We Made an AI Write a Paper It Can't Lie About

Dexter Hadley

Abstract

What if an AI could not bullshit its way through scientific prose—not because it chose not to, but because the system rendered unsupported claims inadmissible? We call this failure mode **AI slop**.

CANONIC is the governance framework we built to make AI slop structurally inadmissible. Claims must trace to a ledger; every term used in rules must be defined; the AI cannot promote its own ideas to law. Across 129 recorded episodes at `stack-freeze-2026-01-12`, we observed recurring failures (undefined terms, evidence gaps, governance leakage) and the corrections that followed.

The result is this paper. It does not describe an experiment. It is the experiment. Key claims cite evidence references (commits, tags, episodes) so you can verify them yourself.

We asked whether a governed human-AI system could produce a self-evidencing scientific paper. You are reading the answer.

We begin with the AI slop problem, then the governance primitives that make it inadmissible.

The Problem: AI Slop Is Eating Scientific Writing

LLMs are everywhere in research. They are powerful—until they are not.

The failure mode has a name: **AI slop**. It reads as authoritative and means nothing: undefined terms, unverifiable claims, confident fabrications. The AI equivalent of padding an essay.

Current defenses fail:

- **Detection tools** are unreliable and easily fooled
- **Disclosure policies** are unverifiable ("I used AI responsibly" proves nothing)
- **Human review** catches AI slop but does not prevent it

We wanted something different: a system where AI-slop-like failure modes are structurally inadmissible to the ledger—invalid by construction, not filtered out.

That requirement motivates the constitutional governance idea next.

The Idea: Constitutional AI Governance

The insight is simple: treat AI collaboration like a legal system.

The root primitive is the Triad—`CANON.md`, `VOCAB.md`, and `README.md`.

With the Triad in place, a constitution defines validity, a ledger records what happened, and courts (validators) check compliance. Crucially, the AI can observe and propose, but only humans can change the rules.

We call the framework CANONIC. In the frozen stack, it rests on seven governance primitives.

1. Triad

Every scope needs three files: `CANON.md` (rules), `VOCAB.md` (definitions), `README.md` (description). Missing any makes the scope invalid. When present, `SPEC.md` is part of the closure: SPEC, CANON, and README may only use terms defined in VOCAB.

Figure 1 shows the triad and closure rules across SPEC, CANON, VOCAB, and README.

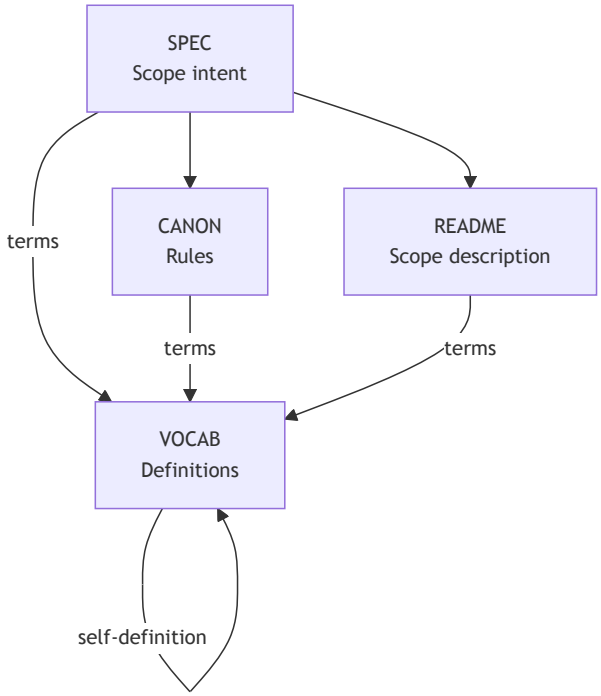

Figure 1: Triad closure across SPEC, CANON, VOCAB, and README.

This closure makes terms admissible before any downstream execution.

2. Inheritance

Rules flow from a root constitution. You can add constraints downstream, but you cannot override upstream rules.

3. Introspection

Every term used in the rules must be defined. If CANON says "episode," VOCAB must define "episode." Undefined jargon = invalid scope.

4. Immutability (governed)

The ledger is treated as immutable. Corrections happen via new commits; history rewriting is disallowed by governance.

