



EXTRACT FROM THE PFMS TREASURY WITHDRAWAL PROPOSAL. CHECK THE FULL PROPOSAL PDF FOR REFERENCES

[Read the full proposal \(PDF\)](#)

Work Packages

The 15 month programme is organised into eight work packages. Six of them build the six engines described in Section 3.2, one engine each, in the order a record travels through them. The seventh puts those engines to work in the five stages of a case and proves them end to end on mainnet. The eighth carries security, independent assurance and governance across the whole period.

#	WORK PACKAGE	ENGINE	PRINCIPAL ACCEPTANCE
WP1	Source Engine: Record Definitions, Field Schemas and Version Registry	01	M1
WP2	Capture Engine: Edge Agent, Store and Forward Queue and Read Only Connectors	02	M2
WP3	Identity Engine: DID Registry, Officer Credentials and Device Identities	03	M2
WP4	Stamping Engine: Fingerprints, Combined Signatures, Batching and Completeness Counts	04	M2
WP5	Commit Engine: Stage Smart Contracts, Mainnet Submission and Signer Service	05	M3
WP6	Verification Engine: Proof Bundles, Verification Layer and Evidence Trail Documents	06	M3
WP7	Five Stage Modules, End to End Testing, Mainnet Load Proving and Handover	Across all	M2 to M4
WP8	Penetration Testing, Open Source Release, Independent Assurance, Governance and Reporting	Across all	M4

Each package states its scope, its deliverables, the evidence by which delivery is judged, and what it depends on. The budget line each package draws on is set out in Section 6.3.

WP1: Source Engine, Record Definitions, Field Schemas and Version Registry

Scope. A record can only be stamped if the system knows what it is. This package writes down every kind of record that PFMS will handle. Numerous distinct record kinds run across the five stages of a case, among them camera footage, control room call and dispatch entries, station diary entries, ANPR captures, armoury issue and return, instrument calibration certificates, crime scene and forensic transfer records, Malkhana movements, prosecution filings, court orders, etc. For each kind the definition states where the record starts and ends, what its fields mean, and what its date and time refer to at that source.

Every definition is versioned. When a definition changes later, records stamped under the earlier version still check, because the version they were stamped under is written into the stamp itself.

Deliverables.

- A written definition for every record kind, checked against real samples of that record.
- A version registry, so that every definition carries a version number and every stamp names the version it used.
- Test vectors for every record kind: sample records and the fingerprint each one must produce, published.
- A guide to testing a record end to end and how to reproduce its fingerprint and check it against Cardano.

Evidence. Working artifacts: the published definitions, the version registry and the test vectors. Reproducible test: a sample record of each kind parses cleanly against its definition, with every field and its timestamp resolved, demonstrated at review.



Dependencies. Records from each source system, which the state has already provided. WP2 and WP4 build on these definitions and cannot be accepted before them.

WP2: Capture Engine, Edge Agent, Store and Forward Queue and Read Only Connectors

Scope. This package builds the software that sits where a record is born. The edge agent runs beside the recorder or the terminal and hashes the raw bytes of a record at the moment they are written, so the file is never converted or saved again, and nothing waits. That hash, with the record's identity, is what the stamping engine in WP4 seals and batches. A store and forward queue holds fingerprints in numbered order on the device and sends them up when the link returns, so a capture point with no network still signs and still queues. Cardano fixes the outer bound. The block that carries a fingerprint proves the record existed by that block, fixed by the network and beyond anyone's reach to move. Three times are kept for every record: the event time claimed by the source, the signing time taken from a synchronised network clock with the device's drift recorded against it, and the block time on Cardano. The first two are signed claims and the third bounds them. The permitted clock error is set per record kind, and a record that exceeds it is committed with its drift declared rather than silently accepted.

The connectors are the other half. They read from the systems the state already runs: CCTNS, DVR and NVR recording systems, control room and voice logging systems, ANPR databases, station diary systems, armoury systems, and the prosecution and court systems. Every connector is read only. It watches for the moment a record is committed in the source system and reads it. It never writes anything back.

Deliverables.

- The edge agent, deployable on any capture point, hashing at write and holding a numbered local queue that survives a restart and an extended loss of network.
- Cardano block time as the outer bound for every fingerprint, with the claimed event time, the device signing time, and the permitted clock error all recorded alongside.
- Read only connectors for each source system listed above, built on one shared connector framework so a new source can be added without new core code.
- An induced failure test: the link is cut, records keep queuing, the link returns, and every record is committed in order with the gap visible and then closed.

