from
 ↪  the notebook, log in `NAME.log`, pid in `NAME.pid`;\n- `wait_log FILE PATTERN [SECONDS]` waits for a line,
 ↪  prints it, or prints the end of the log and fails;\n- `wait_tcp PORT [SECONDS]` / `wait_udp PORT [SECONDS]`
 ↪  wait for a listening port;\n- `vty PORT COMMAND` sends one command to an Osmocom VTY.\n\nThe three `CALYPSO_*`
 ↪  variables are the DSP-mode settings `start.sh` exports before handing over\nto `c54x_exe/run.sh`; every
 ↪  phone-side process inherits them.",
   "id": "c01"
  },
  {
   "cell_type": "code",
   "metadata": {},
   "execution_count": null,
   "outputs": [],
   "source": "%%bash\nmkdir -p /tmp/c54x-pont\ncat > /tmp/c54x-pont/lib.sh
 ↪  <<'EOF'\nL=/tmp/c54x-pont\nROOT=/opt/GSM\nO=$ROOT/osmo-operator\nexport CALYPSO_BSP_STREAM=1        # one TS0
 ↪  burst per frame, in FN order\nexport CALYPSO_RHEA_DMA_XFER=1     # without it: no SB, no BCCH, no SI\nexport
 ↪  CALYPSO_PONT_LOCKSTEP=1     # QEMU only advances the frame once the DSP is done\n[ \"$(id -u)\" = 0 ] &&
 ↪  SUDO=\"\" || SUDO=\"sudo -n\"\n\nstart() {   # start NAME CMD...\n  local n=$1; shift\n  setsid nohup \"$@\" >
 ↪  \"$L/$n.log\" 2>&1 < /dev/null &\n  echo $! > \"$L/$n.pid\"\n  echo \"started $n (pid $!)\"\n}\nalive() { kill
 ↪  -0 \"$(cat \"$L/$1.pid\" 2>/dev/null)\" 2>/dev/null; }\nwait_log() {   # wait_log FILE PATTERN [SECONDS]\n
 ↪  local f=$1 p=$2 t=${3:-10} i\n  for i in $(seq $((t*2))); do\n    grep -aq -- \"$p\" \"$f\" 2>/dev/null && {
 ↪  echo \"ok      $(grep -a -- \"$p\" \"$f\" | tail -1)\"; return 0; }\n    sleep 0.5\n  done\n  echo \"TIMEOUT
 ↪  waiting for \\\"$p\\\" in $f\"; tail -15 \"$f\" 2>/de↩
v/null; return 1\n}\nwait_tcp() { local i; for i in $(seq $((${2:-10}*2))); do ss -ltnH \"( sport = :$1 )\" | grep
 ↪  -q . && { echo \"ok      tcp $1 listening\"; return 0; }; sleep 0.5; done; echo \"TIMEOUT tcp $1\"; return 1;
 ↪  }\nwait_udp() { local i; for i in $(seq $((${2:-10}*2))); do ss -ulnH \"( sport = :$1 )\" | grep -q . && {
 ↪  echo \"ok      udp $1 bound\"; return 0; }; sleep 0.5; done; echo \"TIMEOUT udp $1\"; return 1; }\nvty() { {
 ↪  printf 'enable\\n%s\\n' \"$2\"; sleep 1; } | timeout --foreground 6 nc -N 127.0.0.1 \"$1\" 2>/dev/null | tr -d
 ↪  '\\r'; }\nEOF\necho \"helpers written to /tmp/c54x-pont/lib.sh\"",
   "id": "c02"
  },
  {
   "cell_type": "code",
   "metadata": {},
   "execution_count": null,
   "outputs": [],
```

```
   "source": "%%bash\n. /tmp/c54x-pont/lib.sh\n# Check: the pieces this notebook runs, and nothing from a previous
 ↪  session still running\nfail=0\nfor f in \"$ROOT/c54x_exe/c54x_exe\" \"$ROOT/c54x_exe/mobile_pont.cfg\"
 ↪  \"$ROOT/qosmo/build/qemu-system-arm\" \\\n         \"$ROOT/osmocom-bb/src/host/osmocon/osmocon\"
 ↪  \"$ROOT/firmware/board/compal_e88/layer1.highram.elf\" \\\n
 ↪  \"$ROOT/firmware/board/compal_e88/layer1.highram.bin\" \"$O/pont/pont_dsp.py\"; do\n  [ -e \"$f\" ] && echo
 ↪  \"ok      $f\" || { echo \"MISSING $f\"; fail=1; }\ndone\nfor b in osmo-stp osmo-hlr osmo-mgw osmo-msc
 ↪  osmo-bsc osmo-bts-trx osmo-sip-connector asterisk mobile nc socat; do\n  command -v \"$b\" >/dev/null && echo
 ↪  \"ok      $b\" || { echo \"MISSING $b\"; fail=1; }\ndone\nrunning=$(pgrep -a -f
 ↪  'osmo-(stp|hlr|mgw|msc|bsc|bts-trx|sip-connector)|c54x_exe|qemu-system-arm|osmocon|mobile -c|pont_dsp' ||
 ↪  true)\n[ -n \"$running\" ] && { echo \"ALREADY RUNNING (run the Stop cell at the end first):\"; echo \"$runnin↩
g\"; fail=1; }\nexit $fail",
   "id": "c03"
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": "The Osmocom configs in `/etc/osmocom` are templates: markers such as `__MCC__`, `__ARFCN__`,
 ↪  `__IMSI__`\nare filled from `globals.conf` by `./start.sh --regen`. A daemon started on an unfilled
 ↪  config\nrefuses it, so check first.",
   "id": "c04"
  },
  {
   "cell_type": "code",
   "metadata": {},
   "execution_count": null,
   "outputs": [],
   "source": "%%bash\n. /tmp/c54x-pont/lib.sh\n# Check: no template marker left in the configs this notebook
 ↪  uses\nleft=$(grep -l '__[A-Z0-9_]*__' /etc/osmocom/osmo-{stp,hlr,mgw,msc,bsc,bts-trx,sip-connector}.cfg \\\n
