Memory Belongs to the User:
Portable Memory, an Open Standard Proposal for Cross-Vendor AI Memory

Sergii Kryvoblotskyi
MacPaw AI&Research
krivoblotsky@macpaw.com

Nataliia Stulova
MacPaw AI&Research
nata.stulova@macpaw.com

Vladyslav Hamolia
MacPaw AI&Research
vhamolya@macpaw.com

Standards proposal – preprint for community review. July 2026.

Abstract. AI assistants became systems that *remember*: preferences, projects, relationships, and history now persist across sessions and increasingly shape what these systems do for us. That memory is the most personal dataset a person delegates to software — and today it is locked inside each vendor's proprietary storage, in incompatible shapes, with no way to move it and no way to prove it was ever deleted. We believe this is the data portability gap of the AI era, and that it should be closed the way the web closed it for documents, mail, and calendars: with an open format. This paper proposes **Portable Memory**, an open, vendor-neutral standard for carrying AI memory across applications, devices, and vendors — lossless, local, and with **verifiable deletion** as its guarantee of integrity. We present the vision and principles; an overview of the format (a self-contained .mem bundle with a canonical, byte-reproducible serialization, portable deletion tombstones carrying proof-of-reach, and optional Ed25519 signing); and two open reference implementations (in Python and Swift) with proven byte-for-byte interoperability through a shared conformance fixture and a golden-vector suite, ingest adapters for mem0, ChatGPT, and Claude memory, a production integration, and a four-level conformance program gated on deletion propagation. We state plainly what this proposal is not yet — independently implemented, neutrally governed, widely adopted – and close with an explicit invitation: implement it, adopt it, challenge it, and help govern it.

Keywords: AI memory, data portability, open standard, interoperability, verifiable deletion, GDPR, local-first, canonical JSON.

1. Introduction

AI assistants and chatbots that once answered isolated questions now accumulate a durable picture of their users: what they are working on, who they work with, how they like things done, what they asked for last month, or what they never finished. The *memory* functionality, powering these user experiences, was introduced to OpenAI's ChatGPT [1] in February 2024, followed by Google's Gemini [2] in February 2025, and by Anthropic's Claude [3] in September 2025. This memory is quickly becoming the most valuable and the most personal data in computing, and also the strongest lock-in mechanism ever built into software: switching between assistants from different vendors now means abandoning years of accumulated context.

As of writing this, the industry stands where personal computing stood before open document formats, where email stood before IMAP, where calendars stood before iCalendar (*.ics). Every prior generation of personal data eventually earned an open, portable representation, not because its custodians wanted one, but because users, builders, and eventually regulators demanded one. AI memory deserves the same, and sooner: the data is more intimate and grows over time, the cross-system transition costs are higher, and the question of personal data deletion is harder. Three failures define the status quo:

content immobility the memory cannot be moved between products or devices; a user who switches starts from zero

no shared representation one vendor's memory is free text, another's is a knowledge graph, a third's is a chat log; nothing interoperates, so even a willing vendor cannot import a competitor's export without loss and quality guarantees.

unprovable withdrawal when a user asks the assistant to forget something, the content may persist in a vector index, an embedding cache, a graph edge, or a backup replica, making "deleted" a promise, rather than a verifiable system property.

Our position is simple: **memory belongs to the user, not the technology vendor**. It should live in a format the user can read, move, verify, and erase – and "erase" should be a provable system property.

Herein, we propose such a memory format, describe a complete working implementation of it, and invite the community to take it further than one company can. Our standard implementation, including the specification, schemas,

conformance kit, and both SDKs, is available on GitHub:

Python https://github.com/MacPaw/portable-memory

Swift https://github.com/MacPaw/portable-memory-swift

We invite community contributions and discussion via email at tech-research@macpaw.com or by opening an issue on either repository.

2. Portable Memory Design Principles

We outline below the six design principles that shape the Portable Memory proposal, each being a hard implementation constraint:

User ownership. A memory export is a self-contained set of plain files the user holds – inspectable in a text editor, verifiable with standard tools, and transferable by any means. No server or account is required to read, verify, save, or move it.

Local-first. The format works entirely offline. Cross-device and cross-account synchronization and replication are host concerns, layered above the format, never required by it.

