contextctl: Auditable, Open-Format Context Infrastructure for AI Agents

Ayan Putatunda and Suhas Jangoan
IEEE, USA {ayan.putatunda, suhas.jangoan}@ieee.org

***Abstract*—AI agents act on remembered facts, yet common memory services keep those facts in vendor schemas, keep no content-hash link to the source, and accept agent writes without review. We demonstrate contextctl, which manages agent context as governed data: markdown concept files in the Open Knowledge Format (OKF), Git as write gate and transaction-time record, a content-addressed evidence folder, and a rebuildable SQLite sidecar holding slowly changing dimension (SCD) Type 2 history. The demo runs the storage, versioning, indexing, and a minimal review gate with no language model, including a replay of a public dbt project's history. Accuracy, gate efficacy on model-written drafts, and the two proposed integrity properties are not evaluated here; they remain planned.**

***Index Terms*—agent memory, data provenance, bitemporal data, version control, governance**

I. Introduction

During a migration, an incident-response agent learns that a critical pipeline belongs to an engineer who has left, and routes alerts to nobody for three weeks. Operators cannot say when it learned this, from what source, what it believed before, or how to restore the correct state with proof, although the right owner was in a Slack thread all along.

The gap. Memory services such as Mem0, Letta, and Zep [3], [5], [6] use vendor schemas, lack content-hash links to sources, and trust agent writes. Plain files are a sound base: on LoCoMo, Letta reports 74.0% for an agent over files, above the 68.5% Mem0 reported for its graph variant [3], [4]. The closest recent systems each cover part of the problem (Appendix A): review-gated, hash-cited knowledge (Vouch [7], Assay [8]) or bitemporal facts in SQLite (Memharness [9]). In their public descriptions we found none that documents all four of: (1) custody of the source bytes behind each belief; (2) an explicit authorization step before machine-written knowledge becomes trusted; (3) reconstruction of what was believed, and when; (4) serving only the context a task needs. contextctl targets that combination. No single mechanism is new.

Why this is a Big Data problem. Our measured volume is modest (256 MB at 10^5 synthetic concepts); the argument is operational. Evidence grows without bound, so ingestion must be incremental (watermarks) and idempotent (content addressing stores identical bytes once). Histories grow with every merge, so indexes must be derived and rebuildable rather than a second write path. Heterogeneous sources (repositories, chat, wikis) must each reduce to one evidence object. Review throughput becomes the bottleneck, so reconciliation must remove duplicates before a human sees them: in our replay, 322 of 373 drafts were classified Duplicate.

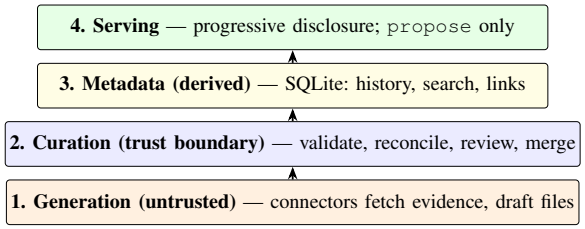


Fig. 1. The four layers. Trusted state is layer 2: OKF files, Git history, and the evidence folder. Layer 3 can be deleted and rebuilt.

Contributions. (i) A four-layer design with a precisely scoped evidence object and reconciliation rule (Sec. II); (ii) a working no-model prototype and replay harness for the storage, versioning, indexing, and gate core (Sec. III); (iii) replay findings that changed the specification. We label each claim *measured*, *demonstrated*, *specified*, or *planned*.

II. Design

Concepts are markdown files in OKF [1]; our OKF-MEM header profile (OKF plus memory fields) adds `actor`, `evidence`, `source`, `confidence`, `supersedes`, and validity dates. Fig. 1 shows four layers, each with one job. Generation is untrusted; curation is the only path into trusted state; the SQLite sidecar is a derived cache; serving may only propose writes, which re-enter staging.

Evidence object and digest scope. At fetch time a connector stores the retained raw bytes at `evidence/`h, where h is the SHA-256 digest of *those bytes only*. The source's uniform resource identifier (URI) and retrieval time sit outside h, in the concept's `sources` header and in a metadata record written beside the blob when it is first stored; both are bound to the bytes only through Git history, which is why a coordinated rewrite of a header and its blob passes the digest check (P1 below). Content addressing makes a retried fetch idempotent (same bytes, same h, one object) and turns an edited source into a new object while the old one is kept. Behavior when a source is deleted is not yet specified. Binary documents pass a deterministic converter whose version is recorded, so *conversion* is reproducible; language-model *drafting* is not, and its output is reviewed, not replayed.