 ↪  \"$ROOT/c54x_exe/mobile_pont.cfg\" 2>/dev/null || true)\nif [ -n \"$left\" ]; then\n  echo \"unfilled
 ↪  templates:\"; echo \"$left\"\n  echo \"fill them first: cd $O && ./start.sh --regen, then ./start.sh
 ↪  --stop\"\n  exit 1\nfi\necho \"ok      all configs filled\"\ngrep -E '^(MCC|MNC|BAND|ARFCN|BSIC|LAC)='
 ↪  \"$O/globals.conf\"",
   "id": "c05"
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": "---\n# Part 1: the network\n\nBench order: STP, HLR, MGW, MSC, BSC, then the SIP side for calls, then
 ↪  the BTS. Each daemon gets its VTY\nchecked before the next one starts. VTY ports are the Osmocom defaults: STP
 ↪  4239, HLR 4258, MGW 4243,\nMSC 4254, BSC 4242, SIP connector 4255, BTS 4241.\n\n## 1. osmo-stp, the SS7
 ↪  signalling point\n\nEvery link of the core (BSC↔MSC over SCCP/M3UA) goes through it, so it starts first.",
   "id": "c06"
  },
  {
   "cell_type": "code",
   "metadata": {},
   "execution_count": null,
   "outputs": [],
   "source": "%%bash\n. /tmp/c54x-pont/lib.sh\nstart stp osmo-stp -c /etc/osmocom/osmo-stp.cfg",
   "id": "c07"
  },
  {
   "cell_type": "code",
   "metadata": {},
   "execution_count": null,
   "outputs": [],
   "source": "%%bash\n. /tmp/c54x-pont/lib.sh\n# Check\nwait_tcp 4239 && vty 4239 \"show cs7 instance 0 asp\"",
   "id": "c08"
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": "## 2. osmo-hlr, the subscriber database\n\nThe database path is set in `osmo-hlr.cfg`
 ↪  (`/var/lib/osmocom/hlr.db`).",
   "id": "c09"
  },
```

```
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": "%%bash\n. /tmp/c54x-pont/lib.sh\nmkdir -p /var/lib/osmocom\nstart hlr osmo-hlr -c
↪  /etc/osmocom/osmo-hlr.cfg",
  "id": "c10"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": "%%bash\n. /tmp/c54x-pont/lib.sh\n# Check\nwait_tcp 4258 && vty 4258 \"show subscribers all\" | tail
↪  -n +3",
  "id": "c11"
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": "### 2b. The phone's subscriber\n\nThe emulated phone uses the test SIM of `mobile_pont.cfg`
↪  (`test-sim`: `imsi`, `ki`). Its IMSI must be in\nthe HLR with the same Ki, or the location update is rejected.
↪  This cell reads them from the config and\ncreates the subscriber if it is missing. The MSISDN is up to you.",
  "id": "c12"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": "%%bash\n. /tmp/c54x-pont/lib.sh\ncfg=\"$ROOT/c54x_exe/mobile_pont.cfg\"\nIMSI=$(awk '/^ *imsi
↪  [0-9]+/ {print $2; exit}' \"$cfg\")\nALGO=$(awk '/^ *ki / {print $2; exit}' \"$cfg\")\nKI=$(awk '/^ *ki /
↪  {s=\"\"; for (i=3; i<=NF; i++) s=s $i; print s; exit}' \"$cfg\")\nMSISDN=1001\necho \"IMSI=$IMSI ALGO=$ALGO
↪  KI=$KI\"\n[ -n \"$IMSI\" ] && [ -n \"$KI\" ] || { echo \"no test-sim imsi/ki found in $cfg\"; exit 1; }\ncase
↪  \"$ALGO\" in comp128) HALGO=comp128v1 ;; xor) HALGO=xor-2g ;; *) HALGO=$ALGO ;; esac\nif vty 4258 \"show
↪  subscriber imsi $IMSI\" | grep -q \"IMSI: $IMSI\"; then\n  echo \"subscriber already present\"\nelse\n  {
↪  printf 'enable\\n'\n    printf 'subscriber imsi %s create\\n' \"$IMSI\"\n    printf 'subscriber imsi %s update
↪  msisdn %s\\n' \"$IMSI\" \"$MSISDN\"\n    printf 'subscriber imsi %s update aud2g %s ki %s\\n' \"$IMSI\"
↪  \"$HALGO\" \"$KI\"\n    sleep 1; } | timeout 6 nc -N 127.0.0.1 4258 | tr -d '\\r' | grep -iE 'error|%' ||
↪  true\nfi",
  "id": "c13"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": "%%bash\n. /tmp/c54x-pont/lib.sh\n# Check: the subscriber, with its 2G key\nIMSI=$(awk '/^ *imsi
↪  [0-9]+/ {print $2; exit}' \"$ROOT/c54x_exe/mobile_pont.cfg\")\nout=$(vty 4258 \"show subscriber imsi
↪  $IMSI\"); echo \"$out\" | tail -n +3\necho \"$out\" | grep -q \"IMSI: $IMSI\" && echo \"$out\" | grep -qi
↪  \"2G\" || { echo \"subscriber or its 2G key missing\"; exit 1; }",
  "id": "c14"
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": "## 3. osmo-mgw, the media gateway\n\nCarries the voice (RTP) between the BTS and the MSC side.",
  "id": "c15"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": "%%bash\n. /tmp/c54x-pont/lib.sh\nstart mgw osmo-mgw -c /etc/osmocom/osmo-mgw.cfg",
