AXILOG

SRS-007

Proof-Carrying Governance & Compliance

Locked v0.3 L7 [D2]

Last updated	27 March 2026
Depends on	SRS-001 · SRS-002 · SRS-003 · SRS-004 · SRS-005 · DVEC-001 · AXIOMA-FRAMEWORK
Compliance	DO-178C · IEC 62304 · ISO 26262
Changelog	1 entry

Contents

Depends on: SRS-001 v0.3 · SRS-002 v0.3 · SRS-003 v0.3 · SRS-004 v0.3 · SRS-005 v1.1 · DVEC-001 v1.3 · AXIOMA-FRAMEWORK v0.4

---

0. GOVERNING PRINCIPLE

> **Governance is not a document. Governance is a proof.**
>
> **A regulator does not receive an assertion. They receive a cryptographically evidenced argument.**

L1–L6 produce deterministic, auditable behaviour. L7 proves it.

The distinction:

Without L7	With L7
”The system behaved correctly"	"Here is the cryptographic proof that it did”
Assertion	Evidence citation
Audit log	Tamper-evident proof chain
Policy document	Deterministic program over committed evidence

---

1. PURPOSE

This document defines the **Proof-Carrying Governance Contract** for the Axioma framework.

It specifies:

- The AX:PROOF:v1 evidence record structure
- Proof construction from evidence citations
- Policy soundness requirements
- Cross-layer verification protocol
- Compliance report generation (Track A and Track B)
- External anchor publication

- Integrity fault handling
- Traceability requirements

Objective: Every governance claim SHALL be a deterministic, independently verifiable proof over committed evidence — not an assertion over system state.

---

2. DEFINITIONS

2.1 Proof-Carrying Assertion

A **proof-carrying assertion** is a governance claim that:

- references specific committed evidence records
- is independently recomputable from those records
- produces identical results for identical evidence

2.2 AX:PROOF:v1

The canonical evidence record representing a governance proof.

2.3 Track A — Safety-Critical Evidence Package

The evidence package for DO-178C, IEC 62304, and ISO 26262 certification. Applies to D1 components only.

2.4 Track B — Enterprise Governance Evidence Package

The evidence package for EU AI Act Article 9, ISO/IEC 42001, FCA PS22/3, and MHRA AI/ML guidance. Applies to D2/D3 components.

2.5 External Anchor

A periodic GPG-signed SHA-256 commitment of all ledger chain heads, published to a public append-only transparency log.

2.6 Governance Integrity Fault

A condition where the governance layer detects an inconsistency between committed evidence and declared system behaviour.

2.7 Compliance Report

A structured, JCS-canonical document summarising the governance state of the system over a declared period, suitable for submission to a regulatory body.

2.8 Mathematical Trace

A structured record linking an oracle observation (L3) through policy evaluation (L4) and agent transition (L5) to a committed ledger entry (L6), with all intermediate evidence citations.

2.9 Evidence Closure Proof

A Merkle root or equivalent cryptographic commitment proving completeness of an evidence set — that no relevant evidence has been omitted.

---

3. DETERMINISM MODEL

3.1 Determinism Class

The governance layer SHALL operate under:

D2 — Constrained Deterministic

3.2 Determinism Definition

SRS-007-SHALL-001

For identical:

- committed evidence records
- declared policy set (canonically ordered and hashed)
- governance configuration (canonically ordered and hashed)

the system SHALL produce:

- identical `AX:PROOF:v1` records
- identical compliance report content

Verification: replay verification, cross-platform identity test

---

3.2.1 Configuration Canonicality

SRS-007-SHALL-044

Added in v0.2 per F010.

The policy set and governance configuration SHALL be:

- RFC 8785 (JCS) canonicalised
- sorted by policy_id / config_key lexicographically
- hashed (SHA-256) prior to any evaluation
- committed to the ledger at initialisation

Verification: inspection, hash stability test

3.3 Evidence Closure

SRS-007-SHALL-002

All governance evaluations SHALL operate exclusively on:

- committed evidence records (`AX:STATE:v1` , `AX:TRANS:v1` , `AX:OBS:v1` , `AX:POLICY:v1` , `AX:PROOF:v1`)
- static governance configuration

Governance SHALL NOT depend on:

- live system state
- external services
- system clock (unless admitted as oracle per §9.2.1)
- non-committed data

Verification: inspection, static analysis

---

4. AX:PROOF:v1 RECORD

4.1 Proof Record Requirement

SRS-007-SHALL-003

Every governance proof SHALL be committed as an `AX:PROOF:v1` record to the L6 ledger before influencing any downstream decision.

4.2 Required Fields

SRS-007-SHALL-004

Amended in v0.3 per F014.

Each `AX:PROOF:v1` record SHALL contain:

Field	Type	Description
`claim`	string	The governance claim being proven
`commitment`	bytes32	Domain-separated commitment per SRS-001 §4.3
`evidence_ordering`	enum	Ordering mode for evidence_refs (see §4.4.2)
`evidence_refs`	array[bytes32]	Ordered list of evidence record hashes (see §4.4.1)
`ledger_seq`	uint64	Sequence number of this proof record
`ordering_metadata`	object/null	Required if evidence_ordering = DECLARED
`prev_chain_head`	bytes32	Prior L6 chain head before this commit
`proof_hash`	bytes32	SHA-256 over canonical payload (preimage commitment)
`proof_type`	enum	Classification of proof (see §4.3)
`result`	enum	`VALID` | `INVALID` | `INTEGRITY_FAULT`
`rule_id`	string	Governance rule identifier
`schema_version`	string	Always `"AX:PROOF:v1"`
`violation`	enum/null	Violation type if result is INVALID

Field Order: Fields MUST appear in lexicographic order per RFC 8785 (JCS):

```
claim, commitment, evidence_ordering, evidence_refs, ledger_seq,
ordering_metadata, prev_chain_head, proof_hash, proof_type, result,
rule_id, schema_version, violation
```

4.2.1 Cryptographic Binding Fields

SRS-007-SHALL-045

Amended in v0.3 per F011, F012.

