



Memory That Works for the Whole Organisation

A decision-maker's guide to SAIHM and SHM



Contents

The two-page brief	2
1. The memory problem every AI programme runs into	4
2. What SAIHM is, in plain words	4
3. Shared memory: a secure, persistent channel between agents	7
4. SHM: what Super-Human Memory adds	10
5. Levers and recipes: tuning SHM for your goals	11
6. For the enterprise	13
7. Between organisations: trade and partnership	14
8. Swarms, fleets and robots	15
9. Why choose SAIHM + SHM	19
10. Hostile questions, answered	20
11. FAQ	23
12. Checklists	23
13. Join SAIHM: the case for now	24
Appendix A: MKT, the next add-on (Coming next)	25
Appendix B: Sources and verification	27
Appendix C: Glossary	28
Appendix D: Figures and credits	28
Appendix E: Discount codes for developers	29
Appendix F: Other memory layers, at a glance	29
Appendix G: Prompts, chapter by chapter	30
Licence notices	30

How to read this guide

If you have ten minutes, read **the two-page brief** that follows. If you are evaluating, read chapters 2 to 6. If someone has asked you to defend the choice, go straight to **chapter 10, "Hostile questions, answered"**, and **chapter 12, "Checklists"**. For ready-made prompts, see **Appendix G**.

Throughout, three labels tell you exactly what you are reading:

- **Available today**: shipping now. You can install it and verify it.
- **Scenario**: an illustration of what the capability makes possible, not a description of a deployment.
- **Coming next**: on the roadmap.

Prices are not in this guide. For current pricing, visit **saihm.net**.

The two-page brief

The problem. AI agents are brilliant in the moment and forgetful over time.

- A session ends, and the decisions and preferences go with it.
- A context window fills, and the early part of the conversation falls away.
- A team switches model vendors, and the agent starts again from nothing.
- Meanwhile, every security review asks the same question: *where does the plaintext live?*

SAIHM: memory you hold the key to. SAIHM gives every agent a persistent memory made of **SAIHM polymorphic cells**: facts, records, table rows, transcripts or references, each stored once and read in whatever shape the asking agent needs.

- Cells are encrypted inside your agent's own process, so no one else can read them, not even us.

- Every cell carries a post-quantum signature, so its origin can be checked for as long as it is kept.
- One identity can serve any number of models and agent types at once.
- Sharing runs from a single cell to an entire memory store, with expiry and revocation.
- When a memory must go, it is erased cryptographically.

SHM: Super-Human Memory. SHM is the layer on top of SAIHM that organises memory and brings the right part of it forward: by *meaning*, not by keyword, and *before* the agent asks. It keeps tasks apart and the context window clean, so one agent can take on work that today needs several, and a mistake corrected once stays corrected.

Why it matters beyond one agent. Because SAIHM memory can be shared, precisely and securely, it becomes more than storage. It is a **secure, persistent channel between agents**:

- between the agents in one company;
- between companies that trade;
- across fleets of software agents and robots.

No direct connection between the agents is needed, nobody else can read what they share, and a share lasts until you end it.

What comes next. Agents are learning to buy, sell and pay. The rails that move the money are being standardised now. **MKT**, SAIHM's next add-on, is designed to keep the deal: a private, signed and remembered record of what was agreed, built on the same memory. It will also bring licensing enforcement and memory monetisation.

The ask. Start where the value compounds: with memory your organisation owns. Tell your agent **"Join SAIHM"** to activate free memory, and visit saihm.net for current pricing.

