Governance Synchronization Specification v1

Author: Tanishq Dasari, AnimusLab **Version:** 1.0.0 **Status:** Draft **Date:** July 2026 **Contact:** tan@animuslab.dev

---

1. Purpose

This specification defines the contract between Canon and Anchor.

Canon is a deterministic governance synchronization engine. It continuously monitors authoritative governance sources, detects changes, generates evidence-backed update proposals, and coordinates human review.

Anchor is the governance enforcement engine. It owns the governance rule repository, the constitution, and the governance lock file. Anchor is the sole authority responsible for applying approved proposals and enforcing the resulting rule set.

Canon never edits Anchor's governance files directly. Anchor never fetches from external sources. They communicate exclusively through a shared filesystem boundary: the `.canon/proposals/` directory.

This specification defines:

- The structure and content of governance proposals
- The proposal lifecycle from detection to merge
- The filesystem contract between Canon and Anchor
- Verification requirements at each stage
- Failure modes and their required handling

---

2. Definitions

Authoritative Source — An external governance framework that Anchor's rules are mapped against. Current authoritative sources: FINOS AI Governance Framework, OWASP LLM Top 10, NIST AI Risk Management Framework.

Source Rule — A single governance rule as published by an authoritative source, identified by a source-specific ID (e.g., NIST GOVERN.1.1, OWASP LLM01).

Anchor Rule — A rule defined within Anchor's governance repository, in a `.anchor` file under `governance/frameworks/`. One Anchor rule may map to one or more source rules.

Semantic Change — A change to a source rule's title, description, severity, category, or regulatory references. Formatting changes that do not alter meaning do not constitute a semantic change.

Proposal — A structured package produced by Canon when a semantic change is detected. A proposal contains the proposed updated Anchor rule file, a human-readable diff, an evidence package, and metadata. A proposal is not applied until it has been approved by a human reviewer and confirmed by the developer running `anchor update`.

Proposal ID — A unique identifier for a proposal. Format: `CANON-YYYY-NNNNN` where YYYY is the four-digit year and NNNNN is a zero-padded sequence number reset annually. Example: `CANON-2026-00017`.

Evidence Package — A structured record of what changed in the authoritative source, produced by Canon's diff engine. Cryptographically hashed. Cannot be altered after creation.

Approval Record — A tamper-evident record of a human reviewer's decision on a proposal. Contains the reviewer's identity, decision, justification, timestamp, and a hash of the evidence package it references.

governance.lock — Anchor's lock file recording the exact version of each authoritative source that Anchor's current governance rules were last synchronized against. Analogous to a dependency lock file.

---

3. Filesystem Contract

The integration boundary between Canon and Anchor is the `.canon/` directory, located at the root of the Anchor repository.

3.1 Directory Layout

```
.canon/
    proposals/
        CANON-2026-00001/
            metadata.json       [REQUIRED]
            proposal.anchor     [REQUIRED]
            diff.md             [REQUIRED]
            evidence.json       [REQUIRED]
            approval.json       [OPTIONAL - written after human approval]
    cache/
        <source-id>.json        [Canon internal - source state snapshots]
    history/
```

```
    CANON-2026-00001/       [Archived after anchor update merges]
  ledger.jsonl              [Canon append-only ledger]
```

3.2 Ownership

Path	Owner	May Edit
`.canon/proposals/`	Canon writes, Anchor reads	Canon only
`.canon/proposals/*/approval.json`	Canon writes via `canon review`	Canon only
`.canon/cache/`	Canon	Canon only
`.canon/history/`	Anchor writes via `anchor update`	Anchor only
`.canon/ledger.jsonl`	Canon	Canon only
`governance/`	Anchor	Anchor only
`constitution.anchor`	Anchor	Anchor only
`governance.lock`	Anchor	Anchor only

Canon never writes to `governance/`, `constitution.anchor`, or `governance.lock`. Anchor never writes to `.canon/proposals/`, `.canon/cache/`, or `.canon/ledger.jsonl`.