Evidence. Working artifacts: tagged releases of the agent and the connectors. Reproducible test: the induced failure test shows that no record is lost and none is reordered, and a read only test in which the source system is checked before and after shows no change. On chain data: fingerprints from live capture points committed on preprod at M2 and on mainnet at M3.

Dependencies. WP1 for the record definitions. WP4 for the fingerprint and batch format. A test instance of each source system, arranged with the state.

WP3: Identity Engine, DID Registry, Officer Credentials and Device Identities

Scope. Nothing can be stamped until the device that made the record and the officer who was on duty are both known. This package builds the credential registry that answers those two questions. It is built on the systems the department already keeps, CCTNS, NAFIS and PIS, and it does not copy personal data out of them. It issues a DID and a signing key to every camera, recorder and terminal before that device can sign anything, and it carries each officer's role, station and the period for which the credential is valid.

The registry is held off chain. What reaches Cardano is a single value that stands for the whole registry at a point in time, carried inside every stamp. A court can later confirm that a device and an officer were valid on the day a record was made, without anyone reading the registry itself.

Deliverables.

- The credential registry, with its data model built to the CCTNS, NAFIS and PIS schemas.
- Device identity issuance, so that every capture device holds a DID and a signing key before it enters the signing path.
- Officer credentials carrying role, station, scope and validity period, each version with the interval it applies to, and a revocation path for when a credential ends.
- The registry state digest, computed on a fixed schedule, published as each version takes effect and carried in every batch commit, so a state cannot be constructed after the fact to authorise a signature.
- The DID issuance, rotation and revocation service for devices and officers, built to the CCTNS, NAFIS and PIS schemas.

Evidence. Working artifacts: the registry, its schema and the issuance and revocation service. A record signed against a registry version older than a revocation is accepted only at reduced assurance and reported as such. Reproducible test: a test credential is revoked, records signed against a later registry version fail verification, and records signed offline against an earlier version are reported at reduced assurance, demonstrated at review. On chain data: registry digests inside committed batches. No personal data, personnel record or biometric appears in any evidence.

Dependencies. Read access to CCTNS, NAFIS and PIS reference data, already received from the state.

WP4: Stamping Engine, Fingerprints, Combined Signatures, Batching and Completeness Counts

Scope. This package takes a record and the identities attached to it and turns them into something that can be written to Cardano and checked years later. It does five things. First, it lays every record of a given kind out in a fixed order before fingerprinting it, so the same record always produces the same fingerprint whichever system exported it. Second, it signs the record, the device and the officer together as a single unit, so nobody can keep the record and change who made it. Third, it combines all the fingerprints from one source over one window into one summarised string of hashes, the Merkle root. Fourth, it counts the records expected, captured and sealed in that window, so a silent camera is noticed. Fifth, it versions every layout, so a seal from year one still checks in year ten.

Deliverables.

- The canonicaliser, which produces the fixed form for each record kind defined in WP1.
- The signing path that seals record, device and officer into one signature.
- The batcher, which builds one summarised string of hashes per source per window.
- The reconciler, which reports expected against captured against sealed for every source and every window.
- The version rule, so that a change to any form does not invalidate what was already stamped.

Evidence. Reproducible test: a record run through the canonicaliser produces the same fingerprint every time, and altering one byte changes the fingerprint. A version bump test: a form is revised, and records stamped under the earlier version still verify. Working artifacts: tagged releases and the batch format specification. On chain data: batches committed with their hashes.

Dependencies. WP1 for the definitions. WP3 for the identities that are sealed in.

WP5: Commit Engine, Stage Smart Contracts, Mainnet Submission and Signer Service

Scope. This package writes the hashes to Cardano mainnet and builds the rules that live there. Records are committed in batches at every stage, grouped by source and by window. Each commit carries the summarised string of hashes, the window, the record count and the registry state at the time, and the state is partitioned so that sources commit concurrently. The block it lands in proves every record inside it existed by that block. The contracts check the transitions carried inside each batch against the state the previous batch left behind, so a move that is not allowed does not change the state. The batch still lands, and the attempt is recorded on chain as a rejected transition, carrying the same signed identity binding as any other record. Each stage has multiple smart contracts to enforce multiple sets of rules such as capture rules accept a capture only from a device holding a valid DID inside a closed window. Only fingerprints, summarised strings of hashes, digests and counts go on chain.

Deliverables.

- The batch commit format and the full set of stage smart contracts, covering every rule each stage needs, written and tested on preprod.
- The committed custody state and the validator that enforces the exhibit state machine inside each batch.
- The signer service and the submission infrastructure, with multi signature signing so that no single key can commit alone.
- A Cardano node operated for PFMS, with monitoring.
- The contract set frozen at M2 and ready for penetration testing, and deployed to mainnet only after every finding has been fixed and retested.