Reconciliation. A draft d is compared with the top-k full-text matches (k=10) and the same-path concept c_0, using TF–IDF (term frequency–inverse document frequency) cosine similarity sim and $fields(d, c)$, the *fact* header fields that both assert with different values (in the replay, 14 dbt

TABLE I
RECONCILIATION RULES, APPLIED IN ORDER (θ_{dup}=0.9, θ_{rev}=0.5)

Rule	Condition	Action
Conflict	$fields(d, c_0) \neq \emptyset$	show both values; human resolves; record `supersedes`
Duplicate	$\max sim \geq \theta_{\text{dup}}$, no new facts	drop draft; log match
Revision	$\theta_{\text{rev}} \leq \max sim < \theta_{\text{dup}}$	rewrite as edit of best match
New	$\max sim < \theta_{\text{rev}}$	add as new concept

metric fields such as `expr` and `dimensions`). Provenance fields (`evidence`, `source`, `actor`, `confidence`, validity dates) are excluded, so a provenance-only update never raises a Conflict; with an unchanged body it is classed Duplicate and dropped, which loses the newer citation, a gap we flag. Facts stated only in the body affect only sim. The first matching rule of Table I applies. A Duplicate or Revision whose best match is a *different* concept (a rename or a sibling) is held for review rather than applied.

Metadata and time. A post-merge hook maintains one SQLite file: a slowly changing dimension Type 2 (SCD2) [2] table `belief_history`, a full-text search (FTS5) index, and a table of links between concepts. `belief_history` tracks header fields plus a digest of the body, so a body change is detected but not itemized; an owner named only in the body is not a queryable field. Valid time comes from headers, transaction time from Git. Appendix B gives measured rows, a query, and a retroactive correction. The standing test is to delete the sidecar, rebuild it, and compare a canonical row dump.

Serving. A specified progressive-disclosure protocol (orient, search, read, history, provenance) plus a staging-only `propose` tool, through a command-line interface (CLI), a Model Context Protocol (MCP) server [10], and a plugin. Only part of the CLI is built.

Proposed properties (hypotheses). Under (A1) SHA-256 collision resistance, (A2) branch protection enforced as configured, and (A3) byte-reproducible conversion: *P1 (evidence-chain integrity)*: each hop concept→commit→evidence→source has a check. For a coordinated header-and-blob rewrite, detection rests on Git history (see digest scope above). *P2 (completeness)*: every machine-written trusted concept has a resolving chain, because the gate rejects unverifiable digests and A2 excludes other paths. On common Git hosts administrators can bypass protection, so P2 holds only where protection binds them. Signed commits would narrow the gap but are optional in the current design. A draft that sets its own `actor` or `confidence` could claim human authorship or lighter review; the gate should set `actor` from an authenticated identity (not implemented).

III. WHAT THE DEMO SHOWS

Built: bundle, evidence store and digest verification, a minimal reconciler and gate, the SQLite sidecar, and a rule-based notes connector. **Specified only:** repository, document, Confluence, Slack, and email connectors; the full CLI; the MCP server; the plugin. All numbers below are *measured* on that built core (details and environments in Appendix C). They show the substrate works; they are not evidence for P1, P2, accuracy, or token savings.

- *Cost:* snapshot commits of 155 ms at 10^4 concepts and 634 ms at 10^5; full sidecar rebuilds under 1 s; sub-millisecond point-in-time queries (synthetic data).
- *Checks:* a nine-check no-model lifecycle passed; in the replay, as-of reads matched `git show` in 665/665 cases and the rebuilt sidecar matched the incremental one row for row.
- *Replay of the public dbt Jaffle Shop history* (scripted admission, no model): 373 drafts gave 16 New, 322 Duplicate, 24 Revision, and 11 Conflict; each Conflict corresponds to an upstream edit of `dimensions` or `expr`.