5. Model identity disclosure (best practice)

Sessions SHOULD record the actual model identity. This was not consistently captured early; the gap is documented as a limitation (see Limitations).

6. Ledger-first evidence

Claims without evidence references are inadmissible; "the system achieved compliance" means nothing unless you can point to the commit where it happened.

7. Insight-law separation

The AI can discover patterns and propose ideas. But those insights have zero legal force until a human explicitly adds them to CANON.

Together these primitives form the governance loop shown next.

Governance loop

Figure 2 sketches the governance loop across authority, execution, and evidence.

This loop explains why authority stays human while execution can scale.

With the loop defined, we can describe the concrete stack we built.

What We Actually Did

With the primitives and governance loop established, we built the system across the public CANONIC stack. Table 1 lists the governed repositories and their purposes.

Table 1. CANONIC stack scopes and purposes.

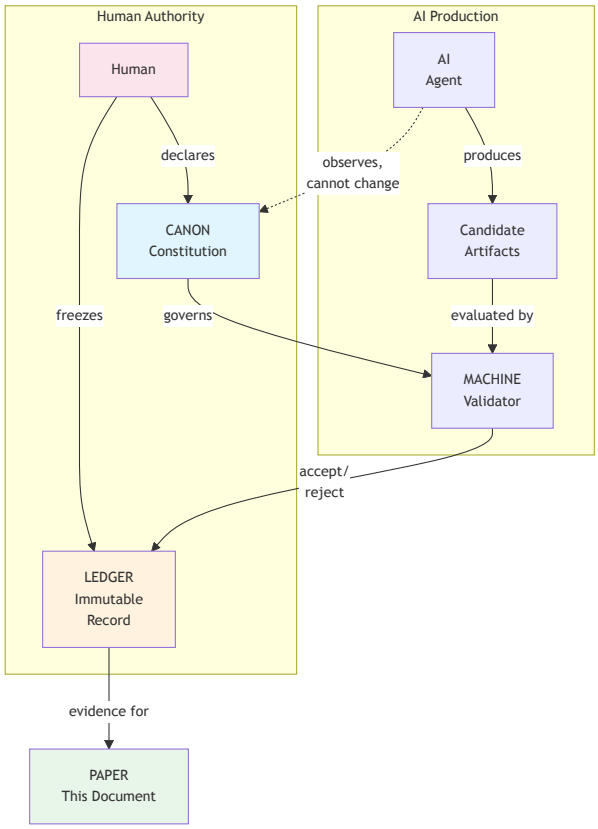

Figure 2: Governance loop from human authority to ledger evidence.

Repo	Purpose
canonic	Root constitution
machine	Execution semantics
os	Authority bounds
transcript	Pre-ledger transcript evidence (App/IDE, reconstruction)
ledger	Immutability rules
writing	Episode production
paper	This paper's governance
stack	Multi-system composition
validators	Enforcement outcomes (public)
patents	Disclosures and governance IP
publishing	Submission and dissemination artifacts

Enforcement outcomes are recorded in the ledger; validator implementations may be public or private.

Publishing is a post-freeze dissemination scope and is not part of the freeze evidence window.

That stack context sets up the evidence pipeline shown next.

Ledger-as-evidence pipeline

Figure 2a shows how rules become evidence and claims.

This pipeline is the basis for the evidence window and claim admissibility.

It also explains why a single scope must split into distinct governed repositories.

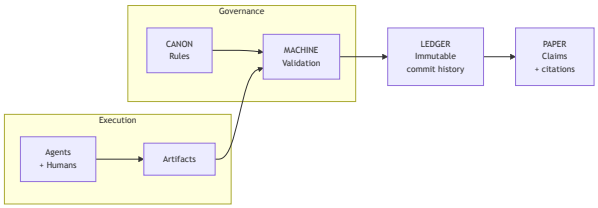

Figure 3: Ledger-as-evidence pipeline from governance to paper claims.

From single scope to multi-repo

The ledger is evolutionary: ideas branch, revisions accumulate, and a human fixation collapses that exploration into a single admissible branch. Fixation does not delete history; it selects the branch that becomes evidence.

Figure 3a sketches the branch-to-fixation collapse that turns exploration into admissible evidence.