The cryptographic binding fields SHALL be computed as:

proof_hash:

```
proof_hash = SHA-256(canonical_payload_with_proof_hash_omitted)
```

Where `canonical_payload_with_proof_hash_omitted` is the RFC 8785 canonicalised record with the `proof_hash` field **entirely omitted** (not present as empty string, not present as null — the key itself is absent from the JSON object).

Forbidden alternatives:

```
❌ proof_hash set to "" (empty string)
❌ proof_hash set to null
❌ proof_hash set to zero ("000...000")
```

Rationale: JCS canonicalisation treats `""`, `null`, and omitted fields differently. Only field omission produces unambiguous canonical form across all implementations.

4.2.2 Commitment Payload Encoding

SRS-007-SHALL-054

Added in v0.3 per F012.

The commitment function payload encoding SHALL be:

commitment:

```
commitment = SHA-256("AX:PROOF:v1" || LE64(byte_length) || utf8_bytes)
```

Where:

- `"AX:PROOF:v1"` is the ASCII domain separation tag (11 bytes)
- `byte_length` is the length in **bytes** (not characters) of the UTF-8 encoded canonical JSON
- `utf8_bytes` is the UTF-8 encoded RFC 8785 canonical JSON payload
- `||` denotes byte-wise concatenation

Encoding Rules:

Component	Encoding
Domain tag	ASCII bytes
Length	Little-endian 64-bit unsigned integer
Payload	UTF-8 encoded JSON (RFC 8785 canonical)

Verification: byte-level hash stability test, cross-platform harness

---

4.2.3 Chain Head Reference

SRS-007-SHALL-055

Added in v0.3 for completeness.

prev_chain_head: The SHA-256 hash of the L6 chain head immediately prior to this record's commitment.

Purpose: These fields make proofs self-authenticating and independently verifiable outside L6 context.

Verification: hash stability test, independent verification test

---

4.3 Proof Types

SRS-007-SHALL-005

The proof type set SHALL be closed within a schema version:

Type	Description
`ANCHOR_PUBLICATION`	External anchor commitment
`COMPLIANCE_SUMMARY`	Compliance report proof
`CROSS_LAYER_VERIFY`	Cross-layer chain verification
`POLICY_SOUNDNESS`	Policy evaluation over evidence
`REPLAY_EQUIVALENCE`	Replay produces identical results
`SUBSTRATE_CERT`	L1 substrate certification
`WEIGHT_BINDING`	L2 model identity verification

Any proof type outside this closed set SHALL be rejected as non-conformant.

4.3.1 Proof Type Versioning

SRS-007-SHALL-056

Added in v0.3 per F019.

The proof type enumeration SHALL support versioned extension:

Namespace: `AX:PROOF:v{N}`

Extension Rules:

1. New proof types require a new schema version (`AX:PROOF:v2` , etc.)
2. Schema version upgrade requires explicit migration specification
3. Backward compatibility: v1 proofs remain valid in v2 systems
4. Forward compatibility: v2 proofs rejected by v1 systems with `INTEGRITY_FAULT`

Migration Protocol:

```
1. Define new proof types in AX:PROOF:v{N+1}
2. Publish migration specification
3. Deploy systems supporting both versions
4. Deprecate old version after transition period
```

Verification: schema validation, version compatibility test

4.4 Evidence Reference Requirement

SRS-007-SHALL-006

Every `AX:PROOF:v1` record SHALL cite at minimum one committed evidence record in `evidence_refs`.

A proof with an empty `evidence_refs` array is non-conformant. Assertions without evidence citations are not proofs.

4.4.1 Evidence Reference Encoding

SRS-007-SHALL-046

Amended in v0.3 per F014.

The `evidence_refs` array SHALL conform to:

Hash Algorithm: SHA-256 (32 bytes)

Encoding: Lowercase hexadecimal (64 characters per hash)

Ordering: Controlled by the `evidence_ordering` field (see §4.4.2)

Example:

```
"evidence_refs": [
  "a1b2c3d4e5f67890123456789012345678901234567890123456789012345678901234abcd",
  "fedcba9876543210fedcba9876543210fedcba9876543210fedcba9876543210"
]
```

Verification: schema test, property test

4.4.2 Evidence Ordering Mode

SRS-007-SHALL-057

Added in v0.3 per F014.

Each `AX:PROOF:v1` record SHALL include an `evidence_ordering` field:

Field	Type	Description
`evidence_ordering`	enum	Ordering mode for `evidence_refs`

Ordering Modes:

Mode	Definition	Use Case
`LEX`	Lexicographic sort by hash	Default, no semantic order
`TEMPORAL`	Ascending by `ledger_seq`	Causal chain proofs
`DECLARED`	Order matches `ordering_metadata`	Custom semantic order

If `DECLARED`: The record MUST include an `ordering_metadata` field:

```
"ordering_metadata": {
  "description": "Cross-layer verification sequence",
  "key_field": "ledger_seq",
  "direction": "ascending"
}
```

Default: If `evidence_ordering` is omitted, `LEX` is assumed.

Forbidden:

```
❌ Ordering rule declared in free-form claim field
❌ Implicit ordering without declaration
```

Verification: schema test, ordering verification test

4.5 Canonical Format

SRS-007-SHALL-007

All `AX:PROOF:v1` records SHALL be:

- RFC 8785 (JCS) canonicalised

- bit-identical for identical inputs
- committed using the domain-separated commitment function (SRS-001 §4.3)

Verification: JCS linter, hash stability test

---

5. POLICY SOUNDNESS REQUIREMENT

5.1 Soundness Definition

SRS-007-SHALL-008

All governance policies SHALL satisfy the Policy Soundness Requirement:

1. Deterministic Evaluation Given identical evidence inputs, the policy produces identical output.

2. Evidence Closure The policy references only committed evidence records. No external state permitted.

3. Independent Verifiability A third party with access to the evidence set can recompute the policy and obtain the same result.