What the replay taught us. (a) Checking Conflict against every candidate flagged 359 of 373 drafts, so it is now scoped to the same path. (b) A coordinated header-and-blob rewrite passed the digest check and showed only as an uncommitted Git change. (c) The full governed bundle cost about $4.6\times$ the tokens of the raw sources, because digests and URIs tokenize poorly. (d) Reverting one merge conflicted with a later merge of the same files.

IV. LIMITATIONS AND PLANNED EVALUATION

This demo has no accuracy result. Planned (not pre-registered) research questions (RQs): **RQ1** accuracy parity, not gains, with memory services on LoCoMo and LongMemEval [11], [12]; **RQ3** whether disclosure tiers beat flat retrieval in tokens at equal correctness, given (c); **RQ5** gate efficacy on model-written and adversarial drafts; **RQ6** systematic tamper detection. The scale check uses two sizes of single-template synthetic data and is not a scaling result. The replay has no model and no human reviewer, and the opening Slack/Confluence scenario is not tested. Retaining evidence conflicts with deletion rights, and harvesting Slack or email keeps employees' messages, which raises consent and access questions.

Conclusion. contextctl is a partially built stack for managing agent context as governed data. The demo shows its storage, versioning, indexing, and gate core working on open components, and a model-free replay finding specification bugs.

AI USAGE DISCLOSURE

Large language model (LLM) tools assisted with (1) writing and debugging the prototype and benchmark harnesses, (2) drafting and revising the manuscript, including simulated peer review, and (3) finding and checking related work. The authors verified all AI-assisted output and take full responsibility for the content and references. All reported measurements come from scripts that were executed, not from model output.

APPENDIX A
CLOSEST SYSTEMS

Memory services and Git-versioned reasoning tools (GCC, GitOfThoughts [13], [14]) address adjacent problems. Closer systems realize pieces of our design: Vouch is a Git-native, review-gated knowledge base whose claims cite sources, exposed through MCP and a CLI [7]; Memharness keeps bitemporal facts with provenance in SQLite behind MCP [9]; a preprint describes a replayable bitemporal memory substrate [15]; Git-bound memory for coding agents uses merge as the admission gate [16]; Assay binds claims to content-hashed evidence reviewed before merge [8]. Table II uses only those descriptions ("–": none documented, not a verified absence); we ran no systematic survey.

TABLE II
CLOSEST SYSTEMS (PUBLIC DESCRIPTIONS, ACCESSED OCT. 2026). THE CONTEXTCTL COLUMN IS DESIGN INTENT; SEE SEC. III FOR WHAT IS BUILT.

Property	Vouch	Memharness	Assay	contextctl
Review gate before belief	yes	–	yes	built (min.)
Content-hash evidence binding	yes	–[a]	yes	built
Bitemporal / SCD2 sidecar	–	yes	–	built (min.)
Governed Slack/wiki ingestion	–	–	–	specified
Specified reconciliation rules	–	–	–	built (min.)
Token-tiered serving	–	–	–	specified

[a] Per-fact provenance; we do not equate it with content-hash binding.

APPENDIX B
WORKED BITEMPORAL EXAMPLE (MEASURED ROWS)

Upstream Jaffle Shop commit `a75613b` (2024-07-04) changed the `food_revenue` expression from `= 1` to `is true`; it reached the mainline in a 2024-09-12 merge, and a 2025-07-01 merge changed it again. The replay held that harvest as a Conflict on `expr` until scripted approval. Resulting rows (abbreviated):

old	new	valid_from	valid_to	snapshot
–	`= 1`	2023-08-30	2024-09-12	16c4a6f
`= 1`	`is true`	2024-09-12	2025-07-01	6104f5c
`is true`	bare boolean	2025-07-01	open	11917f9

"What was the expression on t?" (valid time) is one range query: `SELECT new_value FROM belief_history WHERE concept=:c AND field='expr' AND valid_from<=:t AND (valid_to IS NULL OR valid_to>:t)`; for t = 2024-06-01 it returns `= 1` and for 2025-01-01 `is true`, matching the replay. The transaction-time question (what the bundle said at a snapshot) is an as-of read at that commit.