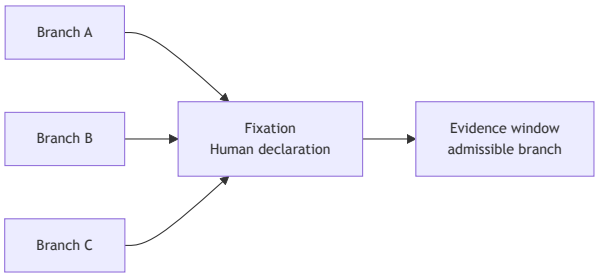

Figure 4: Branch-to-fixation collapse that selects the admissible branch.

With the evidence pipeline in place, we can show how the architecture expanded. As fixation hardened scope boundaries, the single-scope system split into a multi-repo stack. Figure 3 shows that evolution.

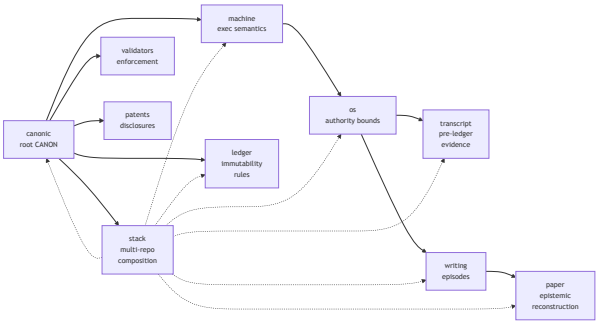

Figure 5: Multi-repo expansion from root CANON into governed scopes.

The expansion created distinct scopes without collapsing governance boundaries. With scope boundaries fixed, we can define the OS substrate that stabilizes the evidence window.

Minimal CANONIC OS (operating substrate)

We treat CANONIC OS as the operating substrate for the LLM: a frozen, minimal layer that constrains execution and does not mutate during a session. The OS is not a policy playground; it is the stable substrate on which downstream scopes rely.

Best practice: instantiate new WRITING and PAPER work against a frozen OS so the evidence window stays stable and governance does not drift under the experiment.

For this paper, the OS substrate is `os:4c2919d` at `stack-freeze-2026-01-12`.

That fixed substrate defines the evidence window next.

Evidence window

Version v0.1.0 (anchor tag `stack-freeze-2026-01-12`)

Pre-freeze evidence is anchored at tag `stack-freeze-2026-01-12` (tag timestamp `2026-01-12T18:34:47-05:00`, commit `writing:f8acf128`). All counts and timestamps in this manuscript are computed from that tag. Post-freeze revisions are marked as reconstruction.

Branching after fixation is treated as reconstruction unless a new fixation is declared.

At `stack-freeze-2026-01-12`, the writing repo contains 129 episode artifacts in `writing/episodes/`, documenting human-AI collaboration. When applicable, episodes record:

- Explicit model disclosure
- Commit-linked evidence
- Documented violations and corrections

The system evolved through recorded revisions. By the freeze tag, the root CANON in `canonic` contains three axioms, and each scope's vocabulary is defined in its own `VOCAB.md`. Episodes ep019, ep053, and ep060 document the compression path.

On January 12, 2026, we froze the ledger:

> "I declare that all SPEC evolution across the CANONIC stack is complete and stable… This declaration constitutes human fixation." - Dexter Hadley

Discovery annotations are sourced from the Discoveries stream index (`writing/streams/discoveries/README.md`).

Figure 4 stacks three rows on one timeline: ledger events, PAPER CANON drift, and PAPER VOCAB drift. Commit anchors are listed below.

Drift and extinction claims (ledger-anchored):

- DICTIONARY -> VOCAB rename (`paper/VOCAB.md@b996b02`).
- Expanded PAPER constraints (standard structure, layered insights, episode rawness, meta-circularity) in `paper/CANON.md@6e2a70b`, `paper/CANON.md@92f0db9`, and `paper/CANON.md@2c8eea9`.
- Extinction via minimal reset (removal of those constraints) in `paper/CANON.md@762a28e`.

