Arai — Audit & Compliance Feature Inventory

How Arai turns CLAUDE.md from advisory prose into enforceable policy with a complete local evidence trail.

1. Evidence trail — every action is recorded, traceable, and queryable

Every guardrail firing is recorded on disk on the developer's own machine, traceable back to the line of policy that produced it. No log shipper required; queryable directly with grep, jq, or the arai audit CLI.

Feature	What it does	Why compliance teams care
Local JSONL audit log	One line per rule firing at ~/.arai/audit/<project>/<YYYYMMDD>.jsonl. Append-only. Records timestamp, hook event, tool, session ID, prompt preview, decision (deny/inject/review), every matched rule.	Tamper-evident record of every AI-assistant action where a guardrail evaluated, on the developer's own filesystem. Day-bucketed files map cleanly to retention policies.
Derivation trace per firing	Each rule entry records the source file, line number, and parser layer that produced it (e.g. "from CLAUDE.md:42, layer-1 imperative").	Auditors can answer "why did this rule fire?" without code spelunking. Maps every enforcement event back to a human-authored line of policy.
Decision provenance	Each firing records the resolved decision (deny / inject / review) plus the severity that produced it.	Distinguishes "advised but allowed" from "blocked" in evidence; supports policies like "all block-severity firings must be reviewed weekly."
arai audit query CLI	Filter the audit log by --since, --tool, --event, --outcome, --rule (substring against subject/predicate/object). JSONL output for pipelines.	Investigations don't need a SIEM or log shipper — grep and jq work directly on the local file.
arai_recent_decisions MCP tool	Read-only audit feed exposed back to the AI agent so the model can self-check what it was just denied for.	Closes the model-side feedback loop; reduces deny-and-retry loops the audit log would otherwise show repeatedly.

Feature	What it does	Why compliance teams care
Hash-chained audit log	Every audit-log line carries prev_hash + hash (SHA-256 over canonical bytes); a per-day .head.YYYYMMDD sidecar anchors the chain tip.	Tampering, reordering, or deletion of any line is detectable. Backs the “tamper-evident” claim with a real mechanism rather than append-only-by-convention.
arai audit --verify	CLI walks the hash chain across every day-bucket for the project and exits non-zero on any tamper / reorder / deletion. JSON output for CI / cron.	Evidence integrity becomes a gateable CI step or scheduled job. Auditors can independently confirm the trail hasn’t been edited.

2. Compliance verdicts — did the model honour the rule?

Pre/Post correlation generates evidence of enforcement effectiveness, not just enforcement attempts. Per-rule ratios surface which rules the model is actually honouring versus routing around.

Feature	What it does	Why compliance teams care
Pre/Post correlation	When PostToolUse fires, Arai correlates against recent PreToolUse firings in the same session and emits a Compliance audit event with an Obeyed / Ignored / Unclear verdict per rule.	Generates evidence of enforcement effectiveness, not just enforcement attempts. Answers "is the policy actually working?"
First-definitive-wins dedupe	One verdict per Pre firing — unrelated subsequent commands in the 5-minute window can't inflate the obeyed/ignored count.	The headline number is statistically defensible. A maintainer saying "92% compliance on rule X" can stand behind that number.
Per-rule compliance ratios	arai stats --by-rule rolls up fires / obeyed / ignored / unclear / ratio per rule, with a warning flag on low-ratio rules with enough volume to mean it.	Single-glance answer to "which rules is the model routing around?" — the rules to either rewrite or escalate.
arai audit --outcome=ignored	Shortcut filter for compliance verdicts where the model ran the action despite the warning.	Direct feed to a compliance review meeting: "here's every time a rule fired and the model ignored it this week."

3. Controlled rollout — graduated enforcement, no all-or-nothing

Severity tiers map natural-language policy weight to enforcement strength. Rollout is graduated — individual rules can move into deny mode independently, with a project-wide kill switch as the rollback path.