Evidence. On chain data: contract addresses, policy identifiers, transaction hashes and counts, on preprod through M2 and on mainnet from M3. Reports: the penetration test report and its remediation record. Reproducible test: a handover that breaks a rule, for example issuing an exhibit that was never returned, leaves the exhibit unmoved and appears on chain as a rejected attempt, demonstrated at review.

Dependencies. WP4 for the batch contents. WP3 for the registry digest. WP8 for the penetration testing that gates deployment to mainnet.

WP6: Verification Engine, Proof Bundles, Verification Layer and Evidence Trail Documents

Scope. Everything before this point exists so that a check can be made later. This package builds that check. It builds the verification layer, which takes a request for any stamped record, reads its proof against Cardano directly, and fails if a single byte has changed. It returns the evidence bundle: the record, the proof that it sat inside a stamped batch, and a reference to the transaction that carried it. And it builds the evidence trail document it returns alongside: what checked out, what did not, which registry version was in force, and whether the officer's credential was valid and unrevoked at the time.

Every check runs through the verification layer, which reads the proof against Cardano directly. Prosecution, defence and the judge each request the same record and receive the same bundle.

Deliverables.

- The evidence bundle format, assembled by the verification layer for any stamped record on request.
- The verification layer, which takes a request from prosecution, defence or the court and checks the record directly against Cardano.
- The evidence trail document, in plain language, returned by the verification layer.
- Written documentation on how to request a record and how to read the evidence trail document returned.



Evidence. Reproducible test: a record returns a bundle that verifies, and altering one byte of the record makes the verification fail, demonstrated at review. Working artifacts: the verification layer and its documentation. Reports: sample evidence trail documents returned for real stamped records, on preprod at M2 and on mainnet at M3.

Dependencies. WP5 for the commit format the proof points to. WP4 for the batch format. WP3 for the credential validity check.

WP7: Five Stage Modules, End to End Testing, Mainnet Load Proving and Handover

Scope. The six engines are shared by every stage. This package builds what each stage needs on top of them, and then rolls the whole system out. Stage 1 gets the six Pre-Crime record modules, for cameras, control rooms, ANPR, station registers, armouries and instruments. Stage 2 builds on our FMS work, which is already proven on Cardano mainnet, and extends it to this state's laboratory network, its forensic vans and the CCTNS instance in use. Stage 3 gets the Malkhana module, with QR intake, issue and return, stock verification and disposal. Stage 4 gets the chargesheet versioning, witness statement, arrest and remand, and exhibit list modules. Stage 5 gets the court verification report.

Testing goes stage by stage. Each module is run against real records from the source systems, the connectors are exercised end to end, and every failure mode is provoked on purpose: a dropped link, a stopped camera, an exhibit issued twice. Load is then proven on mainnet: 100,000 transactions produced by records captured from live sources through the deployed contracts.

Deliverables.

- The Stage 1 record modules, the Stage 2 investigation modules, the Stage 3 Malkhana module, the Stage 4 prosecution modules and the Stage 5 court module, each tested end to end and accepted.
- The integrated pipeline: one record travelling from any stage through all six engines and out to an evidence trail document.
- End to end test results for every stage and every record kind, with every failure mode provoked and recovered.
- Load proving on mainnet: 100,000 transactions from live sources, with the reconciler showing no gap.
- A technical guide for each stage on how to check a record and read the result.

Evidence. On chain data: transactions from live sources on mainnet, with counts per source and per window. Reports: test reports for each stage and the load proving report. Reproducible test: one record is followed from capture to its evidence trail document and confirmed by the verification layer at the far end, and one exhibit is followed through every custody handover from intake to disposal.

Dependencies. All six engine packages. WP8 for the penetration testing that must pass before anything reaches mainnet.

WP8: Penetration Testing, Open Source Release, Independent Assurance, Governance and Reporting

Scope. This package runs across the whole period. It procures the independent assurer as soon as the first tranche is released and publishes the procurement. It covers penetration testing at M2, covering the stage smart contracts and the signer service as well as the edge agent, the connectors and the registry services, and sees every finding fixed and retested before anything reaches mainnet. It publishes the PFMS core codebase under the Apache License 2.0 at M3, once penetration testing and mainnet deployment are complete, and completes the open source handover at M4 with the final tagged production release. It keeps the accounts, reports every milestone, and returns unspent funds.



Deliverables.