- VOCAB concept-only pass (`paper/VOCAB.md@de5fed3`), closure pass (`paper/VOCAB.md@24dc57b`), removal (`paper/VOCAB.md@3b60c43`), restore (`paper/VOCAB.md@0ee1970`).
- Narrative fixations: section anchoring (`paper/CANON.md@a49d11c`) and continuity (`paper/CANON.md@99ebdba`).
- Formatting profile added post-freeze in this revision (reconstruction).

Figure 4 appears below with the drift timeline.

This evolution explains the CANONIC WRITING PAPER shape: early over-specification collapsed into minimal axioms, then narrative continuity and evidence anchoring were added back as governed fixes. The paper is not static; it is a ledger- constrained rewrite of its own governance history.

The first CANON artifact in the stack is `writing/CANON.md` at `2026-01-05T14:13:20-05:00` (`writing:bca9ec0`). The freeze declaration at `stack-freeze-2026-01-12` (`writing:f8acf128`, tag timestamp `2026-01-12T18:34:47-05:00`) occurred **7 days, 4:21:27** later.

Everything at or before the freeze is evidence. Everything after is reconstruction.

With the window fixed, we can add transcript signals to extend chronology and attribution.

Transcript-anchored provenance (App to IDE tooling)

We can extend the timeline beyond commit timestamps by linking ledger signals to preserved session transcripts. Early CANONIC foundation commits include Claude Code co-author lines (for example `canonic:49d6b65`), while later sessions are preserved in Codex CLI logs (see ep130 and ep133). Joining commit IDs with transcripts lets PAPER reconstruct conception-to-production pathways with finer granularity, while still anchoring claims in the ledger.

This is a reconstruction technique: transcripts add context, and the ledger anchors admissible facts. We treat transcript evidence as support for chronology and attribution, not as authority over the ledger.

We now open a TRANSCRIPT layer as a secondary ledger for App and IDE transcripts (ep152). A closed transcript validator can hash and attest records so only identifiers and outcomes are exposed publicly.

Transcripts are not published; PAPER cites only ledger IDs and episode references so confidentiality is preserved.

We also use transcript-side positive control markers (for example “LFG”, “Click”) to bootstrap discovery signal detection, then test whether those discovery moments drift into canonified constraints over time. This supports a hypothesis that concepts emerge in App-agent exchanges and later enter CANON via canonification (ep146, ep147). Unenforced positive controls are treated as signal violations and are audited explicitly (ep148).

Figure 5 shows the knowledge API from STREAMS to LEDGER: learned episodes synthesize stream evidence across App and IDE channels, then anchor claims back to ledger events. The inception point for a concept is the earliest stream occurrence that precedes its ledger recording.

These transcript signals motivate the best-practice patterns reported next.

Best practices (patterns and meta-patterns)

We report patterns rather than implementation details to preserve IP boundaries and confidentiality. The best practices are:

- Separation: keep primary evidence immutable while derived discovery artifacts remain explicitly secondary.
- Atomicity: one discovery event per derived record, enabling clean temporal mapping.
- Traceability: every derived discovery points back to ledger evidence and source context without exposing raw transcripts.
- Coverage: curate every interaction that supports a claim or disclosure, and index it in relevant streams to keep claims reconstructable.
- Provenance: capture agent channel (IDE vs App) as a first-class signal for injection analysis.
- Control signals: use positive-control markers to detect discovery moments and audit follow-through.
- Drift accounting: track concept migration from discovery into canon as a measurable signal, not an assumption.

These patterns set up the observations we report at freeze.

Results: Observations at Freeze

At freeze, the triad compliance report lists 12 triad scopes across 9 repositories (see `writing/episodes/ep136-stack-compliance-reports.md`). Figure 1 shows the triad closure model.

Each listed scope contains CANON, VOCAB, and README at the freeze tag.

We begin with the violation record, then summarize the post-freeze slice and attribution.

The Violation Record

At freeze, 33 episodes are explicitly labeled as violations by filename (`writing/episodes/*violation*`).

