

SYNERGYPARTNERS.AI



TYPE	AUDIENCE	VERSION	DATE
Technical specification	Security Architects / CISO / Audit	1.0	June 2026

The Enterprise Decision Architecture Series

- Volume 1 — Enterprise Decision Control Plane
- Volume 2 — Enterprise Decision Layer Reference Architecture
- Volume 3 — Enterprise Decision Transformation Playbook
- ▶ Volume 4 — Cryptographic Decision Lineage Technical Specification (this volume)
- Volume 5 — Enterprise Decision Governance & Operations Guide

This specification assumes familiarity with *MuSIC — The Enterprise Decision Control Plane*, the *Enterprise Decision Layer Reference Architecture*, and the *Enterprise Decision Transformation Playbook*. It specifies one component of that architecture: Cryptographic Decision Lineage.

Status and scope of this document

This is the public architectural edition of the CDL technical specification. It defines the architecture, trust model, and operational concepts of Cryptographic Decision Lineage. Implementation details — record schemas, field layouts, wire formats, and reference code — are intentionally omitted so the document educates architects without serving as an implementation blueprint.

CDL is enterprise proof infrastructure: a tamper-evident, replayable, independently verifiable record of AI-driven decisions. It is not a blockchain, a cryptocurrency ledger, or a Web3 system.

Notational conventions: the key words MUST, MUST NOT, SHOULD, SHOULD NOT, and MAY are to be interpreted as described in RFC 2119 and RFC 8174 when, and only when, they appear in all capitals.

Contents

Chapter 1 — Executive Summary..... 3

Chapter 2 — Problem Statement.....5

Chapter 3 — Design Principles..... 6

Chapter 4 — CDL Reference Architecture.....7

Chapter 5 — Decision Record Model..... 9

Chapter 6 — Hash Chain Architecture..... 11

Chapter 7 — Digital Signatures & PKI..... 13

Chapter 8 — Evidence Architecture.....15

Chapter 9 — Replay Engine..... 16

Chapter 10 — Verification Engine.....18

Chapter 11 — Storage Architecture.....20

Chapter 12 — Security Model.....22

Chapter 13 — Performance Architecture.....24

Chapter 14 — Compliance Mapping..... 26

Chapter 15 — Legal Defensibility..... 27

Chapter 16 — Decision Interfaces..... 28

Chapter 17 — The Record Family..... 29

Chapter 18 — Operational Guidance.....31

Chapter 19 — Deployment Architectures.....33

Chapter 20 — Future Extensions.....35

Closing Statement..... 37

Chapter 1 — Executive Summary

Purpose

This document specifies Cryptographic Decision Lineage (CDL): an enterprise-grade proof infrastructure for AI-driven decisions. CDL records each governed decision — its reasoning, evidence, policy, alternatives, confidence, approvals, and downstream actions — in a tamper-evident, append-only ledger, and provides mechanisms to replay any decision deterministically and to verify its integrity independently of the operator.

Problem

Enterprises increasingly delegate consequential decisions to AI systems, yet retain no means to prove, after the fact, how a specific decision was reached. Existing telemetry — audit logs, application logs, distributed traces, observability metrics, and model-explainability tooling — records the effects of decisions and the health of systems. None records the decision itself in a form that can be reconstructed and independently verified.

Figure 1.1 — Audit Log vs. Cryptographic Decision Lineage

	TRADITIONAL AUDIT LOG	CRYPTOGRAPHIC DECISION LINEAGE
Records	what changed, when	the decision and the reasoning behind it
Immutability	mutable / operator-trusted	append-only, hash-chained, tamper-evident
Reasoning	not captured	evidence, alternatives, policy, confidence
Reconstruction	not possible	deterministic replay of the exact decision
Verification	trust the operator	independent, cryptographic, no operator trust
Defensibility	limited	court- and regulator-defensible by construction

Figure 1.1 — CDL records what an audit log cannot: the decision, its reasoning, and a means to verify both.

Why enterprise audit logs are insufficient

Audit logs are typically mutable by privileged operators, record state transitions rather than reasoning, and rely on trust in the logging system itself. They answer what changed and when. They cannot answer why a particular AI-driven decision was made, nor demonstrate that the recorded answer has not been altered.

Why explainability is insufficient

Model explainability techniques approximate the rationale of a model in general terms — feature attributions, salient inputs, counterfactuals. They are statistical and post-hoc. They do not produce a per-decision, version-pinned, evidence-bound record that a third party can reconstruct and verify. Explainability describes tendencies; CDL records facts.

Why replay matters

The defining capability of CDL is deterministic replay: re-executing the exact decision-map version against the exact policy version and the exact evidence recorded at decision time, to reconstruct the decision precisely as it was made. Replay converts a decision from an unrepeatable past event into a reproducible artifact — the foundation of both verification and legal defensibility.

Design goals

Goal	Statement
Immutability	A committed record MUST NOT be alterable without detection.
Replayability	Any recorded decision MUST be reconstructable deterministically.
Verifiability	Integrity MUST be provable using only public material.
Non-repudiation	Each record MUST be cryptographically bound to its signer.
Least trust	Verification MUST NOT require trust in the operator.
Privacy	Evidence MUST be minimized and protected.
Scalability	The architecture MUST support millions of decisions.
Independent verification	Third parties MUST be able to verify without the operator.

Chapter 2 — Problem Statement

This chapter establishes precisely which capability is missing from the modern enterprise telemetry stack and why each existing layer, though valuable, cannot provide it.

Figure 2.1 — The Question Existing Telemetry Cannot Answer

Application & audit logs record events and state changes	<i>answers: what changed</i>
Distributed tracing follow a request across services	<i>answers: where & how long</i>
Observability / metrics measure system health	<i>answers: is the system healthy</i>
Model explainability approximate model rationale	<i>answers: plausible feature attribution</i>
None answers: “Why did the AI make this specific decision — and can you prove it?” The missing capability is decision-level proof: reasoning, evidence, policy, and confidence, recorded and independently verifiable. Cryptographic Decision Lineage provides it.	

Figure 2.1 — Each telemetry layer answers a different question; none answers decision-level “why, provably.”

Traditional logging and audit logs

Application and audit logs record events and state changes. They are designed for operations and after-the-fact review, are typically mutable by sufficiently privileged accounts, and capture outcomes rather than reasoning. Even a well-secured, write-once audit log records that a decision occurred — not the evidence, alternatives, policy, and confidence that produced it.

Distributed tracing

Distributed tracing reconstructs the path of a request across services and measures latency and failure. It answers where time was spent and where errors occurred. It does not capture the semantic content of a decision or render it reproducible.

Observability and metrics

Observability aggregates signals to characterize system health. It is statistical and system-oriented. It cannot reconstruct an individual decision.

Explainability

Explainability approximates model behavior. It is probabilistic, model-specific, and post-hoc, and it does not bind a specific decision to the exact inputs, versions, and policy under which it was made.

The missing capability

No existing layer can answer, for a specific decision: **“What exactly was decided, on what evidence, under which policy and reasoning — and can an independent party reconstruct and verify it?”** CDL is the infrastructure that answers this question.

Chapter 3 — Design Principles

The following principles are normative for any conforming CDL implementation. Subsequent chapters specify the mechanisms that realize them.

Figure 3.1 — CDL Design Principles



Figure 3.1 — CDL design principles.

Immutability. Records are append-only and hash-chained; a committed record **MUST NOT** be modifiable in place without detection.

Replayability. The record **MUST** capture sufficient version-pinned state to reconstruct the decision deterministically.

Verifiability. Integrity **MUST** be verifiable by recomputation from the records and public keys alone.

Non-repudiation. Each record **MUST** be signed by a key whose certificate chains to an enterprise trust anchor.

Least trust. No verification step may depend on trusting the operator's assertions or infrastructure.

Privacy. Evidence **MUST** be minimized, may be redacted, and **MUST** be encrypted at rest.

Scalability. Recording and verification **MUST** scale horizontally to enterprise decision volumes.

Chain of custody. The provenance of every record **MUST** be unbroken and attributable.

Court defensibility. Records **SHOULD** be admissible: reproducible, tamper-evident, and independently verifiable.