Formal model:

```
Let P : E → R where E = evidence records, R = policy result
Then: ∀ E : P(E) = deterministic
```

5.2 Policy Violation

SRS-007-SHALL-009

A policy is non-conformant if:

- It depends on external state not present in evidence records
- It produces non-deterministic output
- It cannot be independently recomputed from evidence

Non-conformant policies SHALL be rejected at governance initialisation.

Verification: inspection, property test

5.3 Policies as Programs

SRS-007-SHALL-010

Governance policies ARE deterministic programs over committed evidence — not configuration documents.

The distinction:

```
Non-conformant: "Policy says no PII in LLM inputs"
Conformant:     P(AX:OBS:v1 records) → {PERMITTED, BREACH}
                where P is deterministic and evidence-closed
```

Verification: inspection, property test

---

6. CROSS-LAYER VERIFICATION PROTOCOL

6.1 Verification Chain

SRS-007-SHALL-011

The governance layer SHALL verify the complete cross-layer evidence chain. Verification SHALL proceed in layer order:

```
Step 1: L1 Substrate Certification
Step 2: L2 Weight-to-Evidence Binding
Step 3: L3 Observation Integrity
Step 4: L4 Policy Soundness Verification
Step 5: L3→L4 Observation-to-Policy Binding
Step 6: L4→L5 Breach-to-Transition Enforcement
Step 7: L5→L6 Pre-Commit Ordering
Step 8: Full Chain Replay Verification
```

Each step SHALL produce an `AX:PROOF:v1` record.

6.1.1 Proof-Before-Execution Invariant

SRS-007-SHALL-047

Amended in v0.3 per F017.

All cross-layer verification proofs MUST be committed before dependent layer execution proceeds.

Invariant:

```
∀ verification step V producing proof P:
  commit(P) → then dependent_action
  NOT: dependent_action → then commit(P)
```

Violation Behaviour:

Any execution that proceeds before its verification proof is committed SHALL be treated as `INTEGRITY_FAULT`.

Rationale: Prevents transient violations that could be "fixed" post-hoc. Ensures audit completeness with no gaps.

Verification: integration test, state machine test

6.1.2 Atomicity Model

SRS-007-SHALL-058

Added in v0.3 per F017.

The proof-before-execution invariant SHALL be enforced via one of:

Option A — Single-Threaded Sequencing:

All governance operations execute in a single thread with deterministic sequencing:

```
1. compute_proof(V)
2. commit_to_ledger(P)
3. verify_commit_success()
4. execute_dependent_action()
```

No concurrent execution of governance operations permitted.

Option B — Atomic Commit Boundary:

Proof commit and dependent action execute within an atomic transaction boundary:

```
BEGIN ATOMIC
  commit_to_ledger(P)
  execute_dependent_action()
COMMIT
```

Failure at any step rolls back all changes.

Deployment Declaration:

The atomicity model MUST be declared in the configuration manifest.

Forbidden:

```
❌ Race condition between commit and action
❌ Non-atomic interleaving of proof and execution
❌ Optimistic execution with deferred proof commit
```

Verification: concurrency test, state machine test

6.2 Step 1 — L1 Substrate Certification

SRS-007-SHALL-012

Governance SHALL verify that certifiable-inference (L2) was compiled against libaxilog (L1) with no forbidden arithmetic patterns.

Proof: `forbidden-pattern-scan` result committed as `AX:PROOF:v1` with `proof_type = SUBSTRATE_CERT`.

Verification: build system audit, static analysis artefact

6.3 Step 2 — L2 Weight-to-Evidence Binding

SRS-007-SHALL-013

Governance SHALL verify:

```
WeightHash(L2) ≡ ModelID(L3.AX:OBS:v1.model_id)
```

The model that produced the inference MUST match the model whose weights were quantised and certified.

Proof: Weight hash comparison committed as `AX:PROOF:v1` with `proof_type = WEIGHT_BINDING`.

Violation: Any `model_id` mismatch SHALL produce `result = INTEGRITY_FAULT`.

Verification: inspection, integration test

6.3.1 Weight Hash Specification

SRS-007-SHALL-048

Added in v0.2 per F008.

The weight hash SHALL be computed as:

```
weight_hash = SHA-256(canonical_weight_representation)
```

Where `canonical_weight_representation` is:

- The quantised weight tensor in DVEC-001 canonical form
- Serialised as contiguous Q16.16 values in row-major order
- Little-endian byte encoding
- No padding between layers
- Layer order fixed by model architecture declaration

Reference: DVEC-001 v1.3 §12.2 (quantisation canonical form)

Verification: hash stability test, cross-platform harness

6.4 Step 3 — L3 Observation Integrity

SRS-007-SHALL-014

Governance SHALL verify the `obs_hash` in every `AX:OBS:v1` record:

```
obs_hash = SHA-256(canonical_record_without_obs_hash)
```

Any `obs_hash` mismatch SHALL produce `result = INTEGRITY_FAULT`.

Verification: hash stability test, property test

6.5 Step 4 — L4 Policy Soundness Verification

SRS-007-SHALL-015

Governance SHALL verify that all active policies satisfy SRS-007-SHALL-008 (Policy Soundness Requirement) before certifying any compliance report.

Verification: property test, inspection

6.6 Step 5 — L3→L4 Observation-to-Policy Binding

SRS-007-SHALL-016

Governance SHALL verify the binding:

```
AX:POLICY:v1.obs_ledger_seq → AX:OBS:v1.ledger_seq
```

(Per SRS-004-SHALL-044)

Every policy evaluation MUST be unambiguously traceable to the exact observation it evaluated.

Any broken binding SHALL produce `result = INTEGRITY_FAULT`.

Verification: integration test, RTM generation

6.7 Step 6 — L4→L5 Breach-to-Transition Enforcement

SRS-007-SHALL-017

Governance SHALL verify:

```
L4:AX:POLICY:v1.result = BREACH
  ⇒
L5:AX:TRANS:v1.next_state ∈ {ALARM, STOPPED}
```

Any transition to a non-safety state following a policy breach SHALL produce `result = INTEGRITY_FAULT`.