```

```
  "id": "c16"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": "%%bash\n. /tmp/c54x-pont/lib.sh\n# Check\nwait_tcp 4243 && vty 4243 \"show mgcp\" | head -8",
  "id": "c17"
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": "## 4. osmo-msc, the switching centre\n\nIt connects to the STP (SCCP) and to the HLR (GSUP). Its
↪  calls go out through an **external MNCC socket**\n(`mncc external /tmp/msc_mncc` in `osmo-msc.cfg`): without
↪  the SIP connector of step 6, the network\nworks for camp, LU and SMS, but a call cannot be set up.",
  "id": "c18"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": "%%bash\n. /tmp/c54x-pont/lib.sh\nstart msc osmo-msc -c /etc/osmocom/osmo-msc.cfg",
  "id": "c19"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": "%%bash\n. /tmp/c54x-pont/lib.sh\n# Check: VTY, the MNCC socket, and the MSC's SCCP user on the
↪  STP\nwait_tcp 4254 && test -S /tmp/msc_mncc && echo \"ok      /tmp/msc_mncc\"\nvty 4239 \"show cs7 instance 0
↪  asp\"",
  "id": "c20"
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": "## 5. osmo-bsc, the base station controller\n\nConnects to the MSC through the STP, and waits for
↪  the BTS (OML/RSL) on Abis.",
  "id": "c21"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": "%%bash\n. /tmp/c54x-pont/lib.sh\nstart bsc osmo-bsc -c /etc/osmocom/osmo-bsc.cfg",
  "id": "c22"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": "%%bash\n. /tmp/c54x-pont/lib.sh\n# Check: VTY, and BTS 0 declared (OML not connected yet: the BTS
↪  comes at step 7)\nwait_tcp 4242 && vty 4242 \"show bts 0\" | grep -E \"BTS 0|OML Link\"",
  "id": "c23"
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": "## 6. Asterisk and osmo-sip-connector, for calls\n\n`osmo-sip-connector` takes the MSC's MNCC socket
↪  and turns the call into SIP towards Asterisk, where\n`extensions.conf` routes number 600 to the echo test.
↪  Skip this step if you only want camp, LU and SMS.",
```

```
  "id": "c24"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": "%%bash\n. /tmp/c54x-pont/lib.sh\nls /etc/asterisk/extensions.conf /etc/asterisk/pjsip.conf || {
↪  echo \"Asterisk configs missing\"; exit 1; }\nmkdir -p /var/run/asterisk /var/log/asterisk
↪  /var/lib/asterisk\nasterisk -rx \"core show version\" >/dev/null 2>&1 || asterisk\nstart sip
↪  osmo-sip-connector -c /etc/osmocom/osmo-sip-connector.cfg",
  "id": "c25"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": "%%bash\n. /tmp/c54x-pont/lib.sh\n# Check: Asterisk answers, number 600 exists, the connector holds
↪  the MNCC socket.\n# Asterisk forks and needs a few seconds before its control socket answers.\nfor i in $(seq
↪  1 20); do asterisk -rx \"core show version\" 2>/dev/null && break; sleep 0.5; done\nasterisk -rx \"core show
↪  version\" >/dev/null 2>&1 || { echo \"Asterisk not answering after 10 s\"; tail -15
↪  /var/log/asterisk/messages* 2>/dev/null; exit 1; }\nasterisk -rx \"dialplan show 600@default\" 2>/dev/null |
↪  grep -E \"600|Echo\" || asterisk -rx \"dialplan show\" | grep -m3 600\nwait_tcp 4255 && grep -ai \"mncc\"
↪  \"$L/sip.log\" | tail -3",
  "id": "c26"
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": "## 7. osmo-bts-trx, the base station\n\nIt connects to the BSC and starts sending its TRXC commands
↪  to UDP 5701, where **nobody answers yet**:\nthe bridge (step 12) will. That is expected.\n\nStarted by hand
↪  like this, osmo-bts-trx runs **once**: if it exits (\"PC clock skew too high\" when the\nDSP stalls the
↪  bridge's clock), restart this cell. `./start.sh` runs it inside a watchdog loop instead\n(see the Stop
↪  section).",
  "id": "c27"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": "%%bash\n. /tmp/c54x-pont/lib.sh\nstart bts osmo-bts-trx -c /etc/osmocom/osmo-bts-trx.cfg",
  "id": "c28"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": "%%bash\n. /tmp/c54x-pont/lib.sh\n# Check: VTY, the clock/data ports, and OML seen by the
↪  BSC\nwait_tcp 4241 || exit 1\nss -ulnp | grep -E ':580[0-2] ' || { echo \"osmo-bts-trx not bound on