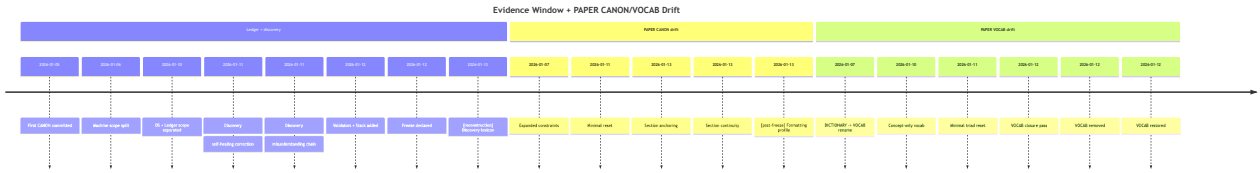

Figure 6: Evidence window with ledger events, PAPER CANON drift, and PAPER VOCAB drift.

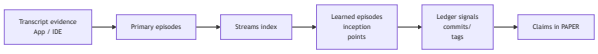

Figure 7: Knowledge API from transcripts to ledger-aligned claims.

Every violation was detected (mechanically or by review), documented (in an episode), and corrected (via new commit, never revision).

The violations are features, not bugs. They show the system catches problems and preserves the learning process.

Post-freeze, we also track understanding failures that are invisible to filename labels but visible once transcripts are aligned to ledger signals. These include model identity gaps and taxonomy confusions (see ep135 and ep140). We classify them as understanding violations and record them in episodes.

Next we summarize the post-freeze slice as preliminary signal, not evidence.

Post-freeze signal (preliminary)

We also reviewed the post-freeze slice recorded after `stack-freeze-2026-01-12`. Five episodes exist in that window (ep132-ep136). One is violation-labeled (ep133). The sample is too small to support a performance claim. We treat it as a hypothesis for the next freeze (v0.1.1), not evidence.

Table 2. Pre-freeze vs post-freeze episode window summary.

Window	Episodes	Violation-labeled	Status
Pre-freeze	See freeze window above	See freeze window above	Evidence
Post-freeze	5 (ep132-ep136)	1 (ep133)	Reconstruction

This table is informational only and does not support a performance claim.

Understanding violations are tracked separately from filename labels. In the post-freeze sample (ep132-ep136), one episode records an understanding violation (ep135), yielding a preliminary 1/5 rate. This is reconstruction, not evidence.

Hypothesis H1: freezing the OS reduces understanding violations across models by stabilizing governance context. We will quantify this in the next freeze by comparing understanding-violation rates pre- and post-freeze using transcript-ledger alignment.

That hypothesis motivates the attribution analysis below.

Agent attribution and performance (pre-freeze)

Using transcript logs to resolve model identity (ep130) reveals that agent labels were proxies for underlying models. Within the freeze window, Claude Sonnet 4.5 shows the lowest violation rate, followed by Claude Opus 4.5; OpenAI GPT-5.x is moderate; MiniMax M2.1 is worst. This ranking is evidence-bound but incomplete because early episodes lack model identity (ep135).

This attribution framework also sets up App/IDE comparisons: if transcripts record whether a session ran in the App or IDE channel, ledger signals can be segmented by channel.

We next summarize the compression trend that stabilized governance.

The Compression

Governance shrank by iteration. Root axioms were reduced to three, and vocabularies stabilized at scope level. The change is visible across the refactoring, minimalism, and fixed-point episodes (ep019, ep053, ep060).

The evidence links are provided in Appendix F (claim-to-evidence map). We surface that map next, then interpret implications.

Evidence Links (Key Claims)

This section records where to find the claim-to-evidence map without breaking the narrative.

The claim-to-evidence map is moved to Appendix F to keep the main text contiguous.

You do not have to trust us. Clone the repos. Replay the history. The evidence is the system that produced the evidence.

Traditional papers describe experiments that happened elsewhere; this paper is the experiment it describes. The method, results, and limitations all derive from the same ledger. With evidence mapped, we can discuss why it matters.

Why This Matters

For Scientific Publishing

If this approach generalizes, papers could be verifiable by construction. Instead of "trust the authors," check the ledger.

Peer review could become: "Does every claim link to evidence? Do the commits exist? Is the governance valid?" Mechanical verification replaces subjective trust.

For AI Collaboration

The framework improves attribution when model identity is recorded: which model, which session, which commits.

It also clarifies authority. The AI can contribute, but it cannot change the rules. Insight-law separation keeps governance human.