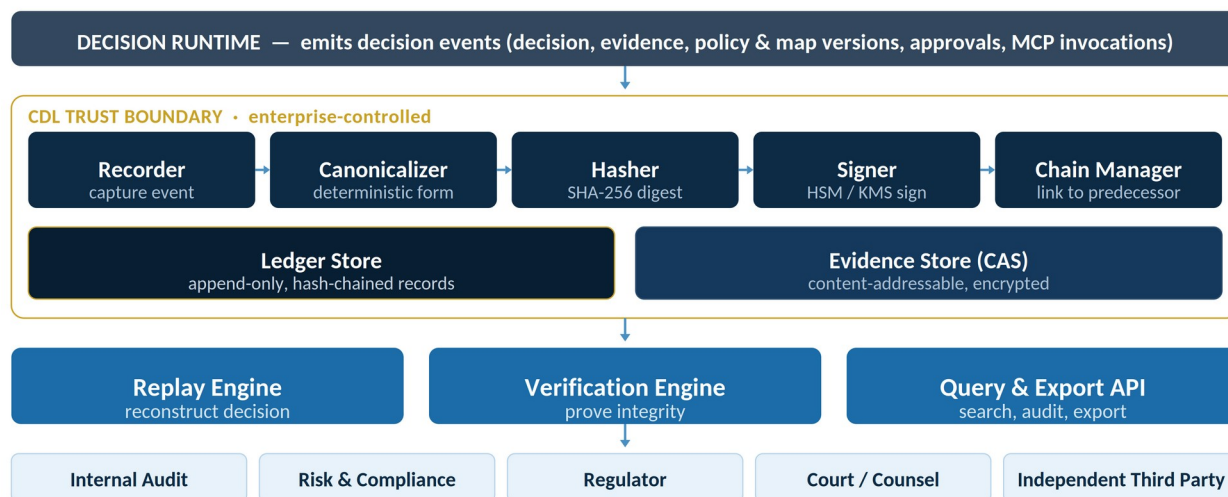
Enterprise ownership. Signing keys and stored data **MUST** remain under enterprise control.

Independent verification. Third parties **MUST** be able to verify records without operator participation.

Chapter 4 — CDL Reference Architecture

CDL comprises an ingestion pipeline, two storage subsystems, and three service engines, all within an enterprise-controlled trust boundary. Decision events enter from the Decision Runtime; verifiers operate outside the boundary using only public material.

Figure 4.1 — CDL Reference Architecture



Verifiers operate outside the trust boundary and require no trust in the operator — only the public verification keys and the records.

Figure 4.1 — CDL reference architecture and trust boundary.

Components and responsibilities

Component	Responsibility
Recorder	Captures decision events emitted by the runtime.
Canonicalizer	Produces a deterministic byte representation of each record.
Hasher	Computes the record digest (SHA-256) and chain linkage.
Signer	Signs the digest using a key held in an HSM or cloud KMS.
Chain Manager	Links each record to its predecessor and orders appends.
Ledger Store	Persists the append-only, hash-chained records.
Evidence Store (CAS)	Stores encrypted evidence, addressed by content hash.
Replay Engine	Reconstructs decisions deterministically from records.
Verification Engine	Verifies hashes, signatures, evidence, and replay.
Query & Export API	Searches, audits, and exports lineage.

Interaction with the Decision Layer

CDL is integral to the Decision Runtime, not a downstream consumer. Recording occurs as part of the decision transaction: a decision is not considered committed until its record is signed and durably appended. The runtime supplies the decision, the resolved policy and decision-map versions, the evidence, the alternatives and confidence, any human approval, and the MCP invocations and downstream actions.

Figure 4.2 — CDL Within the Enterprise Decision Layer



CDL is integral to the runtime, not a downstream consumer. Recording is part of the decision transaction.

Figure 4.2 — CDL within the Enterprise Decision Layer.

Trust boundaries

Three boundaries are defined (Chapter 12). The Decision Runtime is semi-trusted: its agents and models may be compromised. The CDL core is high-assurance and enterprise-controlled; signing keys never leave the HSM/KMS. External verifiers are untrusted and rely solely on public verification material. The central security property is that a compromise within the runtime cannot forge, alter, or suppress a committed record undetectably.

Chapter 5 — Decision Record Model

The decision record is the atomic unit of CDL: the structure in which every governed decision is captured. This public edition describes the record conceptually — the logical components it captures and how they bind together — without specifying its internal field layout, data types, or naming, which are implementation details.

Figure 5.1 — Logical Components of a Decision Record



A conceptual view. Field names, data types, and the record layout are implementation details, omitted from this public edition.

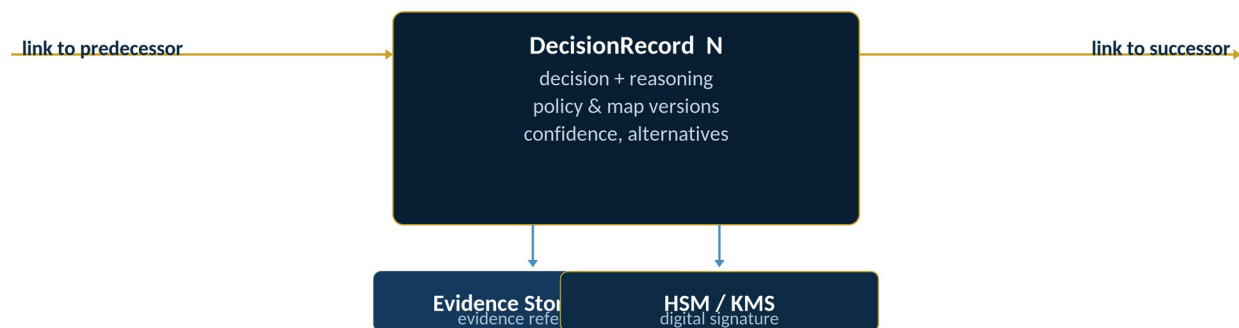
Figure 5.1 — The logical components of a decision record.

What a record captures

Conceptually, a record answers six questions about a governed decision: what was decided and what action followed; the reasoning and the evidence relied upon; the governance context — the policy and decision logic in force; the human oversight exercised, where required; the execution trace of tool invocations and downstream actions; and the integrity binding — the signature and chain linkage that make the record tamper-evident. Together these make a decision reconstructable and verifiable rather than merely logged.

Composition and linkage

A record references its evidence rather than embedding it: large inputs are stored once and addressed by their content, so records stay compact while their contents remain verifiable. Each record links to its predecessor and is bound by a digital signature, placing it in an ordered, tamper-evident chain (Chapters 6 and 7).

Figure 5.2 — Record Composition & Linkage*Figure 5.2 — Record composition and linkage to the chain, evidence store, and signer.***Why the record layout is not specified here**

This public edition is an architectural specification. It defines what a decision record must capture and how it is bound, not the field-level schema by which it is serialized.

The logical components above are stable architectural concepts; their concrete representation is an implementation concern.

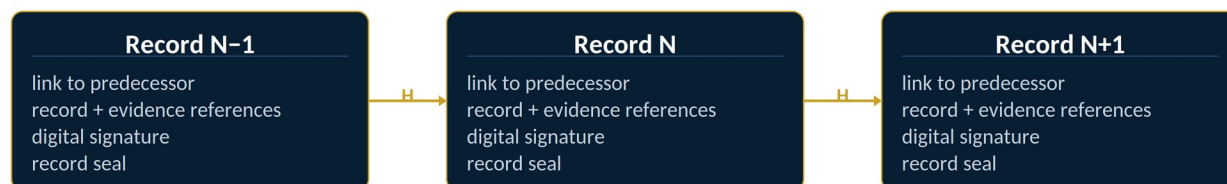
Chapter 6 — Hash Chain Architecture

CDL maintains an append-only hash chain per partition. Each record commits cryptographically to its predecessor, so altering any historical record invalidates every record that follows it. This chapter describes chain construction, batch anchoring, and tamper and fork detection at an architectural level.

Chain construction

Each record carries a cryptographic seal derived from the record together with its predecessor's seal. The derivation is deterministic, so the same record always yields the same seal, and any change to the record — however small — produces a different one. Because each seal incorporates the one before it, the records form an ordered chain in which every entry attests to the entire history that precedes it.