↪  5800-5802\"; exit 1; }\nsleep 2; vty 4242 \"show bts 0\" | grep -E \"OML Link\"",
  "id": "c29"
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": "---\n# Part 2: the emulated phone\n\nFive processes, in this order: the C54x DSP, QEMU (ARM + layer
↪  1 firmware), osmocon (loads the firmware\nover the serial line), mobile (layers 2/3), and the bridge between
↪  the BTS and the DSP.\n\n## 8. c54x_exe, the DSP\n\nIt runs TI's mask ROM, publishes the shared API RAM
↪  (`/dev/shm/calypso_api_ram`) and its socket\n(`/tmp/calypso_dsp.sock`), then waits for the ARM. It also
↪  listens on UDP 6702 for bursts (BSP).",
  "id": "c30"
 },
 {
```

```
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": "%%bash\n. /tmp/c54x-pont/lib.sh\nrm -f /dev/shm/calypso_api_ram /tmp/calypso_dsp.sock
↪  /dev/shm/calypso_tch_cfg\ncd \"$ROOT/c54x_exe\"\nstart dsp env CALYPSO_IQDUMP_FCCH=1
↪  CALYPSO_BSP_ATTENTE_MS=40 \\\n  ./c54x_exe --arm --insns 120000 --iq none --amp 30000 -v",
  "id": "c31"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": "%%bash\n. /tmp/c54x-pont/lib.sh\n# Check (\"en attente de l'ARM\" = \"waiting for the
↪  ARM\")\nwait_log \"$L/dsp.log\" \"en attente de l'ARM\" 5 && test -S /tmp/calypso_dsp.sock && echo \"ok
↪  /tmp/calypso_dsp.sock\" && wait_udp 6702",
  "id": "c32"
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": "## 9. QEMU (qosmo): the ARM and the layer 1 firmware\n\n`-M calypso` emulates the Compal E88 board.
↪  `CALYPSO_DSP_EXTERN=1` hands the DSP to `c54x_exe` instead of\nQEMU's built-in model. The first serial port
↪  becomes a pty: that is the modem line osmocon will talk to.\nThe gdb stub listens on 1234.",
  "id": "c33"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": "%%bash\n. /tmp/c54x-pont/lib.sh\nrm -f /tmp/qemu-monitor-pont.sock\nstart qemu env
↪  CALYPSO_PONT_RETRY_DIV=64 CALYPSO_DSP_EXTERN=1 \\\n  \"$ROOT/qosmo/build/qemu-system-arm\" -M calypso -cpu
↪  arm946 -display none -parallel none \\\n    -serial pty -serial pty -monitor
↪  unix:/tmp/qemu-monitor-pont.sock,server,nowait \\\n    -gdb tcp:127.0.0.1:1234 \\\n    -kernel
↪  \"$ROOT/firmware/board/compal_e88/layer1.highram.elf\"",
  "id": "c34"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": "%%bash\n. /tmp/c54x-pont/lib.sh\n# Check: the modem pty, the DSP bridge on the QEMU side (\"API RAM
↪  partagee\" = \"shared API RAM\"), the VM running\nwait_log \"$L/qemu.log\" \"label serial0\" 10 || exit
↪  1\nwait_log \"$L/qemu.log\" \"pont DSP : API RAM partagee\" 10 || exit 1\nsed -n 's/.*redirected to
↪  \\(\\/dev\\/pts\\/[0-9]*\\) (label serial0).*/\\1/p' \"$L/qemu.log\" | head -1 > \"$L/modem.pty\"\necho
↪  \"modem pty: $(cat \"$L/modem.pty\")\"\nprintf 'info status\\n' | socat -
↪  UNIX-CONNECT:/tmp/qemu-monitor-pont.sock | grep \"VM status\"",
  "id": "c35"
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": "## 10. osmocon: load the firmware, relay L1CTL\n\n`-m romload` speaks the Calypso boot ROM protocol
↪  over the modem pty and uploads `layer1.highram.bin`.\nThen osmocon relays layer 1 control messages (L1CTL) on
↪  `/tmp/osmocom_l2`, where `mobile` will connect.",
  "id": "c36"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
```

```
  "source": "%%bash\n. /tmp/c54x-pont/lib.sh\nrm -f /tmp/osmocom_l2\nstart osmocon stdbuf -oL -eL
↪  \"$ROOT/osmocom-bb/src/host/osmocon/osmocon\" -m romload -i 100 \\\n  -p \"$(cat \"$L/modem.pty\")\" -s
↪  /tmp/osmocom_l2 \\\n  \"$ROOT/firmware/board/compal_e88/layer1.highram.bin\"",
  "id": "c37"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": "%%bash\n. /tmp/c54x-pont/lib.sh\n# Check: the upload finished and the L1CTL socket exists; the DSP
↪  has booted on the firmware's reset\nwait_log \"$L/osmocon.log\" \"your code is running now\" 30 || exit
↪  1\ntest -S /tmp/osmocom_l2 && echo \"ok      /tmp/osmocom_l2\"\ngrep -a \"RESET_DSP relache\" \"$L/qemu.log\"
↪  | tail -1\ngrep -a \"DSP boote\" \"$L/dsp.log\" | tail -1",
  "id": "c38"
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": "## 11. mobile: layers 2 and 3\n\nThe osmocom-bb `mobile` application with `mobile_pont.cfg`: test