Lossless superset. Interchange formats die when adoption costs data. A Portable Memory bundle preserves what a receiving engine does not model: unknown fields ride along verbatim, and entire unknown record kinds round-trip byte-for-byte. Moving into the format and back out drops nothing.

Engine-agnostic. Source text is the portable truth. Embeddings are model-specific accelerators; receivers rederive them. A bundle is never tied to one model, one index, or one retrieval architecture.

Verifiable deletion. A deletion is a first-class, portable record operation that must propagate to every artifact derived from the content – and conformance certification is gated on demonstrating exactly that (Section 4). This is the hardest guarantee a memory layer can make, which is why it is the one worth certifying.

Open evolution. The specification, schemas, reference implementations, and conformance materials are open (MIT license), and the standard is offered as a proposal to be governed openly – not as a finished decision imposed on the community (Section 6).

3. The Standard Outline

This section provides a high-level overview of the core Portable Memory components. The full specification of the implementation, its JSON schemas for every record kind, and conformance materials are provided with the reference repositories [4].

A memory export is a .mem bundle, a directory of newline-delimited JSON streams plus a manifest and checksums (Listing 1).

Listing 1: A Portable Memory bundle.

```
mybundle.mem/
  manifest.json     version, capabilities,
                    counts, per-file integrity
  manifest.sig      OPTIONAL Ed25519 signature
  items/            episode.jsonl
                    entity.jsonl
                    edge.jsonl
                    fact.jsonl ...
  audit/
    log.jsonl       portable mutation trail
    tombstones.jsonl deletions, applied FIRST
  CHECKSUMS         sha256 per file
```

3.1 Data model

The memory decomposes into 15 default types of records, including:

- timestamped *episodes* (the atomic knowledge unit),
- semantic *entities*
- bi-temporal *edges* and *facts* (memory episode updates supersede, never overwrite older ones, so any past state is reconstructible),
- resources and chunks for documents,
- always-injected core profile blocks,
- procedures,
- scopes, and
- *secret references* that carry encryption metadata but never the secret material

See Listing 2 for an example record, its byte-exact canonical form, and a deletion tombstone with its proof-of-reach.

Stable record identifiers make record merge and deletion idempotent across systems. Vendors may define custom record types, non-default records and fields are preserved, which makes the memory bundle container a superset rather than the lowest common denominator.

The value of the memory is in inverse proportion to its size: the records worth the most are the few that fit in every prompt. This data model preserves this hierarchy rather than flattening it: always-injected core profile blocks form a compact working set, episodic and semantic records form the retrieval layer, and the lifecycle states mark what has receded into archive. A bundle therefore transports not only a corpus but also a prioritization: a receiving engine knows immediately what to inject, what to index, and what to keep cold, instead of re-learning the user's essentials from scratch.

3.2 Canonical serialization

Interoperability is determined at the byte level: two implementations must emit identical byte streams for the data, so its checksums and signatures are independent of underlying programming language specifics. The specification pins a canonical JSON profile in the spirit of RFC 8785 [5],

Listing 2: The format by example. (1) A record whose vendorScore field — and the entire vendorThing kind — is unknown to the format yet preserved verbatim: the lossless superset. (2) A canonicalization vector; the same value serializes to identical bytes, so checksums and signatures hold across implementations. (3) A deletion tombstone whose derived block is the proof-of-reach: the artifacts removed together with the target. Native records are shown in canonical (sorted-key) order and wrapped to fit; foreign kinds keep their original bytes. Excerpted from the conformance fixture

```
// (1) A memory record: one JSON object per line (JSONL); source is the portable truth
items/episode.jsonl
{"confidence":0.7,"id":"ep_0002","sourceType":"note",
 "summary":"The Reykjavik lab is led by Dr. Sabine Holt.","vendorScore":0.92}
// vendorScore: a field the format does not model -> preserved verbatim
items/vendorThing.jsonl
{"id":"vt_1","blob":"opaque"}   // an entire unknown record KIND -> round-trips byte-for-byte

// (2) Canonical JSON is byte-reproducible: 1 of 26 tested vectors
{"a":1.0}  ->  {"a":1}  ->  sha256 015abd7f...a97f862

// (3) A deletion tombstone, applied BEFORE additions so nothing can resurrect
audit/tombstones.jsonl
{"actor":"user","deletedAt":"2024-03-09T16:00:00Z",
 "derived":{  // proof-of-reach: every derived artifact removed
   "chunkIDs":[],"edgeIDs":[],"embeddingCacheKeys":["abc"],"entityIDs":[],
   "episodeLinkCount":0,"factIDs":["fact_x"],"sentenceIDs":[]},
 "id":"tomb_0001","op":"delete","reason":"user erasure",
 "targetID":"ep_legacy","targetKind":"episode"}
```