Figure 6.1 — Per-Partition Hash Chain



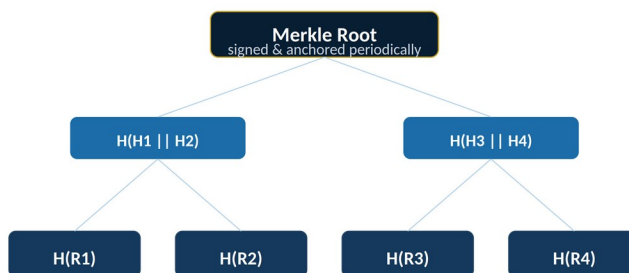
Each record's seal is derived from the record and its predecessor, so every record commits to the one before it.

Figure 6.1 — Per-partition hash chain.

Merkle batch anchoring

To enable efficient inclusion proofs and periodic external anchoring, an implementation SHOULD aggregate record seals into a Merkle tree and sign the root at a fixed interval. A signed Merkle root lets a verifier prove that a specific record was included in the ledger at a point in time without processing the entire chain.

Figure 6.2 — Merkle Batch Anchoring



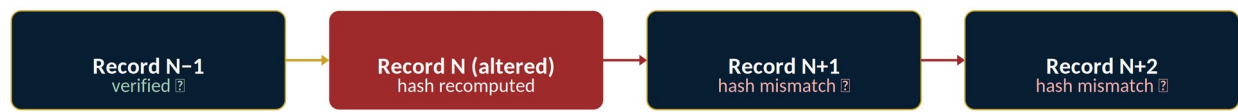
Batches of records are committed under a signed Merkle root, enabling efficient inclusion proofs and periodic external anchoring.

Figure 6.2 — Merkle batch anchoring.

Tamper and fork detection

Verification recomputes each record's seal and confirms that it links correctly to its predecessor. Any altered record produces a mismatching seal, and every subsequent link then fails — localizing the tampering to the first broken link. A fork is detected when two records claim the same predecessor; resolution follows the signed-anchor policy, which designates the anchored branch as authoritative.

Figure 6.3 — Tamper & Fork Detection



Altering record N changes its hash; every downstream link then fails verification. Tampering is localized and detectable. A divergent chain (fork) is detected when two records claim the same predecessor; resolution follows the signed-anchor policy.

Figure 6.3 — Tamper and fork detection.

Architectural requirements (chain)

Each record's seal MUST be derived deterministically from the record and its predecessor.

The ledger MUST be append-only; in-place modification or deletion MUST NOT be possible.

Implementations SHOULD anchor signed Merkle roots at a defined interval.

Verification MUST recompute and confirm linkage for the verified range.

Chapter 7 — Digital Signatures & PKI

Non-repudiation in CDL derives from digital signatures over each record, produced by keys whose certificates chain to an enterprise trust anchor. Private keys **MUST** reside in a hardware security module (HSM) or an equivalent cloud key-management service and **MUST NOT** be exportable.

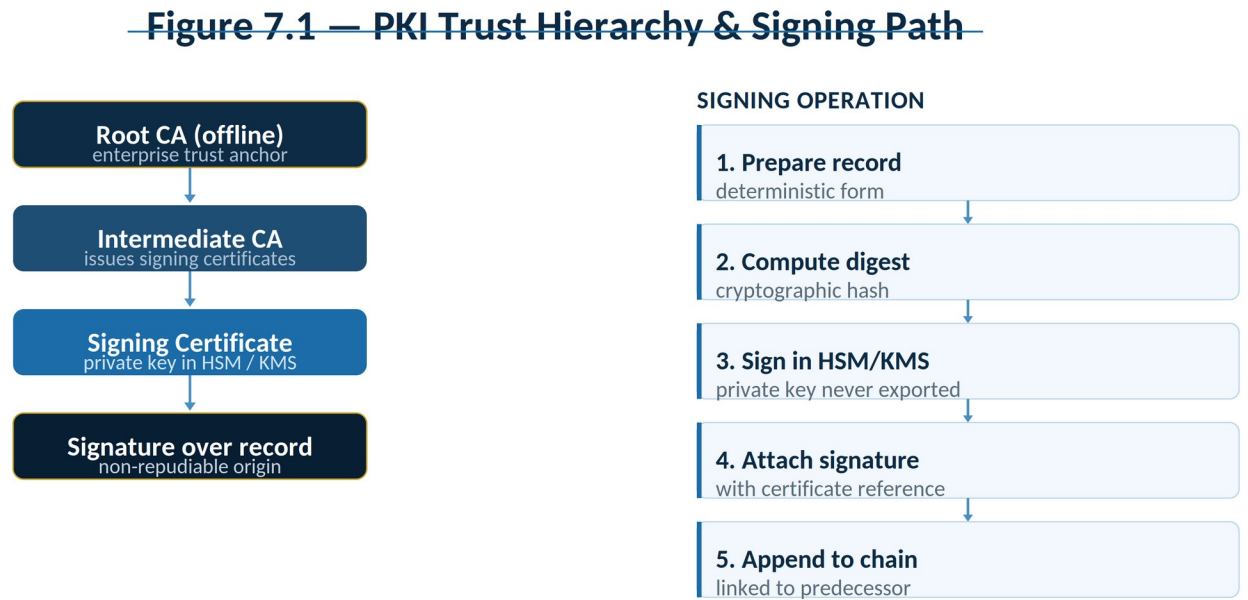


Figure 7.1 — PKI trust hierarchy and signing path.

Trust anchors and certificate hierarchy

An enterprise root CA, held offline, serves as the trust anchor. It issues an intermediate CA, which issues signing certificates bound to keys in the HSM/KMS. Verifiers validate a record's signature against the certificate chain up to the root. The root certificate is the only material a verifier must obtain through a trusted channel.

Key custody

Platform	Key custody mechanism
Azure	Azure Key Vault Managed HSM (FIPS 140-2 Level 3).
AWS	AWS KMS, or CloudHSM for dedicated single-tenant HSMs.
Google Cloud	Cloud KMS, or Cloud HSM for hardware-backed keys.
On-premises / air-gapped	Enterprise HSM appliance; keys never leave the device.

Key rotation and revocation

Signing keys **MUST** be rotated on a defined schedule using an overlap window, so that no decision is ever recorded without a valid active key. Retired keys remain valid for verifying historically signed records. Revocation is published via CRL or OCSP; signatures produced before revocation remain verifiable against the retained certificate, consistent with their recorded timestamps.

Figure 7.2 — Signing-Key Lifecycle



keys remain valid for verification of historically signed records. Revocation is published via CRL/OCSP; prior signatures remain verifiable against the retained ce
Rotation uses an overlap window so no decision is recorded without a valid active key.

Figure 7.2 — Signing-key lifecycle.

Chapter 8 — Evidence Architecture

Evidence is the set of inputs a decision relied upon. CDL stores evidence in a content-addressable store (CAS) and references it from records by content hash, so the chain remains compact while evidence integrity stays independently verifiable.

Figure 8.1 — Evidence Capture & Content-Addressable Storage

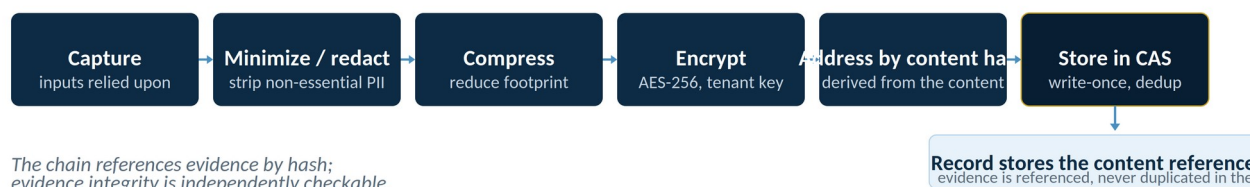


Figure 8.1 — Evidence capture and content-addressable storage.

Capture, minimization, and protection

Evidence is captured at decision time, minimized to what is necessary for reconstruction and proof, optionally redacted to remove non-essential personal data, compressed, and encrypted at rest under a tenant-scoped key. The CAS addresses each blob by its content hash, which provides write-once semantics and natural deduplication.