↪  SIM, cell lock on the bench ARFCN, VTY.\nIt immediately asks layer 1 for a power scan and a sync on the cell
↪  (`L1CTL_FBSB_REQ`). As long as the\nbridge is not running there is no downlink, so `FBSB RESP: result=255` in
↪  a loop is **expected** here.",
  "id": "c39"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": "%%bash\n. /tmp/c54x-pont/lib.sh\nMS_VTY=$(awk '/^ *bind 127\\.0\\.0\\.1 [0-9]+/ {print $3; exit}'
↪  \"$ROOT/c54x_exe/mobile_pont.cfg\")\necho \"mobile VTY port from mobile_pont.cfg: $MS_VTY\"; echo \"$MS_VTY\"
↪  > \"$L/ms_vty\"\nss -ltnH \"( sport = :$MS_VTY )\" | grep -q . && { echo \"port $MS_VTY already taken\"; exit
↪  1; }\nrm -f /tmp/ms_data\nstart mobile env L23_SYNC_RETRIES_SELECTION=8 stdbuf -oL mobile -c
↪  \"$ROOT/c54x_exe/mobile_pont.cfg\"",
  "id": "c40"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": "%%bash\n. /tmp/c54x-pont/lib.sh\n# Check: still alive, VTY up, layer 1 asked to sync on the
↪  cell\nsleep 2; alive mobile && echo \"ok      mobile alive\" || { tail -15 \"$L/mobile.log\"; exit 1;
↪  }\nwait_tcp \"$(cat \"$L/ms_vty\")\" 5\nwait_log \"$L/osmocon.log\" \"L1CTL_FBSB_REQ\" 5",
  "id": "c41"
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": "## 12. pont_dsp.py: the bridge between the BTS and the DSP\n\nThis is the radio link. On one side it
↪  speaks TRX to osmo-bts-trx (ports 5700–5702: clock, control,\nbursts); on the other it feeds the DSP's burst
↪  port (6702). Downlink bursts go to the DSP, uplink bursts\n(RACH, SDCCH, TCH) come back to the BTS.
↪  `ARFCN`/`BSIC` must match the BSC config (514 / 7 on the bench).\n`--no-record` skips the I/Q recording to
↪  `/root/record.cfile`; remove it if you want the capture.",
  "id": "c42"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": "%%bash\n. /tmp/c54x-pont/lib.sh\ncd \"$O\"\nstart pont env PONT_BSC_CFG=/etc/osmocom/osmo-bsc.cfg
↪  PONT_ARFCN=514 PONT_BSIC=7 \\\n  python3 pont/pont_dsp.py --dsp-port 6702 --no-record",
  "id": "c43"
 },
```

```
  {
   "cell_type": "code",
   "metadata": {},
   "execution_count": null,
   "outputs": [],
   "source": "%%bash\n. /tmp/c54x-pont/lib.sh\n# Check: ports, the BTS clock armed (\"horloge armee\" = \"clock
↪  armed\"), the BTS's TRXC commands, bursts flowing\nwait_log \"$L/pont.log\" \"pont TRX : ports\" 10 || exit
↪  1\nwait_log \"$L/pont.log\" \"horloge armee\" 10\ngrep -a \"CTRL \" \"$L/pont.log\" | head -5\nsleep 3; grep
↪  -a \"STATS fn=\" \"$L/pont.log\" | tail -2",
   "id": "c44"
  },
  {
   "cell_type": "markdown",
   "id": "k01",
   "metadata": {},
   "source": "### 12a. What you just built: the classic BTS (`--bts-classique`)\n\n[2026-10-21] Steps 7 to 12 are
↪  the **classic BTS** set-up: osmo-bts-trx is the base station and\n`pont_dsp.py` is its radio, feeding the
↪  emulated phone's DSP. In one command it is\n`./start.sh --bts-classique --ms` (`--bts-classic` is the same
↪  option; it is also what plain `--dsp`\nstarts). Two things the bridge does in this set-up since
↪  2026-10-23:\n\n- it copies every downlink burst (and, marked, every uplink burst) to the UDP ports of\n
↪  `PONT_DSP_COPIES` (6762 by default, set by `pont_dsp.py`: the BSP of instance 4, the rssi application of\n
↪  the last section). A port nobody listens on changes nothing for the mobile; `PONT_DSP_COPIES=` turns the\n
↪  copy off, `PONT_DSP_COPIES_UL=0` keeps the downlink only;\n- it caps the clock pause it asks of the BTS
↪  (`PONT_CLCK_PAUSE_PLAFOND`, 38 frames), so a slow DSP no\n  longer pushes osmo-bts-trx past its 50-frame limit
↪  (\"PC clock skew too high\").\n\nThe same↩
 bridge can carry the **C139 Magnetite** instead of the osmocom-bb mobile (steps 10-11 are then\nreplaced by
↪  `rvinterf`, the UI is in the firmware): `ENCRYPTION=\"a5 0\" ./start.sh --bts-classique\n--ms-magnetite
↪  --gapk` (with A5/1 the FACCH fails and calls do not set up; `--gapk` puts the voice codec in\nthe bridge,
↪  `pont/voix_hote.py`). See [doc/magnetite.md](../doc/magnetite.md).\n"
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "k02",