Feature	What it does	Why compliance teams care
Severity tiers	never / forbids / must_not → Block (deny tool call); always / requires / enforces → Warn (advise); prefers / learned_from → Inform (soft nudge). Severity derived from rule predicate.	Maps natural-language policy weight to enforcement strength deterministically. The writer's grammatical choice (should vs must) sets the response.
Per-rule severity override	arai severity <pattern> block|warn|inform pins individual rules' enforcement strength in a dedicated DB column that survives arai scan re-classification.	Incremental rollout: ship a rule set in advise mode, watch the compliance ratios, escalate individual rules to deny one at a time once the model is honouring them.
Project-wide deny-mode kill switch	ARAI_DENY_MODE=off forces advise-only for the whole rule set without changing any rule.	Rollback path that doesn't require editing policy files or restoring config from backup.
Rule expiry	Trailing (expires YYYY-MM-DD) / (until YYYY-MM-DD) annotation; expired rules are filtered out at load.	Time-boxed policies (incident-driven rules, migration windows) self-prune without manual cleanup.

4. Policy authoring & governance

Policy authoring tools that make CLAUDE.md edits reviewable, testable, and distributable across an organisation.

Feature	What it does	Why compliance teams care
Shared policies via arai:extends	Inherit org-wide rules from a trusted upstream URL via an arai:extends HTML comment. Per-URL trust list, HTTPS only, 512 KB cap, 24-hour cache. SSRF-hardened (no loopback, RFC1918, link-local, cloud metadata).	Org-wide policy distribution without a hosted policy service. The trust list plus transport hardening covers the supply-chain security checks an enterprise InfoSec team will ask.
arai diff <file>	Preview rule-set delta before saving an instruction-file edit (added / removed / moved). JSON output for pre-commit hooks.	Policy change review: surface every rule that an edit would add or remove, before the change lands. Pairs with PR review of CLAUDE.md edits.

Feature	What it does	Why compliance teams care
arai lint <file>	Parse a file and show extracted rules with classified intent — read-only, no DB write.	Policy authors iterate on rule wording with immediate feedback on what would extract.
arai why <action>	Replay a hypothetical tool call through the live match pipeline. Shows matched rules, severity, source, layer, match-percentage. Read-only.	Policy QA: verify a candidate rule fires (or doesn't) on a given action before deploying.
disable_rule API	Disable a specific rule by (subject, predicate, object) tuple — load_guardrails skips it on every read.	Surgical override without editing the source policy file (which may be under SOX-style change control).
Rule health checks	arai status flags duplicate rules (same SPO from multiple sources) and opposing predicates (same subject with both prohibitive and required predicates).	Surfaces policy drift and conflicts at review time, not at firing time.

5. Determinism & regression protection

Policy enforcement is testable. Regression tests, captured firings, and a pinned parser-coverage corpus prevent silent behavioural drift across releases.

Feature	What it does	Why compliance teams care
arai test scenario harness	Replay synthetic hook payloads through the same match_hook pipeline the live hook handler uses. CI-friendly JSON output.	Policy regression tests: every change to CLAUDE.md must keep specified scenarios passing. Pluggable into existing CI.
arai record	Build scenario files from recent audit-log entries — capture a real firing as a regression test.	Turns observed enforcement into pinned tests with one command. The compliance team builds a regression corpus from real incidents.
Parser-coverage corpus	tests/parser_coverage/ ships a synthetic CLAUDE.md exercising every pattern with positive and negative cases; arai lint --json is driven against it as a regression bar.	Guarantees the parser doesn't silently start under-extracting (rules slipping through) or over-extracting (architecture docs treated as rules) across releases.

Feature	What it does	Why compliance teams care
Schema migration framework	PRAGMA user_version-tracked migrations in the audit DB; migrations are append-only, idempotent, and crash-safe.	Long-lived audit databases survive Arai upgrades without data loss or manual intervention. Critical for retention windows that span multiple releases.
WAL journal + synchronous=NORMAL	Audit DB uses Write-Ahead Logging with normal-durability syncs.	Concurrent reads (queries, dashboards) don't block writes (live hook events). Crash-recovery preserves committed audit lines.

