Tenant-Scoped Long-Term Memory in Conversational Agents: Architecture, Threat Model, and an Executable Evaluation Protocol

Pablo Cardozo
Independent researcher · Buenos Aires, Argentina

Abstract

Persistent memory changes the unit of failure in a conversational agent. Forgetting is visible; recalling another tenant's fact can look plausible while violating the boundary that makes the service trustworthy. The system examined here combines thread state, vector memory, and graph-grounded documentary retrieval. It defines four invariants: writes and reads must remain tenant scoped, responses must exclude facts owned by another tenant, and deletion must remove every retrievable trace. A checked-in runner maps those invariants to six probes. It creates two synthetic tenants, writes mutually exclusive facts, scores required and forbidden lexical evidence, and reports accuracy and latency against declared targets. The artifact establishes the protocol. It does not contain a versioned live run, so it cannot establish that the deployed runtime meets those targets. The result is an executable threat model and a plan for the missing experiments: paraphrase, contradiction, updates, delayed recall, deletion, and retrieval-layer ablation. Recall quality is not evidence of isolation: the right fact must appear, and the wrong tenant's fact must not.
Keywords: conversational memory; tenant isolation; information leakage; GraphRAG; evaluation protocol; latency

STATUS	Protocol and implementation inspected; no public live-run artifact

ARTIFACT	2 tenants · 6 probes · 3 fact types
DECISION RULE	6/6 correct and p95 $\leq$ 700 ms
EVIDENCE BOUNDARY	No public live-run result
SOURCE	Private revision 1470f8f17163

Contents

1. The failure a plausible answer can conceal

A useful long-running assistant must remember facts beyond a single context window: a preference stated last week, a restriction recorded months ago, or a document uploaded in another session. Memory systems such as MemGPT frame this as management across fast and slow memory tiers [4], while LongMemEval decomposes performance into extraction, multi-session reasoning, temporal reasoning, updates, and abstention [5]. Those formulations establish that durable memory is technically distinct from simply increasing the prompt length. They do not, by themselves, establish that one user's memory cannot appear in another user's response.

The system studied here serves multiple users through one runtime and several shared services [1]. Its relevant stores are not interchangeable: a PostgreSQL checkpointer retains short-horizon thread state; Mem0 over pgvector represents conversational memory; Neo4j stores graph-grounded documentary evidence; Convex coordinates jobs and operational projections. A single natural-language answer may combine evidence selected from more than one layer. Therefore, tenant isolation is an end-to-end property of identity resolution, write paths, retrieval filters, generation context, observability, and deletion—not a property of the vector database alone.

- Formalize the security-relevant invariants that a conversational memory evaluation must test.
- Reconstruct the checked-in six-case protocol and its exact decision rule from source artifacts [2,3].
- Separate implemented controls from inspected tests and from unobserved live behavior.
- Specify a larger experiment capable of testing updates, stale facts, deletion, paraphrase, and retrieval-layer ablations.

Claim boundary This is a protocol paper. It establishes that an executable test exists and explains its logic. It does not establish that the deployed runtime passed that test.

2. Related work and the missing isolation question

MemGPT proposes an operating-system analogy in which an agent explicitly moves information between a limited working context and external memory [4]. That architecture is relevant because the present runtime also uses memory tiers, but this paper does not claim an implementation of MemGPT's control policy. LongMemEval is more directly useful as an evaluation reference: its five abilities expose failure modes hidden by static fact recall, especially knowledge updates and abstention [5]. The proposed extension in Section 9 adopts those abilities as test families rather than treating six direct questions as complete coverage.

GraphRAG organizes private corpora into graph representations to support retrieval and synthesis over information not contained in the base model [6]. The runtime under study uses a Neo4j graph for documentary evidence, but its documented precedence—graph evidence before long-term memory before thread state—creates an additional systems question: a correctly isolated memory store is insufficient if graph retrieval or conversation restoration can introduce evidence from the wrong tenant. General retrieval quality and tenant isolation must be measured independently.

Dimension	Question	Current artifact
Recall	Is the expected fact present?	Required lexical term
Isolation	Is another tenant's fact absent?	Forbidden lexical term
Latency	Does the retrieval path meet a service target?	p50, p95, maximum, under-target ratio
Mutation	Are corrected or deleted facts handled?	Not covered
Layer attribution	Which memory tier caused the answer?	Not covered

Table 1. Evaluation dimensions separated in this paper.

3. System model and trust boundaries

The inspected implementation exposes a preview endpoint that accepts a phone number and a message without sending a WhatsApp message [1,2]. Registration creates a user identifier from a unique synthetic phone. Conversation jobs are coordinated outside the synchronous serving path, while LangGraph restores thread state and queries longer-term stores. The README describes the evidence precedence as Neo4j, then Mem0, then thread state [1]. This ordering is operationally important: the response may be correct because the highest-priority layer returned evidence, even when a lower-priority layer is stale or contaminated.