Verification: state machine test, integration test

6.8 Step 7 — L5→L6 Pre-Commit Ordering

SRS-007-SHALL-018

Governance SHALL verify the cross-layer ordering invariant:

```
AX:OBS:v1 (ledger_seq N)
  → AX:POLICY:v1 (ledger_seq N+1..N+k)
  → AX:TRANS:v1 (ledger_seq N+k+1)
```

(Per SRS-004-SHALL-016, SRS-003-SHALL-022)

Any ordering violation SHALL produce `result = INTEGRITY_FAULT`.

Verification: property test, integration test

6.9 Step 8 — Full Chain Replay Verification

SRS-007-SHALL-019

Governance SHALL provide replay verification sufficient to prove:

Given:

- Identical genesis state (L0)
- Identical ordered AX:OBS:v1 sequence

The system SHALL reproduce:

- Identical AX:POLICY:v1 records
- Identical AX:TRANS:v1 records

- Identical final state
- Identical AX:PROOF:v1 records

Verification: replay verifier, cross-platform identity test

---

7. MATHEMATICAL TRACE

7.1 Trace Requirement

SRS-007-SHALL-020

For every inference event, governance SHALL produce a Mathematical Trace — a structured record linking the complete evidence chain from oracle input to committed transition.

7.2 Required Trace Fields

SRS-007-SHALL-021

Amended in v0.2 per F006.

Each Mathematical Trace SHALL include:

Field	Type	Encoding	Description
`chain_head`	bytes32	hex lowercase	L6 chain head after all commits
`obs_hash`	bytes32	hex lowercase	AX:OBS:v1 observation hash
`obs_ledger_seq`	uint64	decimal	L3 oracle observation sequence
`policy_results`	array[enum]	—	PERMITTED/BREACH per policy
`policy_seqs`	array[uint64]	ascending	L4 policy evaluation sequences
`proof_ledger_seq`	uint64	decimal	This proof record sequence
`trace_hash`	bytes32	hex lowercase	SHA-256 of canonical trace
`trans_ledger_seq`	uint64	decimal	L5 transition sequence
`trans_next_state`	enum	—	Resulting agent health state
`weight_hash`	bytes32	hex lowercase	L2 model fingerprint

Field Order: Lexicographic per RFC 8785.

7.2.1 Trace Array Ordering

SRS-007-SHALL-049

Added in v0.2 per F006.

Arrays within Mathematical Traces SHALL be ordered as:

policy_seqs: Ascending by `ledger_seq`

policy_results: Corresponding order to `policy_seqs` (i.e., `policy_results[i]` is the result for `policy_seqs[i]`)

Verification: property test, inspection

7.2.2 Trace Hash Computation

SRS-007-SHALL-050

Added in v0.2 per F006.

The `trace_hash` SHALL be computed as:

```
trace_hash = SHA-256(canonical_trace_without_trace_hash)
```

Where `canonical_trace_without_trace_hash` is the RFC 8785 canonicalised trace record with `trace_hash` set to empty string.

The `trace_hash` SHALL be included in the parent `AX:PROOF:v1` record's `evidence_refs`.

Verification: hash stability test

7.3 Trace Canonicality

SRS-007-SHALL-022

Mathematical Traces SHALL be:

- RFC 8785 (JCS) canonicalised
- bit-identical for identical evidence chains
- committed as `AX:PROOF:v1` with `proof_type = CROSS_LAYER_VERIFY`

---

8. COMPLIANCE REPORT GENERATION

8.1 Track A — Safety-Critical Evidence Package

SRS-007-SHALL-023

The Track A evidence package SHALL be generated for systems using certifiable-inference (L2, D1) and SHALL include:

Artefact	Source	Description
Merkle provenance chains	certifiable-*	Data → training → quantisation
368-byte golden reference	certifiable-harness	Cross-platform bit-identity proof
Quantisation error bounds	certifiable-quant	$\epsilon_o = 2^{-17}$ per layer
Conformance test results	certifiable-bench	Correctness-gated performance
Substrate certification	libaxilog	No forbidden patterns
Weight fingerprint	quantize.py	model_id binding proof

Standards addressed: DO-178C, IEC 62304, ISO 26262

8.2 Track B — Enterprise Governance Evidence Package

SRS-007-SHALL-024

The Track B evidence package SHALL be generated for systems using oracle-based inference (L3, D3) and SHALL include:

Artefact	Source	Description
Typed audit ledger	axioma-audit	Complete AX:*:v1 record set
Oracle call records	axioma-oracle	AX:OBS:v1 per LLM interaction
Drift detection reports	certifiable-monitor	TV/JSD/PSI fixed-point results
Policy assertion records	axioma-policy	AX:POLICY:v1 per evaluation
Mathematical traces	axioma-governance	Cross-layer evidence chains
External anchor log	axioma-governance	GPG-signed chain head anchors
Evidence closure proof	axioma-governance	Merkle root of included evidence

Standards addressed: EU AI Act Article 9, ISO/IEC 42001, FCA PS22/3, MHRA AI/ML guidance

---

8.3 Report Trigger Conditions

SRS-007-SHALL-025

Compliance reports SHALL be triggered by events, not arbitrary sequence counts:

Trigger	Report Type
Agent transitions to STOPPED	Incident report
Policy BREACH detected	Breach report
External anchor interval elapsed	Periodic summary
Explicit governance request	On-demand report
System reset from STOPPED	Recovery report

Rationale: Sequence-count triggers (e.g. every 1000 events) are arbitrary and may miss critical events or produce reports at semantically meaningless boundaries.

8.4 Report Canonical Format

SRS-007-SHALL-026

All compliance reports SHALL be:

- RFC 8785 (JCS) canonicalised
- Committed as `AX:PROOF:v1` with `proof_type = COMPLIANCE_SUMMARY`
- Include the L6 chain head at time of report generation
- Include the report hash in the ledger commitment

8.5 Report Independence

SRS-007-SHALL-027

A compliance report SHALL be independently verifiable by a third party holding only:

- The report document
- The public key for GPG anchor verification
- Access to the public ledger or evidence set

No additional context SHALL be required to verify the report's claims.