6. Supply chain & operational hardening

Defensive controls covering binary distribution, upstream policy fetching, MCP input bounds, and the architectural separation of local audit data from anonymous telemetry.

Feature	What it does	Why compliance teams care
SHA-256 checksum verification	The npm postinstall and install.sh paths verify the downloaded binary against checksums.txt published with each GitHub release. Refuses to install on mismatch.	Defense against compromised CDN or release-pipeline tampering. Auditable supply-chain control.
arai:extends SSRF hardening	Refuses loopback, RFC1918, link-local (including 169.254.169.254 cloud metadata), CGNAT, multicast, IPv6 ULA. Disables redirects. Forces Accept-Encoding: identity. Refuses to follow symlinks in cache paths.	Closes the egress-side and decompression-bomb surfaces a trust-list-fetched policy file could otherwise expose.
MCP input limits	arai_add_guard caps rule bodies at 1 KiB, reasons at 4 KiB, MCP-source rules at 1000 per project.	Prevents a malfunctioning or compromised agent from filling the rule store with garbage or DOS'ing the parser.
Anonymous opt-out telemetry	Aggregate counters only — never collects file paths, rule text, code content, API keys, or anything project-identifying. ARAI_TELEMETRY=off opts out. Architecturally separate from the local audit log.	The local audit log can never accidentally leak via the telemetry channel — they share no code paths. Privacy-policy-friendly distinction.

Feature	What it does	Why compliance teams care
No network on the hook hot path	PreToolUse / PostToolUse hooks consult only the local SQLite DB and audit JSONL. No outbound calls during enforcement.	Enforcement keeps working offline, in air-gapped environments, and during outages. No external service to monitor for the SLA.
Single-binary distribution	Lean default binary; full binary with optional ONNX-based local enrichment. No runtime dependencies for users.	Simplifies allowlisting / endpoint-management review. One artifact to track, one checksum to pin.
MCP authentication	Optional shared-secret via ARAI_MCP_AUTH_TOKEN. When set, initialize.params.auth_token must match (constant-time compare); subsequent calls on the same stdio connection inherit auth. Open by default for backwards compatibility.	Closes the "agent on stdio is implicitly trusted" surface for organisations that want to gate which agents may drive Arai's MCP tools. Notification handling stays JSON-RPC 2.0 compliant.
Extends cache-at-rest signature	Cached upstream-policy files now carry a <hash>.md.sha256 sidecar. Mismatched or missing sidecars are treated as a cache miss in both the fresh-read and stale-while-error paths.	Closes the cache-tampering surface beneath the trust list. An attacker with write access to the cache directory can no longer swap a cached policy out without the next read detecting it.
Windows audit ACLs pinned	First audit write on Windows shells once to icacls /inheritance:r /grant:r USER:(OI)(CI)F and drops a .arai_acl_set marker so subsequent writes skip the call.	Matches the Unix 0700/0600 lock-down on the audit directory. Removes the "typically user-only by default" soft assumption on a Windows fleet.
Telemetry queue size cap	Hard 2 MiB cap on telemetry_queue.jsonl, enforced via a single metadata() syscall in track(). Events above the cap are silently dropped (upstream sink dedups on rule_hash anyway).	Bounds on-disk growth for installs that only ever invoke hooks (and never the non-hook CLI commands that flush). No risk of the telemetry queue itself becoming a long-term storage liability.

7. Token economics — secondary signal, enterprise-relevant

Secondary positioning: Arai's primary mission is correctness, but its design also reduces token spend on long sessions and avoidable retry cycles. The numbers are calibrated estimates with documented constants, not measurements.