Privacy and data minimization

Because records reference evidence by hash rather than embedding it, evidence can be subject to independent retention and access controls. Where regulation requires erasure of personal data, the evidence blob MAY be deleted while the record and its hash reference are retained; the record then attests that evidence existed and what its hash was, without retaining the data itself. This reconciles tamper-evidence with data-minimization and erasure obligations.

Normative requirements (evidence)

Evidence **MUST** be referenced by content hash, not embedded in the chained record.

Evidence at rest **MUST** be encrypted under enterprise-controlled keys.

Implementations **SHOULD** minimize and **MAY** redact evidence prior to storage.

Deleting an evidence blob **MUST NOT** alter or invalidate the chained record.

Chapter 9 — Replay Engine

Replay is the deterministic reconstruction of a recorded decision. It re-executes the exact Decision Map version against the exact policy version and the recorded evidence, and compares the reconstructed outcome to the recorded outcome.

Figure 9.1 — Deterministic Replay Architecture

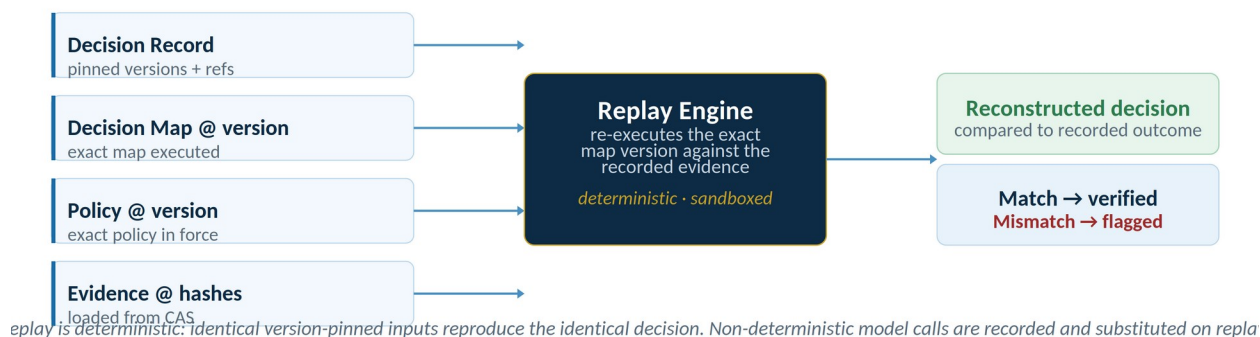


Figure 9.1 — Deterministic replay architecture.

Determinism and version pinning

Replay determinism depends on pinning every input. The record carries the decision-map version, the policy version, and the evidence hashes; the replay engine loads precisely those artifacts. Non-deterministic operations — in particular calls to language models — MUST have their outputs recorded at decision time and substituted during replay, so that replay reconstructs the decision that was made rather than re-sampling a new one.

Figure 9.2 — Replay Sequence

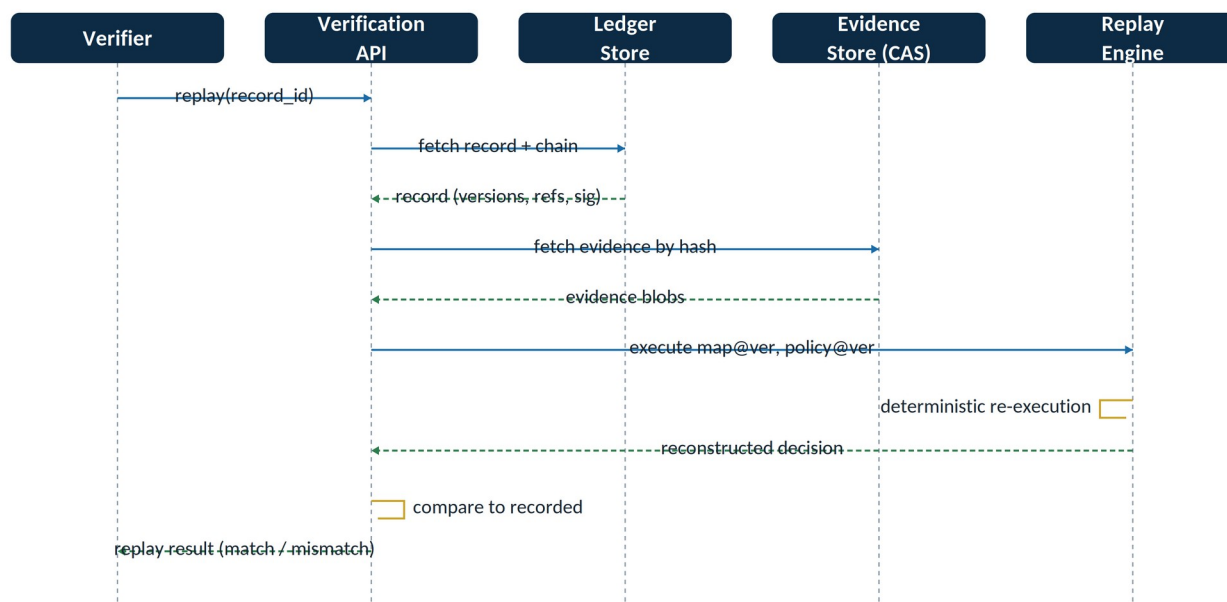
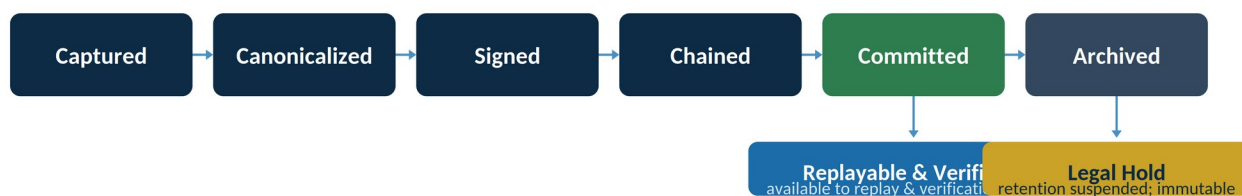


Figure 9.2 — Replay sequence.

Record lifecycle

A record is Captured, Canonicalized, Signed, Chained, and Committed; from the Committed state it is immutable, replayable, and verifiable. Archival changes the storage tier without altering content or verifiability, and legal hold suspends retention expiry.

Figure 9.3 — Decision Record Lifecycle (States)



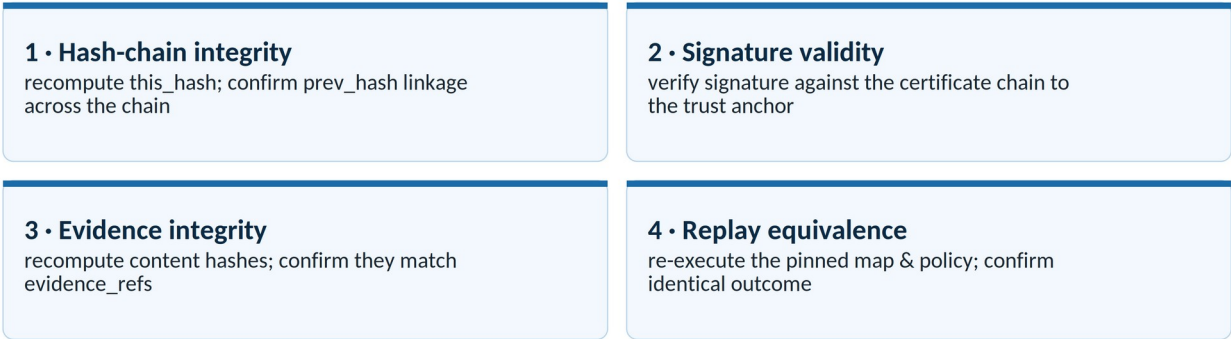
Records are immutable from the Committed state onward. Archival changes storage tier, not content or verifiability.

Figure 9.3 — Decision record lifecycle states.

Chapter 10 — Verification Engine

Verification establishes that a record is intact, authentic, and reproducible. CDL defines four independent checks, all computable from the records and public keys alone.