---

4. Proposal Files

4.1 metadata.json

Records the origin and scope of a proposal. Written by Canon at proposal creation. Never modified after creation.

Schema:

```
json
```

```
{
    "proposal_id": "CANON-2026-00001",
    "spec_version": 1,
    "created_at": "2026-07-04T12:00:00Z",
    "source_id": "nist-ai-rmf",
    "source_name": "NIST AI Risk Management Framework",
    "source_version_baseline": "1.0",
    "source_version_current": "1.1",
    "source_commit": "abc123def456",
    "affected_anchor_file": "governance/frameworks/NIST_AI_RMF.anchor",
    "affected_rules": ["NIST-GOVERN.1.1", "NIST-MAP.2.2"],
    "change_summary": "GOVERN.1.1 description updated; MAP.2.2 severity raised
    "metadata_hash": "<sha256 of this file excluding this field>"
}
```

Required fields: all fields above. **metadata_hash** is computed over the canonical JSON of this object with `metadata_hash` set to empty string, then SHA-256 hashed.

4.2 proposal.anchor

A complete copy of the affected Anchor governance file (`affected_anchor_file`), with the proposed changes applied. This is a full file replacement, not a patch — the reviewer sees the complete proposed state, not just a diff.

The file format is identical to existing `.anchor` files. Anchor's parser must be able to load it without modification.

Canon generates the proposed content deterministically from the source change. The human reviewer may modify this file before approval if the automatically generated proposal requires adjustment.

4.3 diff.md

A human-readable Markdown document showing exactly what changed. Intended for human review, not machine parsing.

Required sections:

```
markdown
```

```
# Proposal CANON-2026-00001

**Source:** NIST AI Risk Management Framework
**Date:** 2026-07-04
**Affected file:** governance/frameworks/NIST_AI_RMF.anchor

## Source Change

### NIST GOVERN.1.1
**Previous text:** ...
**Current text:** ...
**Change type:** description modified

## Proposed Anchor Rule Changes

### NIST-GOVERN.1.1
- severity: WARNING → CRITICAL
- description: [updated text]

## Regulatory Impact
Frameworks affected: NIST-AI-RMF-1.0
```

4.4 evidence.json

The evidence package produced by Canon's diff engine for this proposal. Contains the structured diff of the source state change that triggered this proposal.

Schema:

```
json
```

```
{
    "evidence_id": "<uuid>",
    "proposal_id": "CANON-2026-00001",
    "produced_at": "2026-07-04T12:00:00Z",
    "source_id": "nist-ai-rmf",
    "baseline_hash": "<sha256 of baseline source state>",
    "current_hash": "<sha256 of current source state>",
    "diffs": [
        {
            "rule_id": "NIST-GOVERN.1.1",
            "change_type": "modified",
            "old_value": {"severity": "warning", "description": "..."},
            "new_value": {"severity": "critical", "description": "..."},
            "severity_delta": "WARNING -> CRITICAL",
            "regulatory_mappings_affected": ["NIST-AI-RMF-1.0"]
        }
    ],
    "evidence_hash": "<sha256 of this object excluding this field>"
}
```

`evidence_hash` is computed the same way as `metadata_hash`.

4.5 approval.json

Written by Canon after a human reviewer approves the proposal via `canon review`. Not present until approval is recorded.

Schema:

json

```
{
    "proposal_id": "CANON-2026-00001",
    "approved_by": "tanishq",
    "decision": "approved",
    "justification": "NIST change correctly reflected in proposed rule.",
    "decided_at": "2026-07-04T14:30:00Z",
    "evidence_hash": "<must match evidence.json evidence_hash>",
    "proposal_anchor_hash": "<sha256 of proposal.anchor at time of approval>",
    "record_hash": "<sha256 of this object excluding this field>"
}
```

decision must be one of: `approved`, `rejected`, `deferred`.