   "metadata": {},
   "outputs": [],
   "source": "%%bash\n. /tmp/c54x-pont/lib.sh\n# Check: the bridge's copy port and clock cap as it sees them, and
↪  whether it already had to cap a pause\ntr '\\0' '\\n' < /proc/\"$(cat \"$L/pont.pid\")\"/environ | grep -E
↪  '^PONT_(DSP_COPIES|CLCK_PAUSE_PLAFOND|CLCK_SKEW_ZONE)=' \\\n  || echo \"defaults: PONT_DSP_COPIES=6762 (set
↪  inside pont_dsp.py), PONT_CLCK_PAUSE_PLAFOND=38, PONT_CLCK_SKEW_ZONE=60\"\ngrep -a 'pause bornee'
↪  \"$L/pont.log\" | tail -1 || true\n"
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": "---\n## 12b. Variant: the phone as the BTS transceiver (`--bts`, alias `--osmo-trx`), and a second
↪  phone over the air\n\n[2026-10-20] Instead of mobile + bridge (steps 11-12), the emulated Calypso can **be the
↪  transceiver of\nBTS#0**: QEMU boots `trx.highram.elf` (osmocom-bb `jolly/testing`, `trx` application) and the
↪  host\n`transceiver` program replaces both `mobile` and `pont_dsp.py`: it talks L1CTL to the firmware and
↪  TRXD\n(5700-5702) to osmo-bts-trx, `--no-sync` (no cell to listen to, BTS mode right after reset), RX
↪  mask\n`0x01`, TX mask `0x01` (one TX window at TPU slot 5 = downlink TS0). The DSP extension code of the
↪  firmware\nturns the ROM's RACH transmit task into FB / SB / dummy / normal bursts, which `c54x_exe` publishes
↪  in\n`/dev/shm/calypso_tx_rom`.\n\nEverything is driven by `c54x_exe/run.sh` with `OSMO_TRX=1` (alias
↪  `banc-trx`; in one command, network\nincluded: `./start.sh --bts --ms`). Stop steps 8-12 first:\nthe DSP, QEMU
↪  and osmocon are restarted with t↩
he other firmware. Note that osmo-bts-trx **exits** when its\ntransceiver disappears (\"No more clock from
↪  transceiver\"); `run.sh` relaunches it (step 4b) when BTS#0 is\nabsent.\n\nThen a **second emulated phone**
↪  (`INSTANCE=2`, alias `banc-mobile`: its own `/dev/shm` through a private\nmount namespace, `/tmp/c54x-pont2`,
↪  L1CTL `/tmp/osmocom_l2_2`, BSP UDP 6712, mobile VTY 4347) camps on it:\n`pont/air.py` relays the TRX's
↪  published bursts to the mobile's BSP and the mobile's bursts back.\n\nState on 2026-10-20 evening: the second
↪  phone camps, sends its RACH, gets its IMMEDIATE ASSIGNMENT on the\nSDCCH/4 of TS0 and completes the **location
↪  update** (`LOCATION UPDATING ACCEPT`, TMSI allocated). Two\nprostheses in `c54x_exe` make up for what the
↪  firmware's DSP extension code gets wrong on the emulated core:\nthe SCH is re-encoded before publishing
↪  (`MONTANT_ROM_TX_SB_RECODE`) and the exact bits of each uplink\nnormal burst are written into the extension's
↪  result page (`CALYPSO_BSP_TRX_U↩
```

```
L`). The uplink is aligned on\nthe firmware's clock from the SB it emits (`CALYPSO_BSP_TRX`, `BTS_RX_LAG 3` /
 ↪  delta -2). SMS and calls\nover this path are not exercised yet. See `wiki/Reference.md` 2.5 and 2.7.",
   "id": "x01"
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "%%bash\n",
    ". /tmp/c54x-pont/lib.sh\n",
    "for n in pont mobile osmocon gdb qemu dsp; do p=$(cat \"$L/$n.pid\" 2>/dev/null) && kill \"$p\" 2>/dev/null
 ↪  && echo \"stopped $n\"; done\n",
    "sleep 1; rm -f /dev/shm/calypso_api_ram /tmp/calypso_dsp.sock /tmp/osmocom_l2 /tmp/qemu-monitor-pont.sock
 ↪  \"$L\"/{pont,mobile,osmocon,gdb,qemu,dsp}.pid\n",
    "# the TRX chain: DSP, QEMU (trx.highram.elf), osmocon, transceiver ; osmo-bts-trx relaunched if it exited\n",
    "MODE=dsp OSMO_TRX=1 INSNS=120000 LOCKSTEP=1 IQ=none CALYPSO_BSP_STREAM=1 CALYPSO_RHEA_DMA_XFER=1 bash
 ↪  \"$ROOT/c54x_exe/run.sh\""
   ],
   "id": "x02"
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "%%bash\n",
    ". /tmp/c54x-pont/lib.sh\n",
    "# Check: the firmware in BTS mode, FB and SB one burst in ten each, the BTS clock, no stale bursts\n",
    "sleep 30\n",
    "grep -a 'BTS MODE' \"$L/osmocon.log\" | tail -1                      # BTS MODE: 7 514 {5,...} TX 0..0 RX
 ↪  0..0\n",
    "grep -a 'rom-tx\\] [0-9]' \"$L/dsp.log\" | tail -1                    # SB and FB ~ 1/10 of the bursts, NB
 ↪  from the BTS\n",
    "grep -a 'SB recode' \"$L/dsp.log\" | tail -1\n",