Figure 10.1 — Verification Architecture: Four Independent Checks



All four checks are computable by any party holding the records and the public verification keys. None requires trust in the operator.

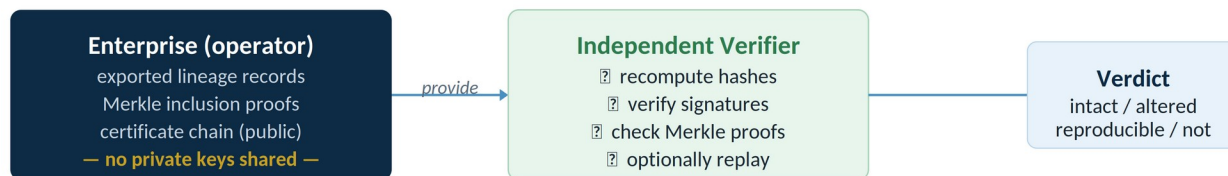
Figure 10.1 — The four independent verification checks.

Check	Procedure	Establishes
Hash-chain integrity	Recompute each record's seal; confirm linkage to its predecessor.	Records are unaltered and correctly ordered.
Signature validity	Verify signature against the certificate chain.	Record origin and non-repudiation.
Evidence integrity	Recompute evidence content hashes; confirm they match the references.	Evidence is the evidence actually used.
Replay equivalence	Re-execute pinned map & policy; compare.	The decision is reproducible as recorded.

Independent and third-party verification

All four checks are designed to be performed by a party other than the operator. The enterprise provides exported records, Merkle inclusion proofs, and the public certificate chain — never private keys. A regulator, court-appointed expert, counterparty, or auditor recomputes hashes, verifies signatures, checks inclusion proofs, and optionally replays decisions, reaching an independent verdict.

Figure 10.2 — Independent Third-Party Verification



*A regulator, court-appointed expert, counterparty, or auditor reaches an independent verdict using only public material.
Trust derives from cryptography and reproducibility — not from the operator's assurances.*

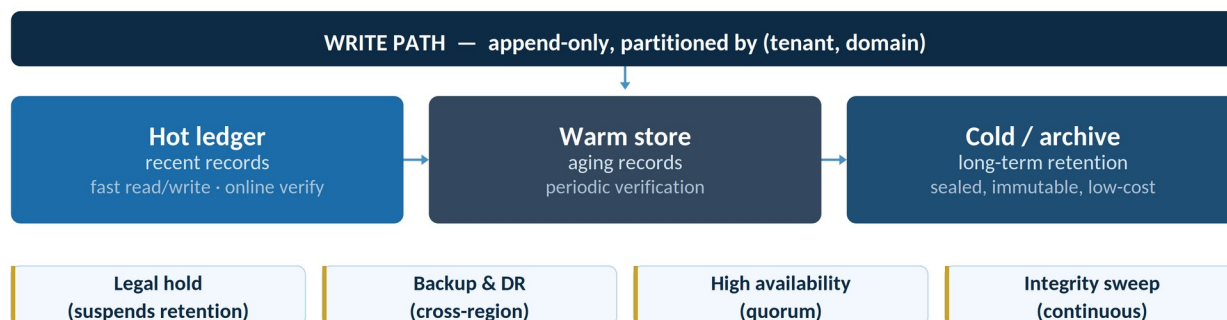
Figure 10.2 — Independent third-party verification.

**Trust derives from cryptography and reproducibility —
not from the operator's assurances.**

Chapter 11 — Storage Architecture

CDL storage is an append-only ledger, partitioned for scale and isolation, with tiered retention and explicit support for legal hold, backup, and disaster recovery.

Figure 11.1 — Storage Architecture



Retention and tiering never mutate records. Archival and legal hold preserve the exact bytes and their verifiability.

Figure 11.1 — Storage architecture.

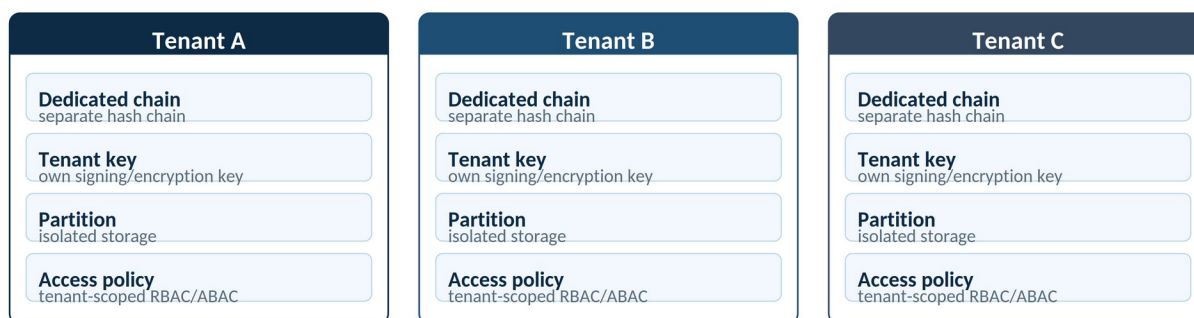
Append-only ledger and partitioning

The write path is append-only and partitioned by tenant and domain. Each partition maintains an independent, ordered hash chain, which preserves causal ordering within a partition while allowing partitions to scale independently. Records move through hot, warm, and cold tiers by age; tiering changes storage cost and latency, never content or verifiability.

Multi-tenancy

Tenants share infrastructure but not trust. Each tenant has a dedicated hash chain, dedicated signing and encryption keys, isolated storage partitions, and tenant-scoped access policy. Cross-tenant verification is impossible by construction, because no tenant holds another's keys.

Figure 11.2 — Multi-Tenant Isolation



Tenants share infrastructure but not trust: separate chains, keys, partitions, and access policy. Cross-tenant verification is impossible by construction.

Figure 11.2 — Multi-tenant isolation.

Retention, legal hold, and resilience

Capability	Requirement
Retention	Records MUST be retained for a configurable, policy-defined period.
Legal hold	A hold MUST suspend retention expiry without altering records.
Backup	Backups MUST preserve exact bytes and remain independently verifiable.
Disaster recovery	Restores MUST pass integrity verification before being marked authoritative.
High availability	The write path SHOULD use quorum durability to avoid loss of committed records.

Chapter 12 — Security Model

This chapter defines the threat model, trust boundaries, and the controls that mitigate each identified attack. The central security property is that no single compromise can forge, alter, or suppress a committed record undetectably.

Figure 12.1 — Threat Model & Trust Boundaries



Security goal: a compromise inside Boundary 1 cannot forge, alter, or suppress a record in Boundary 2 undetectably.

Figure 12.1 — Threat model and trust boundaries.

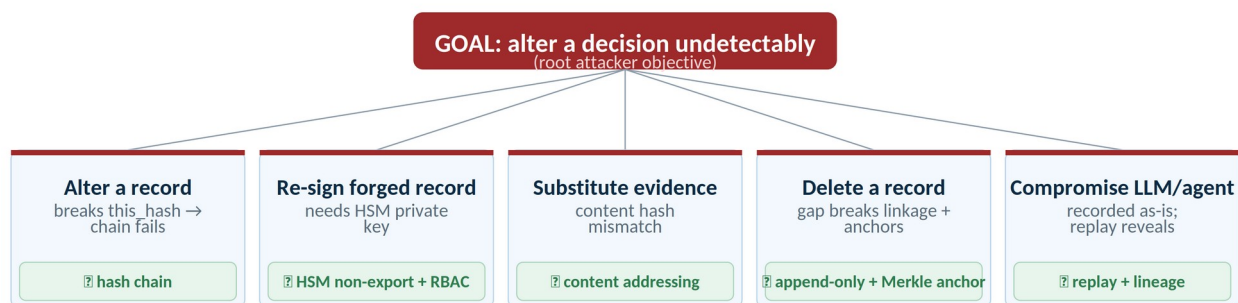
Trust boundaries

Boundary 1, the Decision Runtime, is semi-trusted: LLMs, agents, and planners may be manipulated or compromised. Boundary 2, the CDL core, is high-assurance and enterprise-controlled, with signing keys confined to the HSM/KMS. Boundary 3, external verifiers, is untrusted and operates solely on public material. A compromise in Boundary 1 is recorded faithfully and exposed by replay; it cannot reach into Boundary 2.