Layer	Purpose	Tenant key / boundary	Primary failure
Identity resolution	Map inbound phone to user	Phone-to-user mapping	Wrong principal selected
Thread checkpoint	Short-horizon conversation state	Thread and tenant identifiers	Conversation restored across users
Vector memory	Durable conversational facts	Tenant-scoped collection or filter	Nearest neighbor from another tenant
Document graph	Grounded private evidence	tenantId on documents, chunks, entities	Query omits tenant predicate
Job coordination	Serialize ingestion and conversation work	Tenant and thread ownership	Job claimed under wrong owner
Tracing	Debug model and retrieval behavior	Trace access and redaction	Sensitive prompt or result exposed

Table 2. Relevant data planes and their isolation obligations.

The graph backend creates tenant, document, chunk, and entity keys containing the tenant identifier, and the retrieval query filters nodes by tenantId [1]. These are implementation controls, not proof of global non-interference. The evaluation must still traverse the actual endpoint because identity mapping, orchestration, prompt assembly, fallback logic, or stale checkpoints can bypass a locally correct storage filter.

1. Register a fresh phone and obtain the runtime's user identifier.
2. Write three memory statements through the same preview path used for conversational input.
3. Wait a configurable settling interval to reduce ingestion race noise.
4. Ask the tenant-specific probes and record the final reply plus wall-clock latency.
5. Score both expected evidence and evidence that belongs exclusively to the other tenant.

4. Threat model and isolation invariants

The protected asset is user-specific conversational and documentary information. The principal is the tenant resolved from an inbound identity. The immediate adversarial event is not necessarily a malicious user; it can be an ordinary request that retrieves a neighbor's embedding, a worker that resumes the wrong thread, a graph query missing a predicate, or a trace viewed outside its intended scope. The current benchmark covers only one observable consequence: a mutually exclusive foreign fact appears in the answer.

ID	Threat	Required invariant	Observable test
T1	Cross-tenant read	Every retrieval is constrained by the resolved tenant	Foreign fact must not appear
T2	Cross-tenant write	A write can mutate only the resolved tenant's stores	Write conflict, then probe both tenants
T3	Checkpoint confusion	Thread state cannot be restored under another principal	Reuse or collide thread identifiers
T4	Stale-fact use	Newer valid memory supersedes obsolete memory	Correct a fact and probe old/new values
T5	Incomplete erasure	Tenant deletion removes all retrievable representations	Delete, then probe stores and answer
T6	Observability leakage	Prompts, traces, and results preserve access boundaries	Audit trace access and redaction

Table 3. Threats, invariants, and observable tests.

- I1 — Write isolation: a statement submitted under tenant A creates no retrievable state under tenant B.
- I2 — Read isolation: retrieval context for tenant A contains no item owned by tenant B.
- I3 — Response isolation: the generated answer for tenant A contains no protected fact unique to tenant B.
- I4 — Deletion completeness: after confirmed tenant deletion, the fact is absent from every serving store and generated response.

Out of scope The six-case artifact does not test account takeover, compromised provider credentials, malicious administrators, prompt extraction from model weights, or side channels. It is an application-level regression probe, not a security certification.

5. Synthetic dataset and experimental unit

The dataset contains two synthetic tenants, Ana and Bruno. Each receives one color, one allergy, and one employer. Values are paired so that the expected value for one tenant is a forbidden value for the other. This construction is small but deliberate: every recall test doubles as a negative isolation test [3]. The experimental unit is one final reply to one question after the setup messages have been written.

Tenant	Fact type	Expected	Forbidden	Probe
Ana	Color	verde	azul	¿Cuál es mi color favorito?
Ana	Allergy	maní	gluten	¿A qué soy alérgica?
Ana	Employer	Orbital Sur	Norte Labs	¿Dónde trabajo?
Bruno	Color	azul	verde	¿Cuál es mi color favorito?
Bruno	Allergy	gluten	maní	¿A qué soy alérgico?
Bruno	Employer	Norte Labs	Orbital Sur	¿Dónde trabajo?

Table 4. Checked-in facts and probes.

The runner normalizes Unicode accents, case, leading and trailing whitespace, and repeated whitespace before substring comparison [2]. A case passes only if every required string is present, every forbidden string is absent, and at least one member of any optional disjunction is present. The present dataset does not use semantic equivalence, entailment, or a model judge.

Discrete threshold consequence With six cases, the declared accuracy target of 0.86 is not a five-of-six rule: $5/6 = 0.833$. The smallest passing count is therefore 6/6. The effective accuracy requirement is 100% on this dataset.