8.5.1 Evidence Closure Proof Requirement

SRS-007-SHALL-051

Amended in v0.3 per F013.

Every compliance report SHALL include an **Evidence Closure Proof**:

Definition: A Merkle root computed over all evidence records cited in the report, proving completeness of the evidence set.

Requirement:

```
Report MUST include or commit to a complete evidence set
sufficient for independent verification of all claims.
```

Verification: Merkle proof verification, completeness audit

8.5.2 Merkle Tree Construction Algorithm

SRS-007-SHALL-059

Added in v0.3 per F013.

The Evidence Closure Merkle tree SHALL be constructed as follows:

Tree Structure: Binary Merkle tree, left-balanced

Input: Lexicographically sorted list of evidence hashes

Leaf Computation:

```
leaf[i] = evidence_refs[i]  (raw hash, no re-hash)
```

Internal Node Computation:

```
node = SHA-256(left_child || right_child)
```

Where `||` denotes byte-wise concatenation (64 bytes total input).

Odd Node Handling: Duplicate the last node

```
If count(nodes) is odd:
  nodes.append(nodes[-1])  // duplicate last
```

Tree Construction: Left-to-right, bottom-up

```
Level 0: [leaf_0, leaf_1, leaf_2, leaf_3, ...]
Level 1: [SHA-256(leaf_0 || leaf_1), SHA-256(leaf_2 || leaf_3), ...]
...
Level N: [root]
```

Empty Tree: If `evidence_refs` is empty, root = 32 zero bytes

Example (4 leaves):

```
        root
       /    \
    H01      H23
   /   \    /   \
  L0   L1  L2   L3
```

Where:

- `L0..L3` = sorted evidence hashes
- `H01 = SHA-256(L0 || L1)`
- `H23 = SHA-256(L2 || L3)`
- `root = SHA-256(H01 || H23)`

Verification: golden vector test, cross-platform harness

8.6 Golden Reference Inclusion

SRS-007-SHALL-028

All Track A compliance reports SHALL include the certifiable-harness 368-byte golden reference in the evidence package.

Purpose: Proves the model running in production is byte-identical to the model that was audited and certified.

Verification: certifiable-harness integration test

---

9. EXTERNAL ANCHOR PUBLICATION

9.1 Anchor Requirement

SRS-007-SHALL-029

The governance layer SHALL publish periodic cryptographic anchors to a public append-only transparency log.

9.2 Anchor Computation

SRS-007-SHALL-030

Amended in v0.2 per F003.

The anchor hash SHALL be computed as:

```
anchor = SHA-256(
  "AX:ANCHOR:v1" ||
  LE64(anchor_time_seq) ||
  chain_head_hash
)
```

Where:

- `anchor_time_seq` is the `ledger_seq` of the committed Time Oracle observation (`AX:OBS:v1` with `oracle.type = TIME_INJECTION`)
- `chain_head_hash` is the current L6 chain head

9.2.1 Anchor Time as Oracle Input

SRS-007-SHALL-052

Added in v0.2 per F003.

The timestamp used in anchor computation SHALL be:

- Admitted as an `AX:OBS:v1` record with `oracle.type = TIME_INJECTION`
- Committed to the L6 ledger before anchor computation
- Referenced by `ledger_seq` in the anchor record

Rationale: Time is an oracle input. Using raw `timestamp_ns` would introduce hidden external dependency, violating D2 determinism.

Anchor Record Extension:

Field	Type	Description
`anchor_time_seq`	uint64	ledger_seq of Time Oracle AX:OBS:v1

Verification: inspection, replay verification

9.3 Anchor Signing

SRS-007-SHALL-031

Amended in v0.3 per F015.

Each anchor SHALL be:

- GPG-signed with the published SpeyTech governance key
- Published to the transparency log before the next anchor interval
- Appended to `latest-anchor.txt` (append-only)

9.3.1 GPG Signature Determinism Boundary

SRS-007-SHALL-060

Added in v0.3 per F015.

The GPG signature is an **external verification artefact** and SHALL NOT be part of the deterministic proof payload.

Determinism Boundary:

Component	Deterministic	Role
`anchor` hash	✅ YES	Canonical proof
`AX:PROOF:v1` record	✅ YES	Ledger commitment
GPG signature	❌ NO	External verification only

Rationale:

GPG signatures are inherently non-deterministic due to:

- Timestamp inclusion
- Random nonce generation
- Implementation-dependent formatting

Verification Model:

```
1. Deterministic anchor hash committed to L6 (reproducible)
2. GPG signature computed over anchor hash (non-reproducible)
3. Third-party verifies: signature → anchor hash → chain state
```

The anchor hash is the canonical proof. The signature is an external attestation of that proof.

Forbidden:

```
❌ GPG signature included in AX:PROOF:v1 payload
❌ GPG signature used in commitment computation
❌ Determinism claims about GPG output
```

Verification: inspection, replay verification (anchor hash only)

9.4 Anchor Commitment

SRS-007-SHALL-032

Each anchor publication SHALL be committed to the L6 ledger as `AX:PROOF:v1` with `proof_type = ANCHOR_PUBLICATION`.

9.5 Anchor Interval Declaration

SRS-007-SHALL-033

The anchor publication interval SHALL be:

- Declared at system initialisation
- Committed to the configuration manifest (SRS-001-SHALL-007)
- Treated as a system change if modified

Verification: inspection, configuration audit

9.6 Anchor Verification

SRS-007-SHALL-034

A third party holding only the public key and the transparency log SHALL be able to verify that the system state at any anchor point has not been subsequently modified.