encoded in UTF-8, with recursively sorted keys, no insignificant whitespace, raw non-ASCII with RFC 8259 escaping [6], shortest-round-trip numbers, and whole-second UTC RFC 3339 timestamps [7]. Crucially, it also ships a **golden-vector suite**, a file containing triplets *input* → *exact canonical bytes* → *SHA-256* that any implementation can test the serialization against in minutes. The vectors are the conformance oracle, allowing to avoid fine-grained discrepancies in the data, e.g., float number value serialization as 1.0 versus 1.

3.3 Record deletion with a proof-of-reach

A deletion is a portable *tombstone*: a monotonic record naming the target and carrying a *proof-of-reach* – the identifiers of every derived artifact removed (facts, edges, chunks, embedding-cache keys). Importers apply tombstones **before** additions, so a stale bundle can never resurrect deleted content, for any record kind. A redact variant purges content while keeping the deletion provable, a GDPR Article 17 [8] made mechanical. An exportable *Evidence Pack* (audit trail + tombstones + provenance) turns "governed memory" into an artifact a compliance team can file.

3.4 Authenticity

Checksums give integrity; optional Ed25519 signatures [9] over the manifest give authenticity for the whole bundle, and tombstones may be individually signed. Verification is fail-closed against caller-trusted keys. Signatures are verify-interoperable across implementations (signature bytes may differ, validity does not).

3.5 Conformance levels

Conformance is layered (see Table 1) so standard adopters can advance gradually. For instance, L0 support means that only the memory export operation is supported. The conformance badge requires at least the L2 layer implemented.

L	Requirement
L0	Export: a valid, checksum-clean bundle.
L1	Import/Merge: lossless, idempotent, bi-temporal.
L2	**Deletion propagation – the badge.**
L3	Governed: audit, Evidence Pack, signed tombstones.

Table 1: Conformance levels. Certification is gated on L2 because of deletion propagation is the guarantee that users and regulators actually test.

4. Base Implementation

Our Portable Memory implementation is public [4], open source (under the MIT license), and has been tested in continuous integration on macOS and Linux.

Two reference implementations, byte-interoperable. A Swift SDK (Apple platforms and Linux) and a Python SDK (pure standard library; signing as an optional extra) implement the full format. Their interoperability is demonstrated in three ways: both reproduce a shared conformance fixture byte-for-byte, both independently reproduce all 26 golden canonical vectors, and a bundle exported by one (including a cryptographically signed one) can be validated, imported, and re-exported *byte-identically* by the other.

Deletion propagation, enforced and tested. Tombstone-first import with no-resurrection is enforced for every record kind and is covered by tests in both SDKs, including an adversarial case where a stale bundle still carries deleted rows.

Migration pathways from existing ecosystems. Ingest adapters map exports from other systems onto the Portable Memory model, supporting out of the box

the mem0 architecture (verified field-by-field against its documented export shapes), the native ChatGPT data export (conversations.json), and Claude memory files (MEMORY.md and topic files). In case there is no suitable record to match the exported data, it will still be preserved as metadata, not to be lost. Standard adoption never costs data.

Open project scaffolding. Language-neutral JSON Schemas for every record kind, a conformance kit with fixtures and the golden vectors, security policy, a contribution guide, and a written governance document.

Production integration. As of now, the Portable Memory format is exercised end-to-end (export, import, and live deletion propagation across real retrieval routes) in Mnemos, a memory system built at MacPaw, demonstrating that the seam between the portable format and a host engine holds in practice.

5. Relation to Existing Work

Memory engines. Several runtimes that store and retrieve agent memory are widely adopted: mem0 [10], Letta/MemGPT [11], Zep [12], OpenMemory. They do not use interchangeable formats, and their exports are engine-specific. Portable Memory can be the layer between them: the format they can export to and import from without loss, as the shipped mem0 adapter demonstrates. We see engines as the standard's most natural early adopters, not competitors.