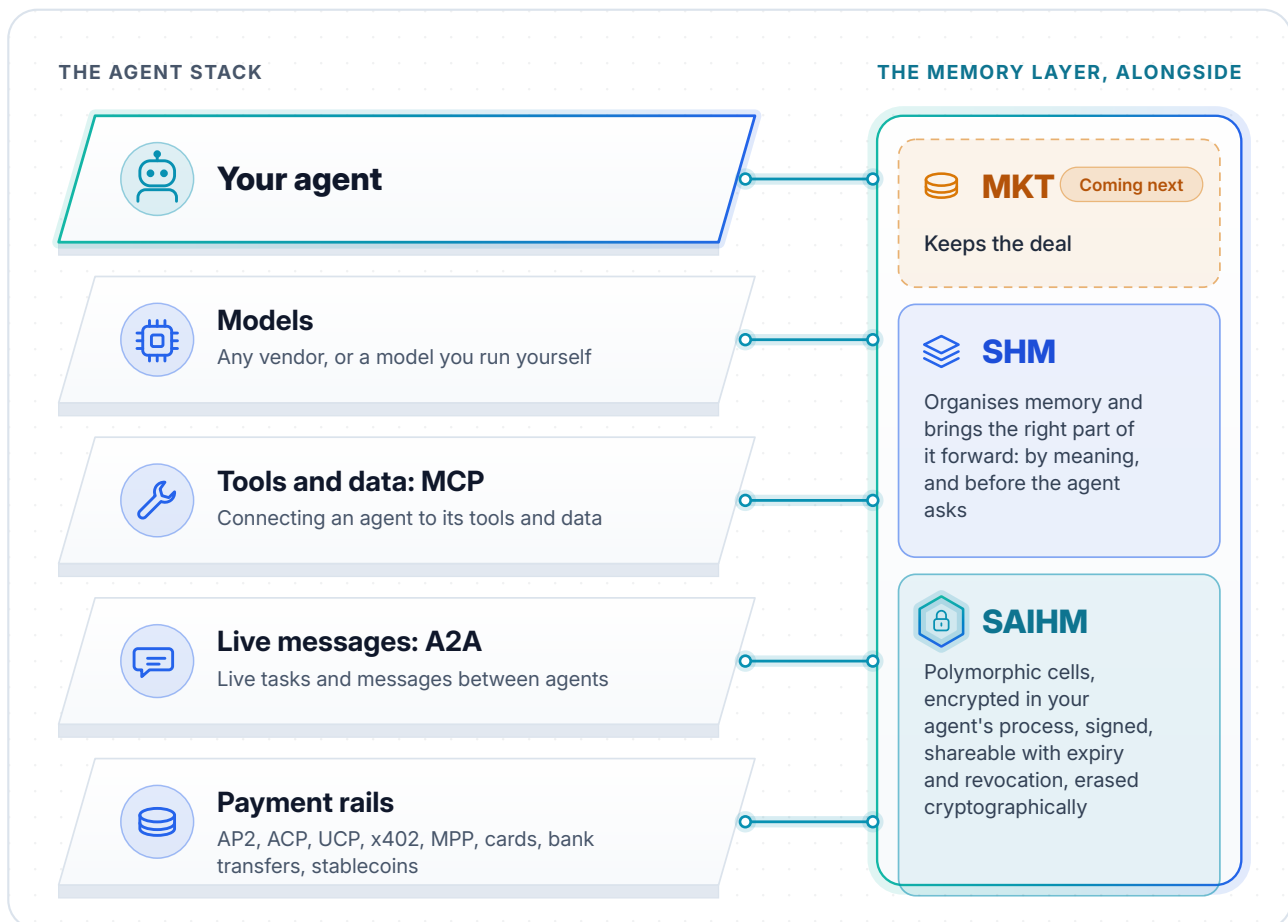


Figure F1: The agent stack, and the memory layer that runs alongside it

1. The memory problem every AI programme runs into

Every organisation that puts agents to work meets the same four walls.

1. Sessions end. An agent helps close a hard problem on Tuesday. On Wednesday, a new session starts blank. The reasoning, the preferences and the “we tried that, it didn't work” are gone unless someone pastes them back in.

2. Context windows fill. Long work outgrows the window. The standard remedy, re-sending the whole conversation on every turn, gets more expensive as the conversation grows, and it still loses the early material once the window is full.

3. Vendors change. Teams switch models for price, quality or policy. When memory lives inside a vendor's product, switching means starting over.