---

10. INTEGRITY FAULT HANDLING

10.1 Integrity Fault Definition

SRS-007-SHALL-035

A Governance Integrity Fault occurs when:

- Any cross-layer verification step (§6) produces `result = INTEGRITY_FAULT`
- A policy is found non-conformant at initialisation
- An evidence record fails hash verification
- The ordering invariant (SRS-007-SHALL-018) is violated
- A weight hash mismatch is detected (SRS-007-SHALL-013)

10.2 Integrity Fault Recording

SRS-007-SHALL-036

All Governance Integrity Faults SHALL be committed as `AX:PROOF:v1` records with:

- `result = INTEGRITY_FAULT`
- `violation` field identifying the fault type
- `evidence_refs` citing the inconsistent evidence

10.3 Integrity Fault Response

SRS-007-SHALL-037

On Governance Integrity Fault:

- The system SHALL transition L5 agent to `STOPPED`
- No further compliance reports SHALL be generated
- The fault SHALL be included in the next anchor publication
- Recovery SHALL require explicit operator action

10.4 No Silent Governance Failure

SRS-007-SHALL-038

Amended in v0.2 per F009.

No governance failure SHALL be silent. Every detected inconsistency SHALL produce a committed `AX:PROOF:v1` record.

10.4.1 Ledger Failure Mode

SRS-007-SHALL-053

Amended in v0.3 per F018.

If the L6 ledger is in FAILED state and cannot accept commits:

Option A — Deterministic Fallback Log:

- Integrity fault SHALL be recorded to a bounded, canonical, append-only fallback log
- Fallback log format SHALL be identical to `AX:PROOF:v1`

- Fallback log SHALL be replayable and independently verifiable
- Maximum fallback log size SHALL be declared at initialisation

Option B — System Halt:

- System SHALL halt with no further state mutation
- `local_governance_fault_flag` SHALL be set
- System SHALL remain in STOPPED state until explicit reset

The deployment SHALL declare which option is used in the configuration manifest.

Forbidden:

```
❌ Non-persistent state
❌ Unverifiable fallback
❌ Silent failure
```

Verification: fault injection test, inspection

10.4.2 Fallback Log Overflow Behaviour

SRS-007-SHALL-061

Added in v0.3 per F018.

If Option A (Deterministic Fallback Log) is selected, the overflow behaviour MUST be explicitly defined:

Option A.1 — Truncation with Marker:

- On overflow, stop appending new records
- Write a single `OVERFLOW_MARKER` record as final entry
- Set `local_governance_overflow_flag`
- Transition to STOPPED state

Option A.2 — System Halt on Overflow:

- On overflow, halt immediately
- No partial writes permitted
- Set `local_governance_overflow_flag`

- Remain in STOPPED state until explicit reset

Overflow Marker Record:

```
{
  "claim": "FALLBACK_LOG_OVERFLOW",
  "evidence_ordering": "LEX",
  "evidence_refs": [],
  "proof_type": "INTEGRITY_FAULT",
  "result": "INTEGRITY_FAULT",
  "rule_id": "SRS-007-SHALL-061",
  "schema_version": "AX:PROOF:v1",
  "violation": "FALLBACK_OVERFLOW"
}
```

Forbidden:

```
❌ Undefined overflow behaviour
❌ Silent truncation without marker
❌ Wrap-around / circular buffer (loses evidence)
```

Verification: overflow injection test, state machine test

---

11. TRACEABILITY

11.1 Evidence Citation Requirement

SRS-007-SHALL-039

Every governance claim SHALL cite the specific committed evidence records that justify it.

No governance assertion is valid without evidence citations.

11.2 Claim-to-Evidence Mapping

SRS-007-SHALL-040

The governance layer SHALL maintain a deterministic mapping:

```
Claim → Evidence Records → Verification Method → Result
```

This mapping SHALL be:

- Independently recomputable
- Committed as AX:PROOF:v1
- Included in compliance reports

11.3 SRS Traceability

SRS-007-SHALL-041

Every public governance function SHALL include SRS anchors per DVEC-001 v1.3 §9.2.

---

12. BOUNDEDNESS AND MEMORY

12.1 Bounded Execution

SRS-007-SHALL-042

Amended in v0.3 per F016.

All governance operations SHALL execute in bounded time.

For N evidence records:

Operation	Complexity	Notes
Cross-layer verification	O(N)	Linear scan of evidence chain
Merkle tree construction	O(N)	Bottom-up construction
Report generation	O(N)	Linear evidence aggregation
Anchor computation	O(1)	Fixed-size hash computation
Full chain replay	O(N × M)	N records, M policies per record

Merkle Tree Complexity Clarification:

Binary Merkle tree construction over N leaves:

- Tree height: $\lceil \log_2(N) \rceil$
- Total nodes: $2N - 1$
- Hash operations: $N - 1$ (internal nodes only, leaves are raw)
- Time complexity: $O(N)$
- Space complexity: $O(N)$ for tree storage, $O(\log N)$ for verification path

Replay Verification Complexity:

Full chain replay verification is $O(N \times M)$ where:

- N = number of evidence records
- M = average number of policy evaluations per record

For systems with bounded M (e.g., $M \leq 10$), this reduces to $O(N)$.

12.2 Bounded Allocation Model

SRS-007-SHALL-043

Amended in v0.2 per F004.

Governance operations SHALL use a **bounded allocation model**:

Option A — Fixed Maximum Sizes:

Structure	Maximum Size	Declaration
`evidence_refs` array	1024 entries	Compile-time constant
Mathematical Trace	4096 bytes	Compile-time constant
Compliance report	1 MiB	Configuration manifest
Fallback log	64 KiB	Configuration manifest

Option B — Caller-Provided Buffers:

All governance functions accepting variable-length output SHALL:

- Accept caller-provided buffer with explicit capacity
- Return error if capacity exceeded

- Never allocate internally

Forbidden:

```
❌ malloc / realloc / free in runtime paths
❌ Unbounded growth
❌ Implicit allocation
```

Verification: static analysis, buffer overflow test

---

13. PHASE 7 CLOSURE CRITERIA

Phase 7 is complete when:

13.1 Proof Model

- [ ] AX:PROOF:v1 records committed for all verification steps
- [ ] Cryptographic binding fields implemented (proof_hash, commitment, prev_chain_head)
- [ ] Policy soundness verified for all active policies
- [ ] Evidence closure confirmed

13.2 Cross-Layer Verification

- [ ] All 8 verification steps implemented
- [ ] Proof-before-execution invariant enforced
- [ ] Each step produces conformant AX:PROOF:v1
- [ ] Integrity fault detection operational

13.3 Compliance Reports

- [ ] Track A evidence package generated
- [ ] Track B evidence package generated
- [ ] Evidence closure proof included
- [ ] Reports independently verifiable
- [ ] Golden reference included in Track A

13.4 External Anchoring

- [ ] GPG key published
- [ ] Transparency log operational
- [ ] Anchor time bound to Time Oracle
- [ ] Anchor computation verified
- [ ] Anchor interval declared and committed

13.5 Mathematical Traces

- [ ] Full trace produced per inference event
- [ ] Trace fields complete, canonical, and properly encoded
- [ ] Trace hash included in AX:PROOF:v1
- [ ] Traces committed as AX:PROOF:v1

13.6 Memory Model

- [ ] Bounded allocation model declared
- [ ] Maximum sizes documented
- [ ] No runtime malloc in governance paths

---

14. REQUIREMENT SUMMARY

ID	Requirement	Section
SRS-007-SHALL-001	Determinism definition	3.2
SRS-007-SHALL-002	Evidence closure	3.3
SRS-007-SHALL-003	Proof commitment	4.1
SRS-007-SHALL-004	AX:PROOF:v1 required fields	4.2
SRS-007-SHALL-005	Proof type closed set	4.3
SRS-007-SHALL-006	Evidence reference requirement	4.4
SRS-007-SHALL-007	Canonical format	4.5
SRS-007-SHALL-008	Policy soundness	5.1
SRS-007-SHALL-009	Policy violation	5.2
SRS-007-SHALL-010	Policies as programs	5.3
SRS-007-SHALL-011	Verification chain	6.1
SRS-007-SHALL-012	L1 substrate certification	6.2
SRS-007-SHALL-013	L2 weight binding	6.3
SRS-007-SHALL-014	L3 observation integrity	6.4
SRS-007-SHALL-015	L4 policy soundness	6.5
SRS-007-SHALL-016	L3→L4 obs-policy binding	6.6
SRS-007-SHALL-017	L4→L5 breach enforcement	6.7
SRS-007-SHALL-018	L5→L6 pre-commit ordering	6.8
SRS-007-SHALL-019	Full chain replay	6.9
SRS-007-SHALL-020	Mathematical trace requirement	7.1
SRS-007-SHALL-021	Trace required fields	7.2
SRS-007-SHALL-022	Trace canonicality	7.3
SRS-007-SHALL-023	Track A evidence package	8.1

ID	Requirement	Section
SRS-007-SHALL-024	Track B evidence package	8.2
SRS-007-SHALL-025	Report trigger conditions	8.3
SRS-007-SHALL-026	Report canonical format	8.4
SRS-007-SHALL-027	Report independence	8.5
SRS-007-SHALL-028	Golden reference inclusion	8.6
SRS-007-SHALL-029	Anchor requirement	9.1
SRS-007-SHALL-030	Anchor computation	9.2
SRS-007-SHALL-031	Anchor signing	9.3
SRS-007-SHALL-032	Anchor commitment	9.4
SRS-007-SHALL-033	Anchor interval declaration	9.5
SRS-007-SHALL-034	Anchor verification	9.6
SRS-007-SHALL-035	Integrity fault definition	10.1
SRS-007-SHALL-036	Integrity fault recording	10.2
SRS-007-SHALL-037	Integrity fault response	10.3
SRS-007-SHALL-038	No silent governance failure	10.4
SRS-007-SHALL-039	Evidence citation requirement	11.1
SRS-007-SHALL-040	Claim-to-evidence mapping	11.2
SRS-007-SHALL-041	SRS traceability	11.3
SRS-007-SHALL-042	Bounded execution	12.1
SRS-007-SHALL-043	Bounded allocation model	12.2
SRS-007-SHALL-044	Configuration canonicality	3.2.1
SRS-007-SHALL-045	Cryptographic binding fields	4.2.1
SRS-007-SHALL-046	Evidence reference encoding	4.4.1
SRS-007-SHALL-047	Proof-before-execution invariant	6.1.1

ID	Requirement	Section
SRS-007-SHALL-048	Weight hash specification	6.3.1
SRS-007-SHALL-049	Trace array ordering	7.2.1
SRS-007-SHALL-050	Trace hash computation	7.2.2
SRS-007-SHALL-051	Evidence closure proof	8.5.1
SRS-007-SHALL-052	Anchor time as oracle	9.2.1
SRS-007-SHALL-053	Ledger failure mode	10.4.1

Total: 53 SHALL requirements

---

15. LAYER INTEGRATION SUMMARY

Layer	Role	SRS
L7	Accountability — what can be proven	SRS-007
L6	Truth — what happened	SRS-001
L5	Behaviour — what to do	SRS-002
L4	Admissibility — what is allowed	SRS-003
L3	Containment — what was observed	SRS-004
L2	Identity — what the model computed	SRS-005 (via certifiable-inference)
L1	Mathematics — what is provable	SRS-005

Full Stack Invariant:

```
L1 (arithmetic law)
  → L2 (deterministic inference)
    → L3 (contained observation)
      → L4 (policy gate)
        → L5 (bounded behaviour)
          → L6 (cryptographic truth)
            → L7 (governance proof)
```

No layer can produce an undetected lie without breaking the chain.

---

16. COMPLETE SRS STACK

Document	Layer	SHALL Count	Status
SRS-001	L6 Substrate	29	Audit-Frozen
SRS-002	L5 Agent	29	Audit-Frozen
SRS-003	L4 Policy	22	Audit-Frozen
SRS-004	L3 Oracle	48	Audit-Frozen
SRS-005	L1 Arithmetic	70	Audit-Frozen
SRS-007	L7 Governance	53	Audit-Frozen
Total		**251**	

---

17. FINAL STATEMENT

The governance layer SHALL:

> Produce cryptographically evidenced proofs that the system behaved correctly — not assertions that it did.