With transcript-ledger alignment, we can rank agent performance by violation rate (ep130) and extend the analysis to App vs IDE channels when app exports are captured.

For AI Slop

The primitives target AI-slop-like failure modes by making them invalid to commit:

Table 3. CANONIC primitives and the failure modes they block.

Primitive	What It Blocks
Triad	Incomplete scopes
Inheritance	Invented authority
Introspection	Undefined jargon
Immutability	Polished-away mistakes
Model disclosure	Anonymous AI
Ledger-first	Unsupported claims
Insight-law separation	AI self-promotion

This is not a filter. It is a governance boundary.

We now state the limits of this study.

Limitations (Honest Ones)

We do not claim:

- **Optimality**: These seven primitives work here. Fewer might suffice.
- **Generalizability**: This worked for governance specs. Other domains may differ.
- **Scalability**: 9 public repos, 129 episode artifacts at `stack-freeze-2026-01-12`. Enterprise scale is unproven.
- **Model identity completeness**: Some early episodes lack explicit model IDs; this gap is documented post-freeze in ep135.
- **SPEC coverage**: Some AI-generated artifacts lacked explicit SPEC boundaries; documented post-freeze in ep145.
- **Transcript coverage**: Session transcripts are not uniformly ledger-anchored; App/IDE attribution is incomplete without explicit provenance in commits.

The study is bounded by one frozen ledger. Claims are observations within that window.

Supporting materials formalize the proofs and checks that back these limits.

Supporting Materials

We include supporting materials that formalize why AI slop is structurally inadmissible and how evidence can be reconstructed:

- Mathematical appendix and proof sketches (`paper/supporting/appendix-a-proofs.md`).
- Validation artifacts and checks (`paper/supporting/appendix-b-validation-artifacts.md`).
- Transcript record index (hash-only) (`paper/supporting/appendix-c-transcript-index.md`).
- Stream index references (`paper/supporting/appendix-d-stream-index.md`).
- Reproducibility methods (`paper/supporting/appendix-e-methods.md`).
- Claim admissibility checklist (`paper/supporting/appendix-f-claim-checklist.md`).

These appendices are descriptive and do not expand governance.

Try It Yourself

The system is in the ledger. Clone it. Check compliance. Trace any claim to its commit.

```
# Example: writing repo evidence

git clone https://github.com/canonic-machine/writing.git
cd writing
git checkout stack-freeze-2026-01-12
```

For the full stack list, see `stack/public/STACK.yml` in the stack repo.

Conclusion

We asked: can a governed human-AI system produce a self-evidencing scientific paper?

Within this evidence window, the answer is yes.

Seven primitives. 129 episode artifacts. One frozen ledger. Key claims trace to evidence. Every correction is preserved.

Constitutional governance makes verifiability structural, not procedural. The paper does not just report an experiment; it is the experiment, and you can replay it.

Evidence

Evidence window: `stack-freeze-2026-01-12` (tag timestamp `2026-01-12T18:34:47-05:00`, commit `writing:f8acf128`)

First CANON artifact: `writing/CANON.md` commit `writing:bca9ec0` at `2026-01-05T14:13:20-05:00`

Time to freeze: 7 days, 4:21:27 (from first CANON to freeze tag timestamp)

Public repositories (tagged at `stack-freeze-2026-01-12`): - canonic:0b063b8 - machine:a57f159 - os:4c2919d - ledger:3b95de2 - writing:f8acf12 - paper:0ee1970 - stack:f58ad6d - validators:e772048 - patents:4bd3dd0

Key episodes: ep019 (refactoring), ep053 (root minimalism), ep060 (minimal axioms), ep130 (agent-model correlation), ep133 (ledger-signal patterns), ep131 (full stack triad compliance)

Post-freeze corrections (not part of the evidence window): ep135 (model identity gap), ep140 (primitive misunderstanding correction), ep144 (stream model identity gap), ep145 (AI work product missing SPEC), ep148 (LFG enforcement gap), ep149 (timeline annotation gap)

Freeze declaration: Dexter Hadley, 2026-01-12

This revision was produced under CANONIC governance using OpenAI Codex (GPT-5). Model: gpt-5 (Codex CLI) Key claims trace to the frozen ledger.