Feature	What it does	Why compliance teams care
Per-session repeat-injection suppression	When a rule fires a second time in the same session, Arai emits a compact one-liner instead of the full payload.	Reduces context-window pollution and token spend on long sessions; secondary signal.
Token-economics estimate in arai stats	Calibrated "tokens saved" estimate from suppressions, denied-and-honored mistakes, and advised-and-honored events. Labelled as estimate, not measurement.	Budget-conscious teams can quantify Arai's contribution to LLM cost reduction; the constants are documented and auditable in source.

8. SOC 2 Trust Service Criteria mapping

Arai is not itself a certified product. The following mapping documents how the controls in sections 1–7 align with the SOC 2 Trust Service Criteria so a procurement / InfoSec reviewer can evaluate Arai against their own framework without redoing the analysis.

TSC criterion	What it covers	Arai feature(s) that satisfy it
CC6.1 — Logical access controls	Restricts logical access to data and system resources to authorised users.	Audit directory locked to owner-only on disk (0700/0600 on Unix; icacls /inheritance:r /grant:r on Windows, pinned on first audit write via a marker file).
CC6.6 — External system / supply chain	Protects information against vulnerabilities introduced by external sources or third-party software.	Every install path verifies the downloaded binary against published checksums.txt (SHA-256). MCP server input caps. Single-binary distribution.
CC6.7 — Data in transit	Protects information in transit, including limits on what leaves the host.	No network on the hook hot path. (noenrich) per-rule opt-out plus pre-send locality verdict for the enrichment channel. arai:extends HTTPS-only with SSRF-hardened transport (refuses loopback, RFC1918, link-local, cloud metadata, redirects).
CC7.2 — System monitoring	Detects security events through monitoring of system activity and configuration changes.	Per-firing JSONL audit log with derivation trace (source file, line, parser layer). Every line carries SHA-256 prev_hash + hash; arai audit --verify walks the chain across every

		day-bucket and exits non-zero on any tamper, reorder, or deletion.
CC7.3 — Evaluation of security events	Evaluates events to determine whether the control environment is operating as designed.	Pre/Post correlation emits Obeyed / Ignored / Unclear verdicts per rule. arai stats --by-rule rolls these into per-rule compliance ratios. Evidence of enforcement effectiveness, not just attempts.
CC8.1 — Change management	Authorises, designs, develops, and tests changes prior to implementation.	arai diff previews rule-set delta before save. arai test replays scenarios through the live match_hook pipeline. arai record captures real firings as regression fixtures. Synthetic parser-coverage corpus locks parser behaviour across releases.
CC9.2 — Vendor / third-party management	Manages risks associated with vendors and business partners (here: the agent that drives Arai's MCP server, and shared upstream policy sources).	Optional shared-secret authentication on the MCP server via ARAI_MCP_AUTH_TOKEN (constant-time compare at initialize). arai:extends per-URL trust list. Cached upstream-policy files carry SHA-256 sidecars; mismatched or missing sidecars are treated as a cache miss.

Suggested enterprise positioning

A short version of the matrix above for a sales / procurement-review summary deck:

1. Local-first audit log — every guardrail firing recorded on disk, traceable to the source CLAUDE.md line, queryable without external services.
2. Compliance verdicts — every PostToolUse correlated against PreToolUse to produce per-rule obeyed/ignored ratios. Evidence of enforcement effectiveness, not just attempts.
3. Graduated enforcement — severity tiers (Block / Warn / Inform) derived from policy language. Per-rule overrides for incremental rollout.
4. Org-wide policy distribution — arai:extends with explicit per-URL trust list and SSRF-hardened transport.
5. Regression-tested policy — arai test plus arai record plus the synthetic parser-coverage corpus turn rule changes into CI assertions.
6. No data egress — no network on the hot path, opt-out telemetry architecturally separate from the audit log, anonymous-counter-only when on.

7. Supply-chain hardened — SHA-256 verified binary downloads, MCP input caps, schema-migrated audit database with WAL durability.