Attack tree

The attacker's goal is to alter a decision undetectably. Each path to that goal is independently mitigated.

Figure 12.2 — Attack Tree: Forge or Alter a Decision Record



Every leaf is mitigated by an independent control. No single compromise achieves the goal undetectably.

Figure 12.2 — Attack tree for forging or altering a record.

Threat	Vector	Mitigation
Record tampering	Alter a stored record.	Hash chain: alteration breaks all subsequent links.
Forged signature	Re-sign a forged record.	Private keys non-exportable in HSM; RBAC on signing.
Evidence substitution	Swap evidence after the fact.	Content addressing: hash mismatch is detected.
Record deletion	Remove a record to hide a decision.	Append-only storage; gaps break linkage and anchors.
Insider attack	Privileged operator manipulates data.	Least trust; independent external verification.
Replay attack	Re-submit a captured record.	Unique record identity and chain position.
Key compromise	Steal a signing key.	HSM custody; rotation; revocation via CRL/OCSP.
LLM / agent compromise	Manipulate the decision itself.	Decision is recorded as-made; replay exposes anomalies.
Supply-chain attack	Tamper with CDL components.	Signed builds; verification logic is open and reproducible.

Defense in depth

No single control is load-bearing. Identity and access control, cryptographic integrity, append-only storage, independent verification, and operational controls form independent layers; defeating CDL requires defeating all of them simultaneously.

Figure 12.3 — Defense in Depth

Identity & access control

authenticated, least-privilege access to all CDL operations

Cryptographic integrity

hash chaining + digital signatures bind every record

Append-only storage

records cannot be modified or deleted in place

Independent verification

external parties verify without trusting the operator

Operational controls

key rotation, integrity sweeps, monitoring, DR

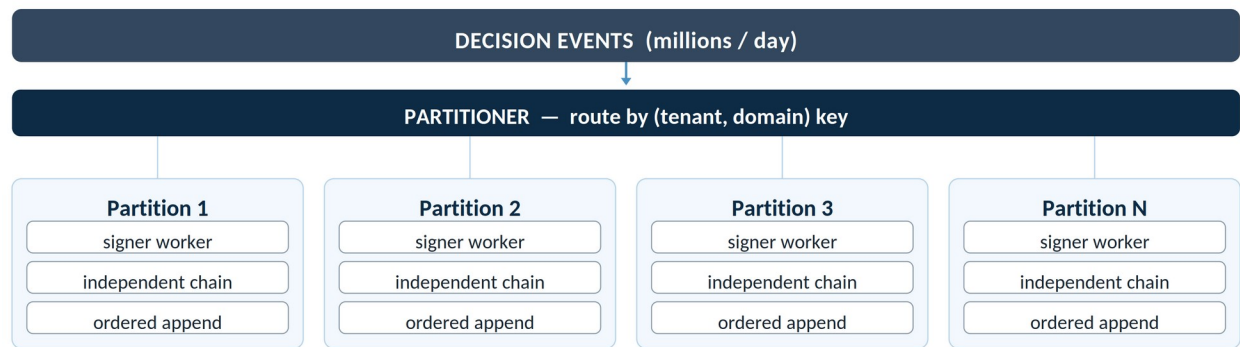
No single control is load-bearing. Defeating CDL requires defeating every independent layer simultaneously.

Figure 12.3 — Defense in depth.

Chapter 13 — Performance Architecture

CDL is designed for enterprise decision volumes — potentially millions of decisions per day. Throughput scales horizontally through partitioning; verification scales through incremental and batch modes.

Figure 13.1 — Horizontal Scaling & Partitioned Write Path



Each partition maintains its own ordered hash chain and signer worker. Throughput scales linearly with partitions; ordering is preserved per key.

Figure 13.1 — Horizontal scaling and partitioned write path.

Write-path scaling

Decision events are partitioned by tenant and domain. Each partition maintains an independent ordered chain and a dedicated signer worker, so aggregate write throughput scales with the number of partitions while per-key ordering is preserved. Signing is the principal per-record cost; batching signature operations and using KMS throughput tiers keep latency bounded.

Verification scaling

Streaming verification checks each record as it commits, providing low-latency tamper detection at bounded incremental cost. Batch verification re-verifies ranges or full chains on a schedule, using Merkle inclusion proofs for efficiency, and underpins audit and disaster-recovery attestation.

Figure 13.2 — Verification Modes

STREAMING VERIFICATION	BATCH VERIFICATION
Continuous verify each record as it commits	Scheduled sweep re-verify ranges or full chains
Low latency detects tampering quickly	Merkle proofs efficient inclusion checks
Bounded cost incremental hash + signature check	Audit / DR full integrity attestation

Figure 13.2 — Streaming and batch verification.

Concern	Approach
Throughput	Horizontal partitioning by (tenant, domain); parallel signer workers.
Write latency	Asynchronous evidence storage; synchronous sign-and-append commit.

Concern	Approach
Storage growth	Evidence dedup via content addressing; tiering to cold storage.
Verification cost	Incremental streaming checks; Merkle proofs for batch.
Read / query	Indexed metadata; chain reads bounded by partition.

Chapter 14 — Compliance Mapping

CDL provides a control substrate that supports evidence for multiple compliance regimes. It does not, by itself, constitute certification; it provides verifiable artifacts that auditors and regulators can rely upon.

Figure 14.1 — Compliance Mapping

Framework	Relevant requirement	How CDL supports it
SOC 2	integrity & audit-trail criteria	immutable records + independent verification
ISO 27001	logging, integrity, evidence controls	tamper-evident ledger + retention
HIPAA	audit controls, integrity (§164.312)	signed records + access logging in CDL
GDPR	accountability, data minimization	evidence minimization + redaction + erasure of refs
EU AI Act	record-keeping & traceability (high-risk AI)	decision-level lineage + replay
NIST AI RMF	measure & manage, traceability	reconstructable, verifiable decisions
PCI DSS	audit trails, integrity monitoring	append-only logs + integrity verification

CDL provides a control substrate; it supports compliance evidence but does not by itself constitute certification.

Figure 14.1 — Compliance mapping.

Framework	Relevant area	CDL contribution
SOC 2	Processing integrity; audit trail.	Immutable, independently verifiable decision records.
ISO 27001	Logging, integrity, evidence (A.12, A.18).	Tamper-evident ledger with defined retention.
HIPAA	Audit controls; integrity (§164.312).	Signed records; access logging; integrity verification.
GDPR	Accountability; data minimization; erasure.	Evidence minimization, redaction, and erasure-by-reference.
EU AI Act	Record-keeping & traceability for high-risk AI.	Per-decision lineage with deterministic replay.
NIST AI RMF	Measure / Manage; traceability.	Reconstructable, verifiable decisions.
PCI DSS	Audit trails; integrity monitoring.	Append-only records; continuous integrity checks.

Scope note

CDL supplies verifiable evidence, not attestation. Certification against any framework remains the responsibility of the enterprise and its assessors; CDL strengthens the evidentiary basis on which those assessments rely.

Chapter 15 — Legal Defensibility

CDL is designed so that a recorded decision can serve as reliable evidence. Defensibility rests not on the operator's testimony but on reproducibility and tamper-evidence that an independent expert can confirm.

Figure 15.1 — Chain of Custody to Court Admissibility



Each link is documented and verifiable. Admissibility rests on reproducibility and tamper-evidence, not on the operator's testimony alone.

Figure 15.1 — Chain of custody to court admissibility.

Chain of custody

Each record is signed at creation, sealed in append-only storage, and preserved under retention or legal hold. Its provenance is unbroken and attributable. When produced in discovery, the record carries everything needed for an independent party to verify it.

Independent verification and expert testimony

An expert witness does not need to trust the enterprise. Given the exported records, the public certificate chain, and the open verification procedure, the expert recomputes hashes, verifies signatures, checks inclusion proofs, and may replay the decision — then testifies to the result of an independent computation. This is materially stronger than testimony about the contents of a conventional log.