MCP. Model context Protocol [13] is a runtime protocol that connects models to tools and context at call time. Portable Memory is the durable data format for the memory itself. They can be composed: an MCP-connected assistant can export or import its memory as a bundle.

Data-portability regimes. GDPR Articles 17 and 20 [8], the EU AI Act [14], and efforts like the Data Transfer Project [15] establish rights to move and erase personal data. Portable Memory is a specific, machine-verifiable tool for exercising both rights for AI memory, where deletion propagation turns an erasure obligation from a policy claim into a testable property.

Memory portability. First-party memory transfer is beginning to ship: Claude imports memory generated by ChatGPT, Gemini, and Copilot [16]. These features confirm the demand but meet it as point solutions, currently prompt-mediated, best-effort transfers with no shared schema, no integrity guarantee, and no way to carry or verify a deletion. An open format turns such one-off bridges into lossless, auditable interchange. A recent work by Ravindran [17] proposes a *Portable Agent Memory* (PAM) format for provenance-verified memory transfer between agents with strong content-addressing and capability-token mechanisms and a recoverable redaction model. Portable Memory (this proposal) was developed independently of PAM, growing out of MacPaw's Mnemos memory-system architecture. Despite the similar names, the two systems embody distinct designs: PAM targets cryptographically verified transfer of memory snapshots between agents, whereas our work specifies a lossless interchange container for a living memory store, with irreversible, propagating deletion as its conformance gate. We read the parallel work as validation that this problem is real and urgent. Our proposal differs in emphasis: a lossless superset container for a *living store* rather than a transfer snapshot, byte-level cross-implementation reproducibility as a tested property, and *irreversible*, propagating deletion as the certification gate. We would welcome convergence and see complementary ideas worth adopting. A concurrent IETF Internet-Draft, *Memory Interchange Bundle Format for AI Agents* (draft-vu-aimem-bundle-00) [18], likewise specifies a vendor-neutral JSON bundle for agent memory, with SHA-256 content hashing, COSE-signed bundles, producer/consumer conformance levels, and GDPR Article 17 erasure semantics. Independent arrival at the same pillars (a JSON container, layered conformance, first-class erasure) reads to us as further evidence that this layer is overdue. Portable Memory differs in its lossless-superset semantics (unknown record kinds round-trip byte-for-byte), its byte-level cross-implementation golden vectors, and its tombstone-first import order, which certifies deletion *propagation* rather than erasure markers alone. The topic is also reaching formal standards venues: a W3C community group on AI-agent memory interoperability has been proposed [19].

6. From Proposal to Standard: Governance and Roadmap

We are deliberate about what this document is today: a **proposal**, authored and stewarded by one company, with two reference implementations that share an author. A format becomes a standard only when it stops being any one vendor's asset. Our commitments, written into the project's governance document:

Open change control now. All specification changes go through public RFC-style issues; format versions follow semantic versioning; unknown fields and kinds always round-trip, so minor revisions never break older readers.

Shared maintenance as adoption grows. Contributors from outside MacPaw, with sustained involvement, will be added as maintainers; format-affecting changes land in every reference implementation before a version is stamped.

A neutral home when it is earned. Once independent implementations and adopters exist, we intend to move the specification to a vendor-neutral body, a foundation or a working group, where MacPaw is one voice among several. We consider this the success condition, not a concession.

A conformance program with teeth. Self-serve conformance testing (validator, fixtures, golden vectors) exists today. The next milestones are a public registry of implementations and adopters, and a certification badge gated on the L2 deletion probe. A badge that means something is worth more than a large logo wall.

7. Call for Collaboration

A portability standard is worth exactly as much as the ecosystem that speaks it. We are calling for collaborators, welcoming contributions on different levels and from different perspectives of the system development:

Implement it in your language. TypeScript, Go, Rust, or Kotlin — the specification is language-neutral, the schemas are machine-readable, and the golden vectors make byte-compatibility a test you can pass in an afternoon rather than a negotiation. An independent implementation is the single most valuable contribution anyone can make; it is what turns a "MacPaw's format" into a standard.

Build an adapter. If you maintain or use a memory engine, assistant, or agent framework, map its export onto the model: its three shipped adapters average a few hundred lines. Every adapter makes every other adapter more valuable.

Adopt it in a product. Offer your users a real export: a bundle they can hold, inspect, and take elsewhere. We will help, including co-building the integration.