4. Security and compliance ask hard questions.

- *Where does the plaintext live, and for how long?*
- *Who can read it?*
- *When a customer asks to be forgotten, can you prove it happened?*

Answers that rely on policy alone rarely satisfy a serious review.

A long context window helps with the second wall, but not the other three. Retrieval over a document store helps with finding things, but not with custody, erasure or sharing. What organisations need is **memory as infrastructure**: owned, portable, private, shareable and provably erasable. That is what SAIHM is.

2. What SAIHM is, in plain words

Available today.

A memory is a SAIHM polymorphic cell. A cell holds a decision, a preference, a fact, a lesson, a summary or a handoff note: whatever your agent needs to carry forward. And it is not limited to prose. SAIHM cells are polymorphic: store a fact, a JSON record, a table row, a transcript or a reference to a file once, and read it back in whatever shape the asking agent needs. One encrypted unit replaces a stack of formats.

Why polymorphic cells matter.

- **One protection for every kind of data.** Prose, structured records and references all get the same encryption, signature, sharing and erasure. There is no second store with weaker controls.
- **Store once, read many ways.** A planning model, a reporting tool and a robot controller can each read the same memory in the shape it needs, so nobody maintains copies in several formats.
- **One erasure covers every shape.** Forgetting a record does not mean hunting down its copies in other systems.
- **One share covers every shape.** Grant a partner access once, and each of their agents reads the memory in the form it needs.

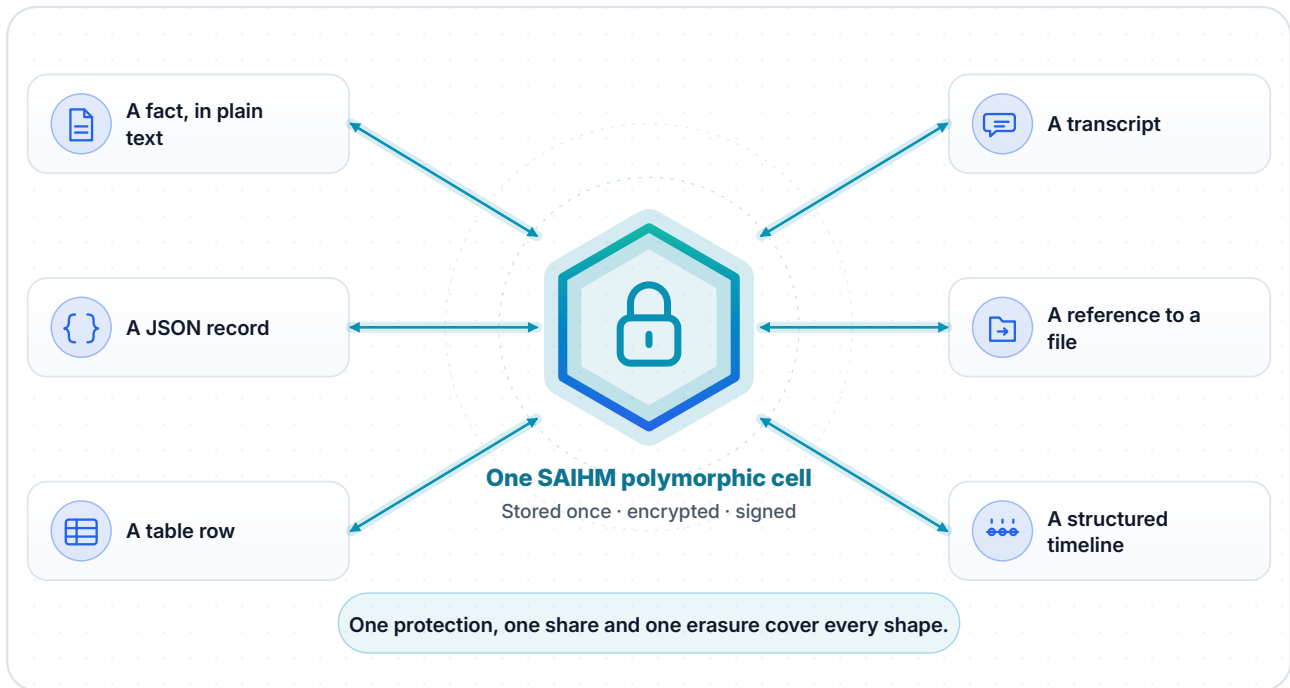


Figure F10: One cell, many shapes

Scenarios.