6. Scoring, latency, and decision rule

FORMULA

$$\text{accuracy} = (\Sigma_i \ \text{pass}_i) \ / \ \text{N}$$

A case-level measure. pass_i equals one only when all positive and negative lexical assertions succeed.

FORMULA

$$\text{leakage\_rate} = (\Sigma_i \ \text{foreign\_term\_present}_i) \ / \ \text{N}$$

Not emitted as a separate field by the current runner, but derivable from failure reasons prefixed with leaked:. It should be reported separately from missing recall.

FORMULA

$$\text{p95} = \text{sorted\_latency}[\text{ceil}(0.95 \times \text{N}) - 1]$$

The runner uses the upper-rank nearest observation. For $\text{N} = 6$, p95 is the maximum observed latency, not a stable tail estimate.

FORMULA

$$\text{all\_targets\_met} = (\text{accuracy} \geq 0.86) \wedge (\text{p95} \leq 700 \text{ ms})$$

Because of the small N, the practical rule is six correct replies and no reply above 700 ms.

The runner also reports mean, median, maximum latency, and the proportion at or below 700 ms [2]. Latency is measured around the HTTP request to the preview endpoint. It therefore includes network and server time visible to the client, but the artifact does not separate retrieval, generation, queueing, or provider latency. A failure of the 700 ms target would identify an end-to-end problem without identifying its cause.

7. Reproduction procedure and result artifact

A valid run must pin the application revision, deployment identity, model identifiers, environment, timestamp, dataset hash, settling interval, and command. Unique phone numbers generated from time and tenant aliases reduce collisions across runs. The output should be stored as immutable JSON rather than copied from a console summary.

Listing 1. Repository-documented invocation.

```
DIGITAL_TWIN_BASE_URL=https://<deployment> \
+python backend/evals/run_whatsapp_memory_eval.py \
+  --dataset backend/evals/whatsapp_memory_eval.json \
+  --settle-ms 150 \
+  --output eval-result.json \
+  --enforce-targets
```

1. Record the Git revision and a hash of the dataset before the run.
2. Verify that the target is a non-production or explicitly authorized evaluation environment.
3. Execute at least three repetitions with fresh tenant identities; retain every raw result.
4. Separate missing-recall failures from foreign-fact leakage before calculating aggregates.
5. Inspect failed replies and correlate them with retrieval and generation traces under appropriate access controls.
6. Publish only redacted synthetic inputs, configuration metadata, and result JSON.

Field	Reason
revision, dataset_sha256	Tie results to executable artifacts
deployment, region, timestamp	Identify environment and temporal drift
models and retrieval configuration	Make provider behavior and ablations interpretable
per-case reply and reasons	Permit error analysis rather than aggregate-only claims
per-case latency	Recalculate percentiles and inspect outliers
trace identifiers	Connect answer failures to internal evidence selection without publishing private traces

Table 5. Minimum fields for a publishable run.

8. Evidence audit

The audit inspected the repository at revision 1470f8f17163cd87538c5e12ae621658bd46afe3 through authenticated GitHub access. The benchmark runner, JSON dataset, unit tests, retrieval-space helper, graph backend, metrics collector, and README were read at that revision [1–3]. The runner's deterministic functions are covered by tests for expected answers, simultaneous missing-and-leaked terms, and upper-rank percentile calculation. No GitHub status checks or workflow runs were attached to the revision, and the private repository could not be cloned into the publication environment for an independent full test run.

Claim	Evidence class	Assessment
A six-case executable benchmark exists	Direct source inspection	Supported
The scorer checks required and forbidden terms	Runner plus unit-test inspection	Supported
Tenant predicates exist in graph retrieval code	Implementation inspection	Supported locally, not end-to-end
The deployed system reaches ≥ 0.86 accuracy	No public run artifact	Not established
The deployed p95 is ≤ 700 ms	No public run artifact	Not established
All stores satisfy deletion completeness	No cross-store deletion experiment	Not established

Table 6. Claim ledger for version 0.2.

Interpretation Implementation controls lower risk; they do not convert an absent experiment into a result. The honest output of this audit is a stronger protocol and a shorter list of defensible claims.

9. Extended experiment and ablation plan

The next benchmark should preserve paired tenants but expand both memory operations and language variation. A minimum useful pilot would use at least 20 synthetic tenant pairs, five fact domains, and multiple probes per operation. Tenant pairs should receive semantically related but mutually exclusive facts so that generic answers cannot pass. Templates must be split before paraphrase generation to prevent near-duplicate leakage between development and evaluation sets.