- Independent assurer procured through competitive procurement, with the shortlist, the selection basis and the confirmation of no conflict of interest published.
- Threat model and security architecture review, reported in summary.
- Penetration testing of the stage smart contracts, the signer service, the edge agent, the connectors and the registry services, with every finding fixed and retested before deployment to mainnet.
- Open source release at M3: the six engines, the stage smart contracts, the DID registry, the verification layer, the batch format and the test vectors published under the Apache License 2.0, while state specific connector configurations, signing keys and credentials are kept private.
- Final open source handover at M4: the final tagged production release, technical documentation, deployment guides, reproducible tests and all five stage modules in the public repository.
- Milestone reports at M1, M2, M3 and M4, and a final reconciliation with the return of unspent ADA.

Evidence. Reports: the procurement record, the penetration test report and its remediation record, assurer reviews and milestone reports. Public repository: the tagged releases, the licence and the reproducible build and test documentation. On chain data: treasury movements and milestone disbursements.

Dependencies. Runs across every package. Penetration testing and its fixes should complete before WP5 deploys to mainnet.

Milestones and Acceptance

Delivery is structured around a kickoff tranche and four delivery milestones across the 15 month funded period. Each milestone has named acceptance evidence. The first tranche (Milestone 0, Kickoff) is released on administrator contract signing.

MILESTONE	TIMING	COVERAGE	PAYMENT RELEASED
M0 Kickoff	On administrator contract signing	Kickoff tranche	1,800,000 ADA
M1 Foundations	Month 3	All record kinds defined and versioned with test vectors; contracts running on preprod; credential registry exercised on test data	2,000,000 ADA
M2 Proven on Cardano Preprod	Month 8	One record end to end through all six engines on preprod for Stages 1 to 3; contract set frozen and submitted for penetration testing	2,000,000 ADA
M3 Tested and on Mainnet	Month 12	Penetration testing passed; contracts on mainnet; all five stages tested end to end and accepted; verification layer live; 100,000 transactions from live sources on mainnet; codebase open sourced	2,000,000 ADA
M4 Sustained Usage	Month 15	At least 390,000 transactions from live sources across all five stages in 3 months; open source handover	1,800,000 ADA

The gate. The independent assurer is procured as soon as the first tranche is released, and the procurement is made public: the shortlist, the selection basis, the appointed assurer and its written confirmation that there is no conflict of interest are all published. The assurer reviews every milestone from M1 onward. No milestone tranche is released until the independent assurer and the Cardano Foundation, as administrator, have verified that milestone's deliverables and they have been reported to the community (Section 7).



M1: Foundations (Month 3)

M1 covers the groundwork that must be in place before anything is built on top of it. All record types listed in the table in Section 4.3.2 are documented across facilities, devices, people and evidences, versioned and published along with test vectors, so that from this point onward the same record always produces the same fingerprint, whichever system it came from. The stage smart contracts are written and running on preprod. The credential registry is in place, with its data model built to the CCTNS, NAFIS and PIS schemas, and is exercised on test data. The edge agent exists as a prototype and has passed its first induced failure test. No contract reaches mainnet during this period.

DELIVERABLE	WP	EVIDENCE TYPE
All record types in Section 4.3.2 defined, versioned and published with test vectors	WP1	Working artifacts + reproducible test
Edge agent prototype hashing at write, with the induced failure test passed in the lab	WP2	Reproducible test
Credential registry built to the CCTNS, NAFIS and PIS schemas and exercised on test data	WP3	Working artifacts
Canonicaliser and signing path producing the same fingerprints on every run	WP4	Reproducible test
Contracts for the capture, transit, custody and filing rules running on preprod	WP5	On chain data (preprod)
Independent assurer procured and the procurement published	WP8	Reports
M1 milestone report	WP8	Reports

The definitions and the registry are shown as working artifacts. The fingerprints and the induced failure are shown as tests that run and pass. The contracts are shown as data on preprod. The procurement is shown as a published record. The gate is that the foundations are correct. It is not a volume gate.

M2: Proven on Cardano Preprod (Month 8)

At M2 a single record shall travel the entire length of the system. A record captured from a working source system shall pass through all six engines and emerge as an evidence trail document, which the verification layer shall confirm against preprod. This shall be demonstrated for Stages 1 to 3, namely the Pre-Crime records, the investigation records and the Malkhana. The connectors shall read from test instances of the source systems. DIDs shall be issued to the test set of devices and officers. The contract set shall be frozen for penetration testing. Nothing shall reach mainnet in this period, and this is by design.