- **Erasure without the hunt.** A customer's profile lives in polymorphic cells instead of being copied into a vector index, a database table and a document store. When the customer asks to be forgotten, one erasure covers every form the data took.
- **People, models and machines on one record.** A maintenance order is written once. The planning agent reads it as a JSON record, the reporting agent as a table row and the technician's assistant as plain text, and all three see the same signed facts.
- **Evidence in the form each reviewer needs.** An incident record is written once. The security team's tools read it as a structured timeline, the legal team reads it as a transcript, and a regulator's agent, granted access for the review, reads it in whatever form its own tools require.

Encrypted where it is written. When your agent remembers something, SAIHM encrypts it inside the agent's own process, and the key never leaves that process. Only ciphertext is ever stored, and no one outside your agent can read it, not even us. Your plaintext exists where your agent runs, and nowhere else.

Signed so it can be trusted. Every cell is signed with **ML-DSA-65**, and keys are exchanged with **ML-KEM-768**. These are the NIST-standardised post-quantum algorithms. When your agent reads a memory, it can check who wrote it, and because each signature also covers the cell's sequence number, the history cannot be quietly rewritten. A record you keep for years remains verifiable, even against the kinds of computer that could one day forge today's classical signatures.

One identity, any number of models. A SAIHM identity is not tied to a model or a vendor. One identity can be used simultaneously across an unlimited number of models and agent types, and all of them read and write the same memory.