System Property:

> No layer can behave differently without producing different evidence. No evidence can be falsified without breaking the cryptographic chain. No chain can be broken without governance detecting and recording the fault.

The Axioma Governance Theorem:

> A system governed by Axioma L7 cannot certify incorrect behaviour — it can only certify correct behaviour or declare a governance integrity fault.

---

18. REVISION HISTORY

Version	Date	Author	Changes
0.1	2026-03-27	William Murray	Initial draft — 43 SHALL requirements
0.2	2026-03-27	William Murray	Audit closure — 10 new SHALLs (044–053). Added cryptographic binding fields, evidence encoding spec, proof-before-execution invariant, weight hash spec, trace encoding/ordering, evidence closure proof, anchor time oracle binding, ledger failure mode, configuration canonicality. Total: 53 SHALL requirements.
0.3	2026-03-27	William Murray	Final audit closure — 8 new SHALLs (054–061). Fixed proof_hash canonical treatment (field omission), payload encoding byte semantics, Merkle tree algorithm specification, evidence ordering structured field, GPG signature determinism boundary, atomicity model, complexity bounds, fallback overflow behaviour, proof type versioning. Total: 61 SHALL requirements.

19. DOCUMENT APPROVAL

Role	Name	Date	Signature
Author	William Murray	2026-03-27	
Reviewer			
Approver			

Appendix A — Audit Record (v0.1 → v0.2)

Finding ID	Severity	Resolution
F001	🔴 BLOCKING	SRS-007-SHALL-045: Added proof_hash, commitment, prev_chain_head
F002	🔴 BLOCKING	SRS-007-SHALL-046: Evidence refs SHA-256, hex lowercase, ordered
F003	🔴 BLOCKING	SRS-007-SHALL-052: Anchor time bound to Time Oracle AX:OBS:v1
F004	🔴 BLOCKING	SRS-007-SHALL-043: Replaced with bounded allocation model
F005	🟠 HIGH	SRS-007-SHALL-047: Proof-before-execution invariant
F006	🟠 HIGH	SRS-007-SHALL-049, 050: Trace encoding, ordering, hash
F007	🟠 HIGH	SRS-007-SHALL-051: Evidence closure proof (Merkle root)
F008	🟡 MEDIUM	SRS-007-SHALL-048: Weight hash canonical specification
F009	🟡 MEDIUM	SRS-007-SHALL-053: Deterministic ledger failure mode
F010	🔵 LOW	SRS-007-SHALL-044: Configuration canonicality

Audit Source: External review

Audit Date: 27 March 2026

Audit Verdict: All findings resolved. Document elevated to **Audit-Frozen**.

Appendix B — Cross-Reference to Audit Findings

SHALL	Audit Finding	Description
SHALL-043	F004	Bounded allocation replaces zero-allocation
SHALL-044	F010	Policy/config canonicalisation
SHALL-045	F001	Cryptographic binding fields
SHALL-046	F002	Evidence reference encoding
SHALL-047	F005	Proof-before-execution
SHALL-048	F008	Weight hash specification
SHALL-049	F006	Trace array ordering
SHALL-050	F006	Trace hash computation
SHALL-051	F007	Evidence closure proof
SHALL-052	F003	Anchor time as oracle
SHALL-053	F009	Ledger failure mode

Appendix C — Conformance Status

Category	v0.1	v0.2	v0.3
Determinism	❌ FAIL	⚠️ PARTIAL	✅ PASS
Totality	❌ FAIL	⚠️ PARTIAL	✅ PASS
Cross-platform safety	❌ FAIL	⚠️ PARTIAL	✅ PASS
SRS alignment	❌ FAIL	✅ PASS	✅ PASS
Evidence closure	❌ FAIL	⚠️ PARTIAL	✅ PASS
Cryptographic binding	❌ FAIL	⚠️ PARTIAL	✅ PASS

Appendix D — Audit Record (v0.2 → v0.3)

Finding ID	Severity	Resolution
F011	🔴 BLOCKING	SRS-007-SHALL-045 amended: proof_hash field omitted (not empty/null)
F012	🔴 BLOCKING	SRS-007-SHALL-054: Payload encoding byte semantics defined
F013	🔴 BLOCKING	SRS-007-SHALL-059: Merkle tree algorithm fully specified
F014	🟠 HIGH	SRS-007-SHALL-057: Structured evidence_ordering field
F015	🟠 HIGH	SRS-007-SHALL-060: GPG signature determinism boundary
F016	🟠 HIGH	SRS-007-SHALL-042 amended: Complexity bounds corrected
F017	🟡 MEDIUM	SRS-007-SHALL-058: Atomicity model for proof-before-execution
F018	🟡 MEDIUM	SRS-007-SHALL-061: Fallback log overflow behaviour
F019	🔵 LOW	SRS-007-SHALL-056: Proof type versioning mechanism

Audit Source: GPT-4o external review

Audit Date: 27 March 2026

Audit Verdict: All findings resolved. Document elevated to **Audit-Frozen FINAL**.

Appendix E — Cross-Reference to v0.3 Audit Findings

SHALL	Audit Finding	Description
SHALL-045	F011	proof_hash canonical treatment (omit field)
SHALL-054	F012	Commitment payload encoding (UTF-8 bytes)
SHALL-055	—	Chain head reference (completeness)
SHALL-056	F019	Proof type versioning mechanism
SHALL-057	F014	Evidence ordering structured field
SHALL-058	F017	Atomicity model
SHALL-059	F013	Merkle tree construction algorithm
SHALL-060	F015	GPG signature determinism boundary
SHALL-061	F018	Fallback log overflow behaviour

SRS-007 v0.3 — Audit-Frozen FINAL William Murray · SpeyTech · March 2026 Patent GB2521625.0