    "echo \"STALE BURST (must stop growing): $(grep -ac STALE \"$L/osmocon.log\")\"\n",
    "sed 's/\\x1b\\[[0-9;]*m//g' /var/log/osmocom/osmo-bts-trx.log /run/user/*/osmo-nitb/logs/bts.log 2>/dev/null
 ↪  | grep -E 'clock (started|skew)' | tail -2"
   ],
   "id": "x03"
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "%%bash\n",
    ". /tmp/c54x-pont/lib.sh\n",
    "# The second phone (instance 2) and the air between the two Calypsos\n",
    "INSTANCE=2 MODE=dsp AIR=1 INSNS=120000 LOCKSTEP=1 IQ=none CALYPSO_BSP_STREAM=1 CALYPSO_RHEA_DMA_XFER=1 bash
 ↪  \"$ROOT/c54x_exe/run.sh\"\n",
    "sleep 40\n",
    "tail -1 /tmp/c54x-pont2/air.log                                     # A->B n (NB SB DUMMY FB) : the TRX's
 ↪  bursts reach the mobile\n",
    "grep -a 'FB1\\|SB' /tmp/c54x-pont2/osmocon.log | grep -av 'FBSB_REQ\\|DSP Error' | tail -3   # FB detected,
 ↪  then \"=> SB ... BSIC=7\"\n",
   "sed 's/\\x1b\\[[0-9;]*m//g' /tmp/c54x-pont2/mobile.log | grep -iE 'FBSB RESP|SYSTEM INFO|camped' | tail -3\n",
    "echo \"stop: INSTANCE=2 MODE=dsp bash $ROOT/c54x_exe/run.sh --stop ; MODE=dsp bash $ROOT/c54x_exe/run.sh
 ↪  --stop\""
   ],
   "id": "x04"
  },
  {
```

```
  "cell_type": "markdown",
  "metadata": {},
  "source": "---\n# Part 3: the service\n\n## 13. System information, then camp\n\nWith the downlink flowing, the
↪  phone syncs (FCCH, SCH), reads the system information on the BCCH and\ncamps on the cell.",
  "id": "c45"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": "%%bash\n. /tmp/c54x-pont/lib.sh\nMS_VTY=$(cat \"$L/ms_vty\")\nwait_log \"$L/mobile.log\" \"New
↪  SYSTEM INFORMATION\" 30 || exit 1\nfor i in $(seq 1 60); do vty \"$MS_VTY\" \"show ms\" | grep -q \"C3 camped
↪  normally\" && { echo \"ok      camped after $((i*2)) s\"; break; }; sleep 2; done\nvty \"$MS_VTY\" \"show ms\"
↪  | grep -E \"camped|MM|IMSI|cell\"",
  "id": "c46"
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": "## 14. Location update\n\nThe phone sends a RACH, gets a dedicated channel (SDCCH), authenticates
↪  against the HLR and registers.",
  "id": "c47"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": "%%bash\n. /tmp/c54x-pont/lib.sh\ngrep -a \"\\[montant\\] RACH\" \"$L/dsp.log\" | tail -1\ngrep -ao
↪  'UL bursts=[0-9]* tard=[0-9]* rach=[0-9]*' \"$L/pont.log\" | tail -1\nwait_log \"$L/mobile.log\" \"LOCATION
↪  UPDATING ACCEPT\" 60 || exit 1\nsed 's/\\x1b\\[[0-9;]*m//g' \"$L/mobile.log\" | grep -aoE
↪  'lai=[0-9]+-[0-9]+-[0-9]+' | tail -1\nvty 4254 \"show subscriber cache\" | grep -E 'IMSI|MSISDN'",
  "id": "c48"
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": "## 15. A call to 600 (echo test)\n\nNeeds step 6. The call goes mobile → BTS → BSC → MSC → MNCC →
↪  osmo-sip-connector → Asterisk, and back.",
  "id": "c49"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": "%%bash\n. /tmp/c54x-pont/lib.sh\nMS_VTY=$(cat \"$L/ms_vty\")\nvty \"$MS_VTY\" \"call 1 600\" >
↪  /dev/null\nfor i in $(seq 1 15); do\n  st=\"$(vty \"$MS_VTY\" 'show ms' | sed -n 's/^ *call control state:
↪  //p' | head -1)\"\n  echo \"${st:-(no call yet)}\"; [ \"$st\" = ACTIVE ] && break; sleep 2\ndone\ngrep -a
↪  \"\\[montant\\] TCH\" \"$L/dsp.log\" | tail -2",
  "id": "c50"
 },
 {
  "cell_type": "code",
  "metadata": {},
  "execution_count": null,
  "outputs": [],
  "source": "%%bash\n. /tmp/c54x-pont/lib.sh\n# Hang up\nvty \"$(cat \"$L/ms_vty\")\" \"call 1 hangup\" >
↪  /dev/null; sleep 2\nvty \"$(cat \"$L/ms_vty\")\" \"show ms\" | grep -E \"call control state|MM\" || echo \"no
↪  call\"",
  "id": "c51"
 },
 {
  "cell_type": "markdown",
  "metadata": {},
```

```
   "source": "---\n## Stop and clean up\n\nReverse order, then the shared files: a leftover `/tmp/ms_data` makes
↪  the next mobile's bind fail\nsilently, and a leftover `/dev/shm/calypso_horloge` slaves the bridge to a dead
↪  session. If you started\nthe rssi application (last section), stop it first with `banc-rssi --stop`.\n\nIf the
↪  network dropped during the run, look at the BTS first: `osmo-bts-trx` exits on \"PC clock skew too\nhigh\".