Discovery, retention, and regulatory review

Requirement	How CDL addresses it
Authenticity	Signatures chaining to an enterprise trust anchor.
Integrity	Hash chain and Merkle anchoring; tamper is detectable.
Completeness	Append-only storage; gaps are detectable.
Reproducibility	Deterministic replay of the exact decision.
Independence	Verification requires no operator trust or participation.
Retention / hold	Policy-driven retention; legal hold suspends expiry.

Chapter 16 — Decision Interfaces

CDL exposes a small set of operations. This public edition describes them as architectural capabilities rather than a wire-level API specification; the concrete request and response formats are implementation details. Every operation is authenticated and authorized, and the verification and export operations are additionally available in a read-only form suitable for external verifiers.

Figure 16.1 — Decision Recording Transaction

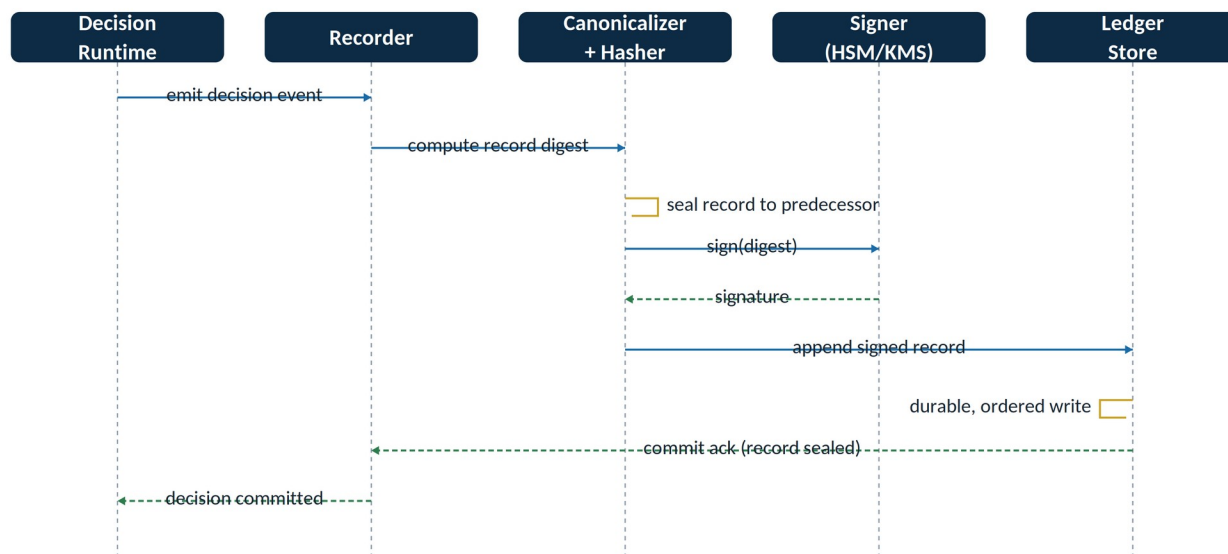


Figure 16.1 — The decision recording transaction.

Core operations

Capability	What it does
Record a decision	Accepts a governed decision and commits a signed, chained record.
Verify a decision	Returns the result of the four integrity checks (Chapter 10).
Replay a decision	Reconstructs the decision deterministically and compares it to the record.
Export lineage	Produces a signed bundle for independent, offline verification.
Search lineage	Locates records by domain, time, or correlation.
Audit a decision	Returns a record together with its evidence and chain context.
Bulk verification	Verifies a range or set of records and reports per-record status.

The recording operation is synchronous with the decision: a decision is not considered committed until its record is signed and durably appended (Figure 16.1). Verification and replay are read-only and may be performed by the enterprise or, using exported material, by an independent party.

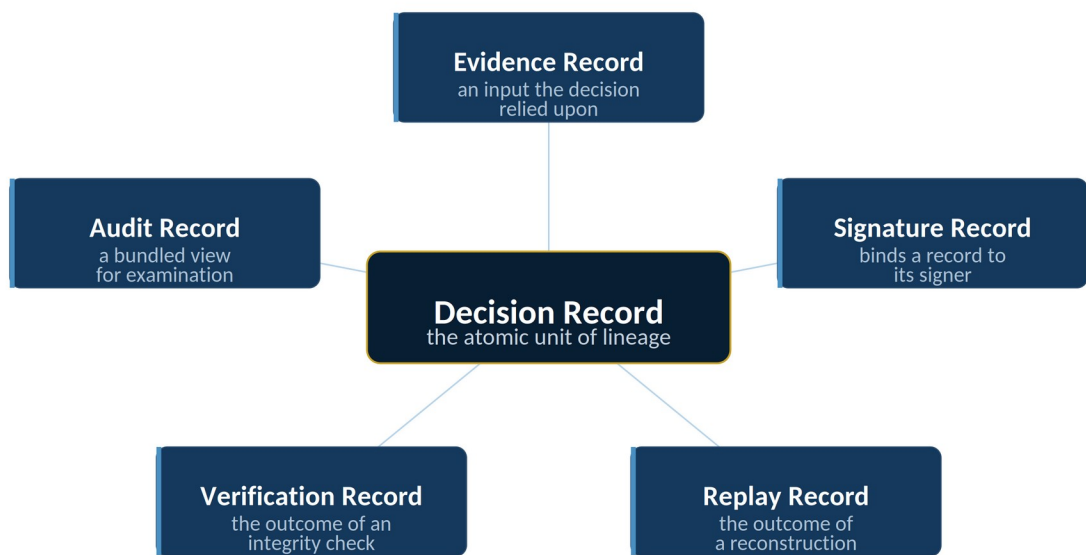
Stable capabilities, unspecified wire format

These capabilities are architecturally stable. Their concrete interface — endpoints, payloads, and encodings — is an implementation concern and is intentionally not specified in this public edition.

Chapter 17 — The Record Family

Beyond the Decision Record (Chapter 5), CDL produces several related record types. This public edition describes each conceptually — the role it plays in producing verifiable lineage — without specifying its fields, types, or serialization, which are implementation details.

Figure 17.1 — The CDL Record Family



Each type plays a distinct role in producing verifiable lineage. Their internal structure is not specified in this public edition.

Figure 17.1 — The CDL record family.

The record types

Record type	Role
Decision Record	The atomic unit of lineage: a single governed decision and its full context.
Evidence Record	Describes one input the decision relied upon, addressed by its content.
Signature Record	Binds a record to the key that signed it, establishing non-repudiation.
Replay Record	Captures the outcome of reconstructing a decision — match or mismatch.
Verification Record	Captures the outcome of the integrity checks performed on a record.
Audit Record	A derived, non-mutating view bundling a record with its evidence and chain context.

Each type exists to make a specific property independently checkable: evidence integrity, signer authenticity, reproducibility, and chain integrity. Producing an audit view never alters the underlying records — examination is always read-only.

Why schemas are not published here

This is an architectural specification, not an implementation specification.

The record types and their roles are stable architectural concepts; their field-level schemas and encodings are

implementation details, intentionally omitted from this public edition.

Chapter 18 — Operational Guidance

This chapter provides operational guidance for running CDL in production. It is advisory except where normative keywords appear.

Figure 18.1 — CDL Operations



Figure 18.1 — CDL operations.

Monitoring and metrics

Operators SHOULD monitor write latency, commit throughput, chain depth per partition, streaming-verification status, signer/KMS availability, and evidence-store utilization. A failed integrity check MUST raise an alert and SHOULD trigger isolation of the affected partition pending investigation.

Metric	Signal	Suggested action on breach
Commit latency (p95)	Sign-and-append time.	Scale signer workers; check KMS throughput.
Verification failures	Any invalid hash/signature.	Alert, isolate partition, begin incident response.
Chain depth / lag	Unverified backlog.	Increase streaming-verify capacity.
Key age	Time since rotation.	Initiate rotation within the overlap window.
Evidence growth	CAS utilization.	Tier to cold storage; confirm dedup.

Key rotation and incident response

Key rotation MUST use an overlap window and SHOULD be performed under dual control. On a suspected tamper or key-compromise event, operators SHOULD: isolate the affected partition; run full-range verification to bound the impact; preserve evidence under legal hold; rotate or revoke affected keys; and produce a verification report. Because verification is independent, the integrity of unaffected records can be re-established by recomputation rather than by assertion.