DELIVERABLE	WP	EVIDENCE TYPE
Edge agent and read only connectors running against test instances of each source system	WP2	Working artifacts + on chain data (preprod)
DIDs issued to the test set of devices and officers, with revocation tested	WP3	Reproducible test
Batches carrying summarised strings of hashes, counts and registry digests committed on preprod for every Stage 1 to 3 source	WP4, WP5	On chain data (preprod)
Contract set frozen and ready for rigorous penetration testing	WP5, WP8	Reports
Verification layer returning bundles and evidence trail documents from preprod records	WP6	Reproducible test
Stage 1, 2 and 3 modules delivered and tested end to end	WP7	Reproducible test + working artifacts
M2 milestone report	WP8	Reports

Every engine has run against a real record from a real source, and the result has been confirmed by the verification layer at the far end.



M3: Tested and on Mainnet (Month 12)

At M3 the system goes live on Cardano mainnet. Penetration testing is complete, every finding is fixed, and the fixes are retested before anything is deployed. All five stages are tested and accepted. DIDs are issued for all categories, and load is proven by producing 100,000 transactions from live sources through the deployed contracts. The policy identifier and the contract addresses are published at this milestone, so every transaction made from here on can be queried.

Following successful penetration testing and mainnet deployment, the PFMS core codebase is published under the Apache License 2.0, including the six engines, the stage smart contracts, the DID registry, the verification layer, the batch format and the test vectors. State specific connector configurations, signing keys and credentials remain private.

DELIVERABLE	WP	EVIDENCE TYPE
Penetration test findings fixed and retested; multi signature signer service live	WP8, WP5	Reports
Contracts and custody validator deployed to mainnet, with the policy identifier and addresses published	WP5	On chain data
All record kinds committing from live sources on mainnet	WP2, WP4	On chain data
Stage 4 and Stage 5 modules delivered and tested end to end	WP7	Reproducible test + working artifacts
100,000 transactions from live sources on mainnet	WP7	On chain data
Verification layer live and returning bundles on real records; first evidence trail documents returned from mainnet records	WP6	Working artifacts + reports
Open source release: the PFMS core codebase published under the Apache License 2.0, while state specific connector configurations, signing keys and credentials are kept private	WP8	Public repository + licence
M3 milestone report	WP8	Reports

Passed security testing, contracts on mainnet, every record kind committing from live sources and a working verification layer each stand on their own and confirm each other. The transaction hashes can be checked against the penetration test report and the tagged release, so a missing piece shows up and a report on its own cannot carry the milestone.

M4: Sustained Usage (Month 15)

M4 is assessed over a sustained mainnet usage period of three months following M3, with performance measured entirely on chain. Acceptance requires a minimum of 390,000 transactions over the period from the sources live at the time, across all five stages, with continuous daily activity throughout. This reflects the first commissionerates and districts coming onto PFMS, not the state at full adoption.

DELIVERABLE	WP	EVIDENCE TYPE
Sustained mainnet usage for three months: a minimum of 390,000 transactions from live sources across all five stages, under the batching policy published at M3	WP7	On chain data
Usage report showing what the chain records over the period: how much was committed, from which stages, from how many of the registered sources, and how steadily it arrived	WP8	On chain data + reports
Guide to testing a record	WP1	Working artifacts
Final open source handover: the final tagged production release, technical documentation, deployment guides, reproducible tests and all five stage modules in the public repository	WP7, WP8	Public repository + tagged release + licence + reproducible build and test documentation
M4 milestone report	WP8	Reports

The whole period sits on Cardano mainnet under published policy identifiers.



How acceptance works

Every deliverable carries at least one of four evidence types: on chain data, working artifact, reproducible test, or report. Each milestone is accepted on several of these together. A report has to match the tagged release it describes, and both have to match what Cardano shows, so a milestone cannot be passed on paperwork alone and cannot fail because one document is missing.

On chain evidence is addresses, transaction hashes, policy identifiers and counts. Where a table says preprod it means preprod, and where it does not it means Cardano mainnet. Modules are proven on preprod through M2 and on mainnet from M3, after penetration testing has passed. No real record is committed to mainnet before penetration testing of the contracts it lands on, with every finding fixed and retested. Working artifacts are tagged releases, specifications and test vectors. Reproducible tests are documented procedures that give the same result every time they are run, and are demonstrated at each milestone review.

No source record is exposed in acceptance evidence. Records stay in their system of record. What acceptance works on is fingerprints, summarised strings of hashes, proofs, commit references, transaction hashes and counts. That is the same rule the system itself holds to: the fingerprint travels, the file does not.

Proving volume and operating volume are different things. The 100,000 mainnet transactions at M3 are produced by records captured from live sources through the deployed contracts. The figures in Section 4.3.2 are projections of operating volume after launch.