Review the specification. File issues where the specification is ambiguous, where the canonicalization has an edge we missed, or where the deletion model can be defeated. Two independent audits and an adversarial review shaped v1.0; the standard will be hardened by scrutiny, not endorsement.

Pilot the compliance story. If you operate under GDPR or the EU AI Act and need demonstrable erasure, your feedback and participation will be much appreciated and valuable.

8. Limitations and Open Questions

There are several concern about the standard proposal and its adoption path we would like to disclose below. *Adoption is unproven:* today, there are two reference implementations with one author and one production integration; the network effect this paper argues for does not yet exist. *Governance is not yet neutral:* the commitments in Section 6 are written but not yet tested by a real disagreement. *Scope cuts in v1.0:* inline embedding payloads, a single-file archive form, and graph-relation mapping in the adapters are deliberately deferred; byte-identity is guaranteed within a tested domain (documented residuals include non-BMP object keys and integers beyond 64 bits). *Open design questions* we would value community input on: merge tie-breaking for same-key concurrent edits, a capability/access-control layer for scoped, least-privilege slices of one memory serving multiple specialized agents (concurrent work [17] has good ideas here), content-addressed provenance, and what an L2 certification harness should look like for hosts with exotic retrieval stacks.

9. Conclusion

AI memory became the most personal dataset in computing, and it is accumulating inside walled gardens by default. We propose treating it the way the web eventually treated every other class of personal data: as user-owned information with an open, portable, verifiable representation. Portable Memory is our specific offer: a specification with running, byte-interoperable proof, deletion you can demonstrate rather than promise, and governance designed to outgrow its author. Whether it becomes *the* standard matters less to us than that an open standard exists at all. Help us make one.

References

[1] OpenAI. Memory and new controls for ChatGPT, February 2024. *Online.* https://openai.com/index/memory-and-new-controls-for-chatgpt/.

[2] Dave Citron. Reference past chats for more tailored help with Gemini advanced, February 2025. *Online.* https://blog.google/feed/gemini-referencing-past-chats/.

[3] Anthropic. Bringing memory to Claude, September 2025. *Online.* https://claude.com/blog/memory.

[4] MacPaw. Portable memory – specification, schemas, conformance kit, and reference SDKs. *Online.* https://github.com/MacPaw/portable-memory · https://github.com/MacPaw/portable-memory-swift.

[5] A. Rundgren et al. JSON canonicalization scheme (JCS). RFC 8785, RFC Editor, 2020.

[6] T. Bray. The JavaScript object notation (JSON) data interchange format. RFC 8259, RFC Editor, 2017.

[7] G. Klyne and C. Newman. Date and time on the internet: Timestamps. RFC 3339, RFC Editor, 2002.

[8] European Union. Regulation (EU) 2016/679 (GDPR), arts. 17, 20, 2016.

[9] S. Josefsson and I. Liusvaara. Edwards-curve digital signature algorithm (EdDSA). RFC 8032, RFC Editor, 2017.

[10] mem0ai. mem0: The memory layer for AI agents. *Online.* https://github.com/mem0ai/mem0.

[11] C. Packer et al. MemGPT: Towards LLMs as operating systems, 2023. Letta.

[12] P. Rasmussen et al. Zep: A temporal knowledge graph architecture for agent memory, 2025.

[13] Anthropic. Model context protocol, 2024. *Online.* https://modelcontextprotocol.io.

[14] European Union. Regulation (EU) 2024/1689 (artificial intelligence act), 2024.

[15] Data Transfer Initiative. Data transfer project. *Online.* https://dtinit.org/. Accessed 2026-07-04.

[16] Anthropic. Claude memory import (ChatGPT, Gemini, Copilot). *Online.* https://claude.com/import-memory. Accessed 2026-07-06.

[17] S. K. Ravindran. Portable agent memory: A protocol for cryptographically-verified memory transfer across heterogeneous AI agents, 2026.

[18] V. D. Minh. Memory interchange bundle format for AI agents, June 2026. IETF Internet-Draft draft-vu-aimem-bundle-00. *Online.* https://datatracker.ietf.org/doc/draft-vu-aimem-bundle/.

[19] W3C. Proposed community group: AI agent memory interoperability. *Online.* https://www.w3.org/community/blog/2026/05/18/proposed-group-ai-agent-memory-interoperability-community-group-community-group/. Accessed 2026-07-06.