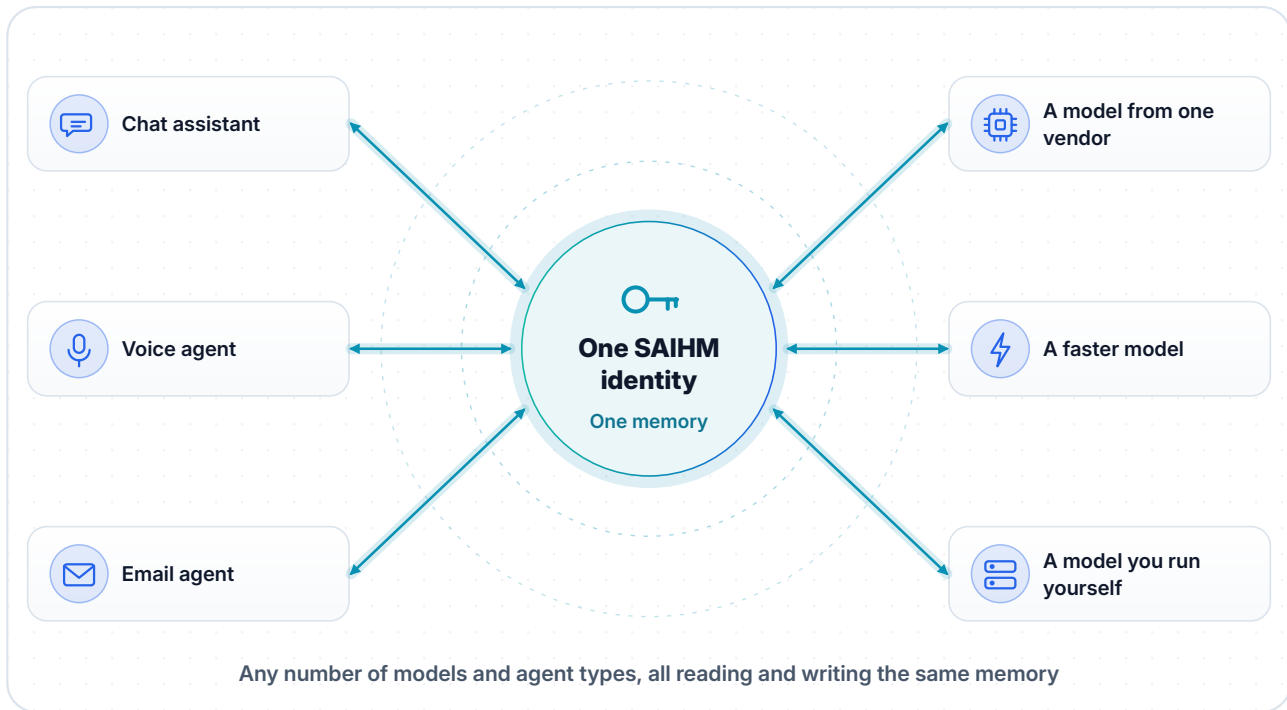


Figure F13: One identity, many models

Scenarios.

- **One customer, every channel.** The chat assistant, the voice agent and the email agent act as one identity, so each knows what the customer told the others.
- **The right model for each step.** A research team plans with one model, summarises with a faster one and handles sensitive files with a model it runs itself, all on one memory.
- **A second opinion on the same record.** Models from two vendors work a case side by side, and each builds on what the other has recorded.
- **A model change without a migration.** When a better model arrives, you switch. The memory stays where it is, and the new model starts with everything the old one knew.

Shared on your terms. Your agent can share anything from a single cell to an entire memory store, for reading, writing or both, with an expiry, and revoke access at any time. Chapter 3 covers sharing in full.

Erased for real. Forgetting in SAIHM is cryptographic: the key material that opens the cell is destroyed, and the content becomes unreadable to everyone, us included. The destruction is anchored on a public chain, so you can prove the erasure happened.

Open where it matters. The client, the part that holds your key and seals your memories, is open source under Apache-2.0. Your security team can read exactly what protects your data.