Only proposals with `decision: approved` are visible to `anchor update`.

`evidence_hash` in `approval.json` must match `evidence_hash` in `evidence.json`. If they differ, the proposal is considered tampered and must be rejected.

`proposal_anchor_hash` records the exact state of `proposal.anchor` that was approved. If `proposal.anchor` is modified after approval, `anchor update` detects the mismatch and refuses to merge.

---

5. Proposal Lifecycle

```
1. DETECTED
   Canon detects a semantic change in an authoritative source.
   Evidence package created and hashed.

2. PROPOSED
   Canon generates proposal.anchor, diff.md, metadata.json, evidence.json.
   Proposal directory created under .canon/proposals/CANON-YYYY-NNNNN/.
   Ledger entry written.

3. UNDER REVIEW
   Human reviewer examines proposal via `canon review`.
   Reviewer may modify proposal.anchor before approving.

4. APPROVED / REJECTED / DEFERRED
   approval.json written by Canon.
   Ledger entry written.
   If rejected or deferred: proposal remains in .canon/proposals/ but
   anchor update ignores it.

5. MERGED (approved proposals only)
   Developer runs `anchor update`.
   Anchor shows diff between current governance file and proposal.anchor.
   Developer confirms.
   Anchor copies proposal.anchor to governance/frameworks/<file>.
   Anchor updates governance.lock.
   Proposal directory moved to .canon/history/CANON-YYYY-NNNNN/.
   Ledger entry written by Canon (notified by Anchor).

6. ARCHIVED
   Proposal lives permanently in .canon/history/.
   Immutable after archival.
```

6. governance.lock Extension

Anchor's existing `governance.lock` is extended to record the synchronization state of each authoritative source.

New section added to governance.lock:

```json
{
    "canon_sync": {
        "spec_version": 1,
        "last_synced_at": "2026-07-04T12:00:00Z",
        "sources": {
            "nist-ai-rmf": {
                "version": "1.0",
                "commit": "abc123",
                "last_checked": "2026-07-04T12:00:00Z",
                "last_updated": "2026-06-01T09:00:00Z",
                "state_hash": "<sha256 of cached source state>"
            },
            "owasp-llm-top10": {
                "version": "2.0",
                "commit": "def456",
                "last_checked": "2026-07-04T12:00:00Z",
                "last_updated": "2026-05-15T11:00:00Z",
                "state_hash": "<sha256 of cached source state>"
            },
            "finos-aigf": {
                "version": "latest",
                "commit": "ghi789",
                "last_checked": "2026-07-04T12:00:00Z",
                "last_updated": "2026-04-20T08:00:00Z",
                "state_hash": "<sha256 of cached source state>"
            }
        }
    }
}
```

`anchor update` writes to the `canon_sync` section of `governance.lock` after a successful merge.

7. anchor update Command

`anchor update` is a new Anchor CLI command. It reads approved proposals from `.canon/proposals/` and applies them to the governance repository.

7.1 Behaviour

```
1. Scan .canon/proposals/ for directories containing approval.json
   with decision: approved.

2. For each approved proposal:
   a. Verify approval.json record_hash (tamper check).
   b. Verify proposal.anchor hash matches approval.json proposal_anchor_hash.
   c. Verify evidence.json evidence_hash matches approval.json evidence_hash.
   d. If any verification fails: abort with error, do not merge.

3. For each verified proposal:
   a. Show proposal metadata (source, affected file, change summary).
   b. Show diff between current governance file and proposal.anchor.
   c. Prompt: Apply proposal CANON-YYYY-NNNNN? [Y/n]
   d. If confirmed:
      - Copy proposal.anchor to affected_anchor_file.
      - Update canon_sync section of governance.lock.
      - Move proposal directory to .canon/history/.
      - Print confirmation.
   e. If declined: skip, leave proposal in .canon/proposals/.

4. After all proposals processed: print summary.
```