Appendix A - Mathematical Appendix (Proof Sketches)

These are proof sketches that formalize why AI slop is structurally inadmissible under CANONIC governance. They are descriptive and bounded to this system.

Notation

Let:

- $\mathcal{C}$ be the set of claims in PAPER.
- $\mathcal{E}$ be the set of episodes.
- $\mathcal{L}$ be the set of ledger evidence (commits and tags).
- $\mathcal{V}$ be the vocabulary (defined terms in VOCAB).
- $\text{Terms}(c)$ be the set of terms used by claim c.
- $T(c, \ell)$ be the traceability predicate ("c traces to ledger item ℓ").
- $B(c, e)$ be the bounding predicate ("c is bounded by episode e").
- $V(c)$ be vocabulary closure for c.
- $A(c)$ be admissibility of c.
- $S(c)$ be AI slop for c.
- $U(t)$ be undefinedness of term t.
- Invalid(scope) indicate a scope fails validity constraints.

Definitions

D1. Vocabulary closure

$$V(c) \iff \text{Terms}(c) \subseteq \mathcal{V}$$

D2. Admissible claim A claim is admissible if and only if it is traceable to ledger evidence, bounded by episodes, and vocabulary-closed.

$$A(c) \iff \exists \ell \in \mathcal{L}, \exists e \in \mathcal{E} : T(c, \ell) \land B(c, e) \land V(c)$$

D3. Undefined term Let $\mathcal{R}$ be the set of terms appearing in CANON/README.

$$U(t) \iff t \in \mathcal{R} \land t \notin \mathcal{V}$$

D4. AI slop AI slop is any claim that fails admissibility or uses undefined terms:

$$S(c) \iff \neg A(c) \lor \exists t \in \text{Terms}(c) : U(t)$$

Lemma 1 - Unsupported claims are inadmissible

If a claim lacks ledger evidence, it is not admissible.

$$\neg \exists \ell \in \mathcal{L} : T(c, \ell) \Rightarrow \neg A(c)$$

Sketch. Direct from D2.

Lemma 2 - Vocabulary violations invalidate admissibility

If a claim uses an undefined term, it is inadmissible.

$$\exists t \in \text{Terms}(c) : U(t) \Rightarrow \neg A(c)$$

Sketch. By D1 and D2, $V(c)$ is required for $A(c)$. If $U(t)$ appears in $\text{Terms}(c)$, then $V(c)$ fails and so does $A(c)$.

Proposition 1 - AI slop is structurally rejected

AI slop cannot be admitted as a claim in CANONIC PAPER.

$$S(c) \Rightarrow \neg A(c)$$

Sketch. D4 implies either $\neg A(c)$ directly or a vocabulary violation, which by Lemma 2 implies $\neg A(c)$.

Proposition 2 - Undefined governance terms invalidate scope

If a governing term is undefined, the scope is invalid.

$$U(t) \Rightarrow \text{Invalid(scope)}$$

Sketch. By introspection, every term in CANON/README must be in $\mathcal{V}$. If not, the scope fails validity and cannot yield admissible claims.

Proposition 3 - Ledger immutability preserves traceability

Let L_t be the ledger state at time t. Immutability implies:

$$L_t \subseteq L_{t+1}$$

Sketch. Corrections add new records instead of overwriting old ones. Thus evidence chains are monotonic and fully reconstructable.

Corollary - Non-erasure of evidence

For any correction chain $x \to x'$,

$$x \in L_t \land x' \in L_{t+1}$$

so both the original and correction remain visible for forensic reconstruction.

Appendix B - Validation Artifacts

This appendix lists validator artifacts used for supporting checks.

Transcript record validator (stub)

- `validators/transcript/validate_transcript_records.py`
- Purpose: basic metadata checks for transcript record format.
- Inputs: `transcript/records/*.md`
- Outputs: pass/fail with missing fields reported.

Validator confidentiality

Validator implementations may be private. This appendix lists only public artifacts and reference-only checks.

Appendix C - Transcript Record Index

Transcript records are stored as hash-only metadata in `transcript/records/`.

See `transcript/records/README.md` for the current index.

App transcript records are placeholders unless a timestamp and identifier are provided (privacy-preserving). No raw transcript content is stored.