Retention, replay, and disaster recovery

Retention is policy-driven; legal hold suspends expiry. Replay operations are run on demand for audit and investigation and SHOULD be rate-limited and access-controlled. Disaster-recovery restores MUST pass full integrity verification before being designated authoritative. Verification schedules SHOULD combine continuous streaming checks with periodic full-chain batch attestation, plus an external attestation cadence appropriate to the regulatory context.

Operational invariants

No decision is committed without a valid active signing key.

No retention or tiering action alters record content.

Every restore is verified before it is trusted.

Every integrity failure is alerted, bounded, and reported.

Chapter 19 — Deployment Architectures

CDL is cloud-neutral. The same architecture maps onto the primitives of each major cloud and onto private, hybrid, and air-gapped environments; records and verification behave identically across deployments. This chapter describes the deployment patterns at an architectural level — not as a prescriptive build for any one platform.

Figure 19.1 — Cloud Reference Mappings

Capability	Azure	AWS	Google Cloud
Signing / keys	Azure Key Vault Managed HSM	AWS KMS / CloudHSM	Google Cloud KMS / HSM
Ledger store	Azure Storage (immutable blob)	S3 Object Lock (WORM)	GCS retention / Bigtable
Evidence (CAS)	Azure Blob + CMK	S3 + SSE-KMS	GCS + CMEK
Eventing	Azure Service Bus / Event Hubs	Amazon MSK / Kinesis	Google Pub/Sub
Compute	AKS	EKS	GKE
Verification	confidential compute (opt.)	Nitro Enclaves (opt.)	Confidential VMs (opt.)

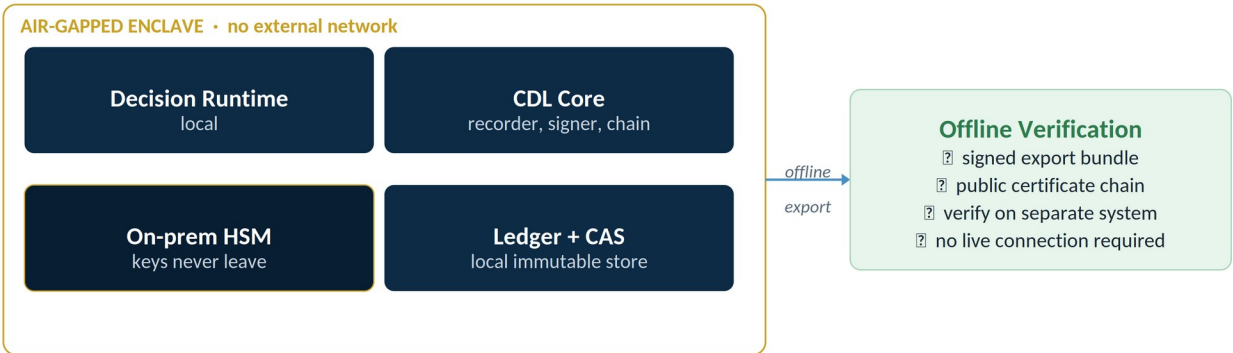
The architecture is cloud-neutral; only the primitives differ. Records and verification behave identically across providers.

Figure 19.1 — Cloud-neutral deployment: the same architecture across providers.

Air-gapped deployment

In high-assurance or classified environments, CDL operates with no external connectivity. Keys reside in an on-premises HSM; records and evidence are stored locally. Verification is performed offline: the enterprise exports a signed bundle and the public certificate chain, and an independent party verifies it on a separate system. Verifiability does not depend on connectivity.

Figure 19.2 — Air-Gapped Deployment



Records are exported as signed bundles and verified on an independent system. Verifiability does not depend on connectivity.

Figure 19.2 — Air-gapped deployment.

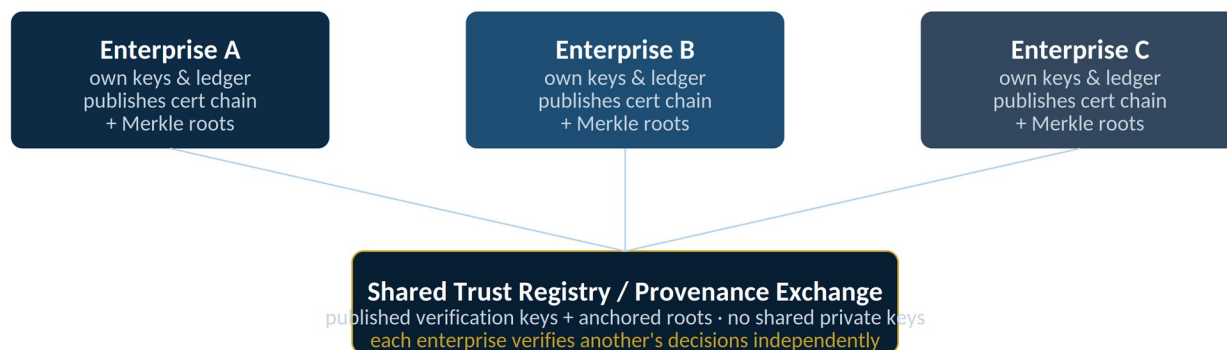
Deployment tradeoffs

Deployment	Primary benefit	Principal tradeoff
Azure / AWS / GCP	Managed HSM/KMS and immutable storage.	Provider trust scope; egress and cost management.
Private cloud	Full control of keys and data residency.	Operational burden of HSM and storage.
Hybrid	Sensitive keys on-prem; scale in cloud.	Connectivity and key-boundary complexity.
Air-gapped	Maximum assurance; no external exposure.	Manual export for external verification.
Single-tenant	Strong isolation; simplest trust model.	Lower infrastructure efficiency.
Multi-tenant	Efficient at scale; per-tenant isolation.	Requires rigorous key and partition separation.

Chapter 20 — Future Extensions

This chapter describes directions that extend CDL beyond a single enterprise. They are presented as architectural intent, not as normative requirements of this version.

Figure 20.1 — Federated Cross-Enterprise Verification



Future direction: cross-enterprise decision provenance with no shared secrets — verification federates while keys remain sovereign.

Figure 20.1 — Federated cross-enterprise verification.

Federated CDL and cross-enterprise trust

Federation allows one enterprise to verify another's decisions without either party sharing private keys. Each enterprise publishes its certificate chain and anchored Merkle roots to a shared trust registry; a counterparty then verifies records independently using only that public material. Trust federates while keys remain sovereign.

Cross-runtime and multi-cloud verification

Because verification depends only on records and public keys, decisions recorded by different runtimes or in different clouds can be verified by a common procedure. This enables uniform verification across heterogeneous estates and across organizational boundaries in a supply chain.

Decision provenance exchange and standardization

A standardized provenance-exchange format would allow decisions — and their proofs — to move between organizations with their verifiability intact, much as signed software bills of materials move today. The schemas and procedures in this specification are intended to be a basis for such standardization.

Extension	Capability
Federated CDL	Cross-enterprise verification without shared secrets.
Cross-runtime verification	Uniform verification across heterogeneous runtimes.
Multi-cloud verification	Provider-independent integrity and replay checks.
Cross-domain lineage	Linked provenance across Operational Domains.
Provenance exchange	Portable, verifiable decision records between parties.

Extension	Capability
Industry standards	Common schemas and verification procedures.

Closing Statement

Cryptographic Decision Lineage specifies how an enterprise records, stores, protects, replays, and verifies AI-driven decisions. Every committed record is immutable by hash chaining, attributable by digital signature, reconstructable by deterministic replay, and verifiable — by the enterprise, by regulators, by courts, and by independent third parties — using only public material and the records themselves.

These properties are not provided by audit logs, distributed tracing, observability, or model explainability. They constitute a distinct capability with a distinct architecture.

**Cryptographic Decision Lineage is not an audit log.
It is a new category of enterprise proof infrastructure.**

This document specifies version 1.0. The schemas, protocols, and procedures defined here are intended to be implementable as written and to serve as a stable reference for security architects, auditors, regulators, and independent verifiers.