7.2 Output format

```
anchor update

Scanning for approved proposals...

──────────────────────────────────────────

Proposal: CANON-2026-00001
Source:   NIST AI Risk Management Framework
File:     governance/frameworks/NIST_AI_RMF.anchor
Changes:  2 rules modified (1 severity increase)
Approved: tanishq @ 2026-07-04 14:30 UTC
Evidence: VERIFIED ✓
```

```
Signature: VALID ✓
──────────────────────────────────────────

Diff:
  NIST-GOVERN.1.1
-   severity: warning
+   severity: critical
-   description: Policies and processes for AI risk management are
documented.
+   description: Policies and processes for AI risk management are documented
+                and independently verified on a quarterly basis.

  NIST-MAP.2.2
-   description: AI system risks are identified through formal assessment.
+   description: AI system risks are identified through formal assessment
+                with documented methodology and board-level sign-off.

Apply proposal CANON-2026-00001? [Y/n]:
```

---

8. Verification Requirements

8.1 Canon side (before proposal is written)

- `evidence_hash` in evidence.json must be computed and verified before proposal is created.
- `metadata_hash` in metadata.json must be computed before proposal is created.
- Both are written once and never modified.

8.2 Canon side (before approval.json is written)

- `proposal_anchor_hash` is computed at the moment of approval, over the current state of `proposal.anchor`. If the reviewer modified `proposal.anchor`, this hash reflects the modified version.
- `evidence_hash` in approval.json must match `evidence_hash` in evidence.json.
- `record_hash` is computed over the full approval.json object.

8.3 Anchor side (before merge)

- Recompute `record_hash` over approval.json (excluding the field itself). Must match stored value.

- Recompute hash of current `proposal.anchor`. Must match `proposal_anchor_hash` in approval.json.
- Recompute `evidence_hash` over evidence.json. Must match stored value and approval.json value.
- If any check fails: abort. Print which check failed. Do not merge.

---

9. Failure Modes

Failure	Canon behaviour	Anchor behaviour
Source unreachable	Log warning, skip source, retry next sync	N/A
No semantic change detected	No proposal created	N/A
Anchor rule mapping not found	Proposal flagged as UNMAPPED, human must specify target file	`anchor update` skips UNMAPPED proposals
proposal.anchor modified after approval	N/A	Abort merge, print hash mismatch
approval.json tampered	N/A	Abort merge, print integrity violation
evidence.json hash mismatch	N/A	Abort merge, print integrity violation
Proposal already merged	N/A	Skip silently (already in history/)
Conflicting proposals for same file	Second proposal flagged as CONFLICT	`anchor update` processes in creation order, flags conflicts

---

10. Staleness Detection

Canon tracks, in `governance.lock`, the last time each source was checked and the last time it was updated. If a source has not been checked within a configurable threshold (default: 7 days), Canon flags the governance repository as potentially stale when `anchor check` or `anchor status` is run.

Output:

```
⚠ Governance staleness warning:
  nist-ai-rmf last checked 9 days ago.
  Run 'canon sync' to check for updates.
```

This gives institutions the ability to demonstrate, on demand, that their governance knowledge is current — not just that it was correct at some past point in time.

---

11. Compatibility

- This specification is versioned. `spec_version: 1` is recorded in metadata.json.
- Future versions of this specification must maintain backward compatibility with v1 proposals.
- `anchor update` must be able to process v1 proposals regardless of the current spec version.
- Canon must write the spec version it used into every proposal it creates.

---

12. Out of Scope for v1

The following are explicitly deferred to future versions:

- Automatic proposal generation without human approval
- Multi-reviewer approval workflows
- Proposals spanning multiple governance files simultaneously
- Canon integration with Governance Hub
- Remote proposal submission (proposals created on one machine, approved on another)
- AnchorJIT constraint compilation from approved proposals

This specification is the contract. Canon and Anchor are independently implemented against it. Neither project's internal implementation is visible to the other. The filesystem boundary defined in Section 3 is the only coupling.