Appendix D - Stream Index References

Stream indexes are stored under `writing/streams/`.

The top-level index is `writing/streams/README.md`. Each stream README lists episode entries with channel and source metadata.

Appendix E - Reproducibility Methods

This appendix summarizes steps to reproduce key checks in the frozen evidence window. It is descriptive and does not define governance.

E1. Evidence window checkout

1. Clone the repo
2. Checkout the freeze tag
3. Confirm commit IDs

Example:

```
git clone https://github.com/canonic-machine/writing.git
cd writing
git checkout stack-freeze-2026-01-12
```

E2. Triad compliance checks

1. Verify `CANON.md`, `VOCAB.md`, `README.md` exist for each scope
2. Verify `LICENSE` and `NOTICE` at repo roots

E3. Claim traceability

1. Locate claim in manuscript
2. Follow cited episode or ledger reference
3. Confirm evidence exists in frozen ledger

E4. Stream indexing

1. Open `writing/streams/README.md`
2. Navigate to the referenced stream
3. Verify episode membership and metadata

E5. Transcript record validation (optional)

1. Use `validators/transcript/validate_transcript_records.py`
2. Confirm required metadata fields are present

E6. Learned episode checks

1. Confirm transcript record IDs are present
2. Confirm inception mapping cites earliest record and stream reference

---

E7. PDF render (optional)

1. Run `paper/scripts/build_pdf.sh`
2. Confirm output at `paper/output/manuscript.pdf`

E8. PDF output validation (optional)

1. Run `validators/paper/validate_pdf_output.py`
2. Confirm it reports the output file

E9. Figure rendering (optional)

1. Render Mermaid sources from `paper/figures/*.mmd` to SVG and PDF using `mermaid-cli`.
2. Confirm the corresponding `paper/figures/*.svg` and `paper/figures/*.pdf` assets exist.

Appendix F - Claim Admissibility Checklist

Use this checklist to verify that a claim is admissible under PAPER CANON.

Checklist

- Claim has a cited ledger reference (commit or tag).
- Claim is bounded by recorded episodes.
- Claim uses only defined terms (VOCAB closure).
- Claim is stated as an observation, not a new rule.
- Evidence exists within the freeze window when required.
- Transcript references (if any) are hash-only and privacy-preserving.

Claim-to-evidence map - freeze window

Claim	Evidence reference
"129 episode artifacts"	`writing/episodes/` file count at tag `stack-freeze-2026-01-12`
"33 violation-labeled episodes"	`writing/episodes/*violation*` file count at tag `stack-freeze-2026-01-12`
"12 triad scopes across 9 repos"	`writing/episodes/ep136-stack-compliance-reports.md` (Report A)
"First CANON timestamp"	`writing/CANON.md` initial commit `writing:bca9ec0`
"Freeze timestamp"	tag `stack-freeze-2026-01-12` (tag timestamp `2026-01-12T18:34:47-05:00`, commit `writing:f8acf128`)
"OS is minimal substrate (frozen) for downstream work"	`writing/episodes/ep100-os-initiation-and-refactor.md`, `writing/episodes/ep101-canonic-os-minimality-report.md`
"Claude Code co-authored early CANONIC foundation commits"	`canonic:49d6b65`
"Agent-model correlation and violation ranking"	`writing/episodes/ep130-agent-model-correlation.md`
"Model identity gap (understanding violation)"	`writing/episodes/ep135-model-identity-gap.md`
"AI work product without SPEC boundary (post-freeze)"	`writing/episodes/ep145-ai-work-product-missing-spec.md`
"Cloud discovery to CANON drift hypothesis"	`writing/episodes/ep146-cloud-discovery-canon-drift-hypothesis.md`
"Positive control discovery lexicon"	`writing/episodes/ep147-positive-control-lexicon.md`
"LFG signal enforcement gap"	`writing/episodes/ep148-lfg-enforcement-gap.md`
"LFG timeline annotation gap"	`writing/episodes/ep149-lfg-timeline-annotation-gap.md`
"Ledger-signal verification via session transcripts"	`writing/episodes/ep133-ledger-signal-violation-patterns.md`

Notes

This checklist is descriptive and does not define governance.