Retroactive correction (derived from the hook's rule, not run). The fix actually applied from 2024-07-04, not 2024-09-12. If the admitted draft carries `valid_from: 2024-07-04`, the hook closes row 1 at 2024-07-04 and opens row 2 at 2024-07-04 while its snapshot (transaction time) stays 2024-09-12. A valid-time query for 2024-08-01 then returns `is true`, but an as-of read of the bundle on 2024-08-01 still returns `= 1`: the two time axes differ exactly over the period in which the agent was wrong. A later commit that edits *only* `valid_from` is recorded in Git but not in `belief_history` in the current harness, a gap we flag.

Rollback. Reverting only the 2024-09-12 merge failed, since the 2025-07-01 merge edited the same files; reverting it and the two later merges restored it.

APPENDIX C
MEASUREMENT DETAILS

Microbenchmarks (single vCPU; Git 2.43, SQLite 3.45, Python 3.12; medians): bundles of 10^2–10^4 concepts, 30 commits touching 1% of files; at 10^4, snapshot commit 155.1 ms, as-of read and revert about 3–5 ms, sidecar rebuild 0.47 s. **Two-size scale check** (same class of environment; 10^4 and 10^5 single-template concepts; 10 commits touching 0.1%): at 10^5, history rebuild 0.67 s, FTS5 build 4.3 s, link extraction 1.7 s, snapshot commit 634 ms; storage 88/25/143 MB (bundle/Git/sidecar). Path-scoped staging was not measured. **Replay** (8 vCPU VM; Git 2.47, SQLite 3.53; Jaffle Shop pinned at commit `7be2c58`; 7 repeats, identical counts): 25 steps, 7 merges, 22 final concepts. Tokens (tiktoken `cl100k_base`, medians): orient+search+read 2,026; full bundle 30,460; raw dbt files 6,579; a full concept read 1,339 versus 190 for its body; a body-only orient+search+read variant 880. No flat-retrieval baseline was run.

REFERENCES

[1] S. McVeety and A. Hormati, "How the Open Knowledge Format can improve data sharing," Google Cloud Blog, Jun. 12, 2026.
[2] R. Kimball and M. Ross, *The Data Warehouse Toolkit*, 3rd ed. Wiley, 2013.
[3] P. Chhikara, D. Khant, S. Aryan, T. Singh, and D. Yadav, "Mem0: Building production-ready AI agents with scalable long-term memory," in *Proc. ECAI*, FAIA vol. 413, 2025, pp. 2993–3000.
[4] Letta, "Benchmarking AI agent memory: Is a filesystem all you need? Letta blog, Aug. 12, 2025. [Online]. Available: letta.com/blog/benchmarking-ai-agent-memory
[5] C. Packer *et al.*, "MemGPT: Towards LLMs as operating systems," arXiv:2310.08560, 2023.
[6] P. Rasmussen *et al.*, "Zep: A temporal knowledge graph architecture for agent memory," arXiv:2501.13956, 2025.
[7] "vouch: A git-native, review-gated knowledge base for AI agents," GitHub repository vouchdev/vouch, accessed Oct. 2026.
[8] O. S. Tiwari *et al.*, "Assay: Claims that decay with the code. Content-addressed evidence graphs for accountable AI-assisted software delivery," arXiv:2609.36170, 2026.
[9] "memharness: A bi-temporal, provenance-carrying memory primitive for AI agents," GitHub repository las7/memharness, accessed Oct. 2026.
[10] Anthropic, "Introducing the Model Context Protocol," Nov. 2024. [Online]. Available: anthropic.com/news/model-context-protocol
[11] A. Maharana *et al.*, "Evaluating very long-term conversational memory of LLM agents," in *Proc. ACL*, 2024, pp. 13851–13870.
[12] D. Wu *et al.*, "LongMemEval: Benchmarking chat assistants on long-term interactive memory," in *Proc. ICLR*, 2025.
[13] J. Wu *et al.*, "Git Context Controller: Manage the context of LLM-based agents like Git," arXiv:2508.00031, 2025.
[14] P. C. Shekar *et al.*, "GitOfThoughts: Version-controlled reasoning and agent memory you can replay, diff, and merge," arXiv:2606.14470, 2026.
[15] M. Vidrih, "Who knew what, when, and why: A replayable, bitemporal memory substrate for AI agents," preprint, 2026, doi:10.5281/zenodo.21654958.
[16] F. Guo, "Why Git is the memory solution for the agentic development lifecycle," arXiv:2607.14390, 2026.