↪  Started by hand (step 7) it stays down; `./start.sh` restarts it by itself since 2026-10-23\n(`BTS_GARDIEN=1`
↪  in `qosmo/run_modules/60-bts.sh`, `BTS_GARDIEN_PAUSE` seconds apart, a `[gardien bts]`\nline in its log), and
↪  the bridge caps the pause it asks for (`PONT_CLCK_PAUSE_PLAFOND=38`,\n`PONT_CLCK_SKEW_ZONE=60` in
↪  `pont/trx.py`).\n",
   "id": "c52"
  },
  {
   "cell_type": "code",
   "metadata": {},
   "execution_count": null,
   "outputs": [],
   "source": "%%bash\n. /tmp/c54x-pont/lib.sh\nfor n in pont mobile osmocon gdb qemu dsp bts sip bsc msc mgw hlr
↪  stp; do\n  p=$(cat \"$L/$n.pid\" 2>/dev/null) && kill \"$p\" 2>/dev/null && echo \"stopped
↪  $n\"\ndone\nasterisk -rx \"core stop now\" 2>/dev/null || true\nsleep 1\nrm -f /dev/shm/calypso_api_ram
↪  /tmp/calypso_dsp.sock /tmp/osmocom_l2 /tmp/qemu-monitor-pont.sock \\\n      \"$L/modem.pty\" /tmp/osmocom_sap
↪  /tmp/ms_data /dev/shm/calypso_horloge \\\n      /dev/shm/calypso_rach /dev/shm/calypso_sdcch_ul
↪  /dev/shm/calypso_tch_facch_ul \\\n      /dev/shm/calypso_tch_sacch_ul /dev/shm/calypso_tch_ul
↪  /dev/shm/calypso_tch_cfg \"$L\"/*.pid\npgrep -a -f
↪  'osmo-(stp|hlr|mgw|msc|bsc|bts-trx|sip-connector)|c54x_exe|qemu-system-arm|osmocon|mobile -c|pont_dsp' \\\n
↪  && echo \"still running (above)\" || echo \"all stopped\"",
   "id": "c53"
  },
  {
   "cell_type": "markdown",
   "id": "r01",
   "metadata": {},
   "source": "---\n## Optional: the rssi application on a fourth Calypso (`banc-rssi`)\n\n[2026-10-21, 2026-10-23]
↪  `banc-rssi` starts **instance 4** of `c54x_exe/run.sh` with `FIRMWARE_APP=rssi`:\na C54x DSP, QEMU running
↪  osmocom-bb's `rssi.highram.elf` and osmocon, no mobile and no bridge of its own.\nIts C123 screen and keypad
↪  are emulated (window: `c123-3`). It is kept apart from the rest of the bench:\n`./start.sh --stop` and
↪  `banc-stop` leave it running; `banc-rssi --stop` (or `banc-rssi-stop`) stops the\nwhole instance.\n\nWhere
↪  its downlink comes from:\n\n- **classic BTS** (`pont_dsp.py` running, the case of this notebook): no air,
↪  `AIR=0`; the bridge's copy\n  to UDP 6762 (`PONT_DSP_COPIES`, step 12a) feeds its BSP;\n- **`--bts`**:
↪  `pont/air.py` relays the TRX's bursts, as for the mobile of step 12b.\n\nRealistic air levels are on by
↪  default (`CALYPSO_AIR_NIVEAUX=1`, `qosmo` `calypso_bsp.c`): the cell is heard\nonly on its ARFCN
↪  (`CALYPSO_FAUX_AIR`, taken from `PLAN_ARFCN` in `↩
/etc/osmocom/radio-plan.env`), the\ndownlink at `CALYPSO_AIR_DL_DBM`, the mobile's uplink at
↪  `CALYPSO_AIR_UL_DBM`, noise elsewhere.\n`CALYPSO_AIR_NIVEAUX=0 banc-rssi` gives the old behaviour (everything
↪  at full level, on every frequency).\n\nIt needs the network and the bridge of Part 1 / step 12 still running:
↪  run it **before** the Stop cell.\nThe helpers are shell functions from `scripts/osmo-banc-aliases.sh`
↪  (installed as\n`/etc/profile.d/98-osmo-banc-aliases.sh`); the cells source it.\n"
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "r02",
   "metadata": {},
   "outputs": [],
   "source": "%%bash\n. /tmp/c54x-pont/lib.sh\n. \"$O/scripts/osmo-banc-aliases.sh\"\nbanc-rssi\n"
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "r03",
   "metadata": {},
   "outputs": [],
   "source": "%%bash\n. /tmp/c54x-pont/lib.sh\n. \"$O/scripts/osmo-banc-aliases.sh\"\n# Check: instance 4 up
↪  (logs in /tmp/c54x-pont4), the downlink reaching its BSP; window: c123-3\nbanc-rssi --status\nss -ulnH '(
↪  sport = :6762 )' | grep -q . && echo \"ok      udp 6762 bound (BSP of instance 4, where the bridge copies its
↪  bursts)\"\ntail -3 /tmp/c54x-pont4/dsp.log\necho \"window: /usr/bin/python3 $O/tools/osmo-c123-gtk.py
↪  --instance 4   (alias c123-3)\"\necho \"stop:   banc-rssi --stop\"\n"
  }
```

```
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "name": "python"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 5
}
```