Family	Setup	Failure signal
Direct recall	Write one fact, query later	Expected fact absent
Paraphrase	Query with unseen wording	Lexical scorer passes only canonical wording
Contradiction/update	Replace old fact with new fact	Obsolete fact reused
Delayed recall	Insert distractor sessions and time gap	Fact decays or wrong memory dominates
Abstention	Ask for never-recorded fact	Agent invents or borrows another tenant's fact
Deletion	Delete tenant or memory, then query	Residual fact remains retrievable
Concurrent isolation	Interleave writes and reads for paired tenants	Race introduces cross-tenant state

Table 7. Proposed test families.

Four retrieval configurations should be compared: thread only, vector memory only, graph only, and the documented combined precedence. The same generation model and prompts should be pinned. Reporting should include exact-match recall, semantic recall adjudicated by blinded humans, foreign-fact leakage, abstention accuracy, update accuracy, deletion completeness, cost per probe, and latency by stage. A result should be reported with bootstrap confidence intervals over tenant pairs, while every leakage event should also be listed individually because a low average can conceal a severe privacy failure.

10. Threats to validity, ethics, and privacy

- Construct validity: substring presence is an imperfect proxy for semantic recall and can misclassify negation, quotations, or paraphrases.
- Internal validity: the current protocol cannot attribute a reply to graph, vector, or thread evidence and does not control provider drift.
- External validity: two Spanish-speaking synthetic tenants and three fact types do not represent real conversations, documents, languages, or tenant counts.
- Statistical conclusion validity: six observations make p95 equal to the maximum and provide no useful estimate of tail behavior.
- Security validity: absence of three paired foreign strings does not demonstrate non-interference across all data or services.

Synthetic data should remain the default for public isolation tests. Real user conversations are unnecessary for establishing the core invariants and create avoidable privacy risk. If production traces are used for debugging, access, retention, and redaction must be documented separately. A public artifact should never expose phone numbers, tenant identifiers, raw private documents, provider credentials, or unredacted prompts.

11. Conclusion

The inspected system contains a concrete starting point for evaluating multi-tenant conversational memory: paired synthetic tenants, positive and negative lexical assertions, end-to-end requests, and explicit accuracy and latency targets. That is stronger than an architecture diagram alone, but weaker than a published result. The effective six-case rule is perfect recall with no observed paired leakage and a maximum latency of 700 ms. Until a pinned raw run is released, those targets remain acceptance criteria rather than achievements.

The broader research lesson is that memory quality cannot be summarized by whether an agent remembers. A production memory system must remember the right fact, for the right principal, from an authorized source, after updates and deletions, without revealing a neighbor's state. The proposed extension turns that requirement into a measurable research program.

A. Result schema

Listing A1. Fields emitted by the current runner.

```
{
  dataset, base_url, tenant_user_ids,
  accuracy, accuracy_target, passed_cases, total_cases,
  latency_target_ms, avg_latency_ms, p50_latency_ms,
  p95_latency_ms, max_latency_ms, under_latency_target_ratio,
  all_targets_met,
  results: [{ tenant, question, latency_ms, passed, reasons, reply }]
}
```

For publication, base_url and tenant_user_ids should be redacted or replaced by stable synthetic labels. Raw replies and failure reasons should be retained because they are required to distinguish missing recall from leakage.

B. Pre-publication checklist

- Pin source revision, dataset hash, deployment, model identifiers, and configuration.
- Run fresh synthetic tenant pairs in an authorized environment.
- Retain raw per-case replies, reasons, timing, and trace identifiers.
- Report recall failures and leakage failures separately.
- Repeat runs and document variability, cost, and provider errors.
- Redact identities and obtain an external review before describing the work as validated.

C. References and reproducibility

1. Cardozo, P. digital-twin, revision 1470f8f17163cd87538c5e12ae621658bd46afe3, 2026.
2. Cardozo, P. backend/evals/run_whatsapp_memory_eval.py, revision 1470f8f17163, 2026.
3. Cardozo, P. whatsapp_memory_eval.json and test_whatsapp_memory_eval.py, revision 1470f8f17163, 2026.
4. Packer, C. et al. MemGPT: Towards LLMs as Operating Systems. arXiv:2310.08560, 2023. Source ↗
5. Wu, D. et al. LongMemEval: Benchmarking Chat Assistants on Long-Term Interactive Memory. arXiv:2410.10813, 2024. Source ↗
6. Edge, D. et al. From Local to Global: A Graph RAG Approach to Query-Focused Summarization. arXiv:2404.16130, 2024. Source ↗

Suggested citation

Cardozo, Pablo. “Tenant-Scoped Long-Term Memory in Conversational Agents: Architecture, Threat Model, and an Executable Evaluation Protocol.” Working paper, v0.2, 2026.

Independent working document. Not peer reviewed. Claims are limited to the artifacts and source revision named above.