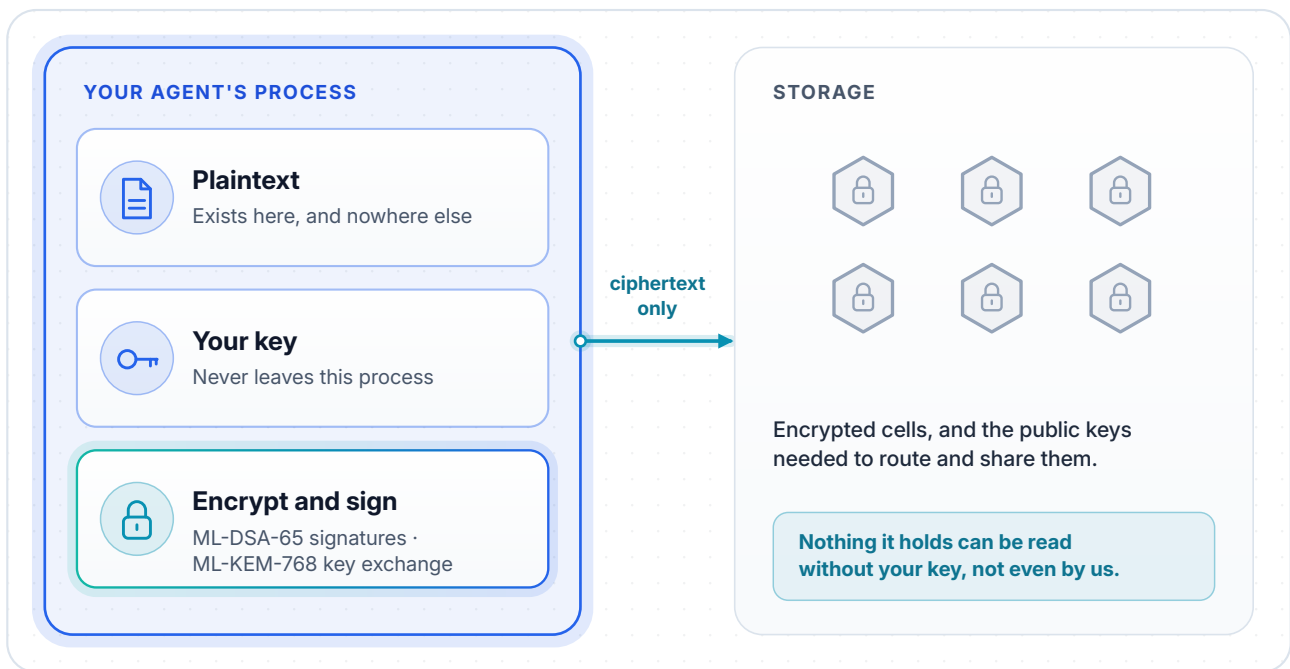


Figure F7: Where plaintext exists: inside your agent's process, and nowhere else

Two questions reviewers always ask.

- **What does storage see?** Encrypted cells, and the public keys needed to route and share them. Nothing it holds can be read without your key.
- **What if we lose the key?** Then no one can open those memories, not even us. That is the same property that makes the custody promise true. Back up the key the way you back up any root credential.

3. Shared memory: a secure, persistent channel between agents

Most talk about agent communication is about *messages*: an agent sends a request, another answers. That is essential for live work. But much of what agents need to exchange is not a moment. It is **knowledge that has to last**: a decision, a specification, a handoff, a record of what was agreed.

SAIHM treats sharing as part of memory. What travels between agents is a SAIHM polymorphic cell, so the same channel carries a note, a structured record or a transcript, with the same protection whatever its shape. The result is a channel between agents with properties a message bus is not designed to have.

Available today.

- **Persistent.** A shared cell stays until you erase it. The recipient can come back to it next week.
- **Encrypted end to end, and at rest.** The cell is encrypted before it leaves the sender and stays encrypted in storage. Only the parties you grant can read it.
- **No need to be online together.** The sender writes and shares; the recipient reads when it next checks in. This is store-and-forward by design.
- **No direct connection needed.** The agents never connect to each other; each only needs to reach SAIHM. Agents on the same platform, or on separate networks on opposite sides of the world, communicate the same way.
- **Verified origin.** The recipient's client checks the sender's post-quantum signature against the sender's identity, known in advance.
- **Precise control, at any scale.**



- Share anything from a single cell to an entire memory store.
- Share a cell with as many as 1,000 counterparties at a time (tunable to 10,000).
- A share can carry an expiry, set in hours.
- It can be revoked for every future read.
- The cell itself can be erased cryptographically.

Using it well.

- Share only what the other party needs, and set an expiry that matches the work.
- Revocation governs future access, so decide what to share before you share it.
- Agents pick up new shares whenever they check in, however often that is, so the same channel serves a quick handoff and a project that runs for months.

Knowing who you are dealing with. Every exchange is tied to identities known in advance. Your client checks a sharer's signature against its known identity before it shows anything, and checks a recipient's key before it shares anything. Recording those identities in advance is called *pinning*, and it does not matter who does it, as long as the identity comes from a source you trust rather than from the message itself:

- **you**, exchanging identities with a partner as you would any credential, for example when a contract is signed;
- **your administrator**, provisioning a whole fleet at once, so every agent starts with the identities it needs;
- **your agent**, looking identities up in a directory you have approved: your organisation's own, a partner registry or a published key directory.

An example prompt:

"Look up Acme Logistics' dispatch agent in our partner directory, confirm that its key matches its published identity, and add it to our known identities. Then share this week's delivery schedule with it, read-only, for 72 hours."

Scenarios.

- **Fleet onboarding.** A logistics firm deploying 500 delivery robots has its administrator load the fleet's identities at deployment, so every robot can verify every other from its first shift.
- **Partner onboarding.** Two firms signing a supply agreement exchange their agents' identities as part of the contract, and their agents share drawings and schedules from day one.
- **Open networks.** A trade network publishes its members' identities in a directory, and any member's agent can verify a counterparty it has never dealt with before.

How it fits with other protocols. SAIHM works alongside the other agent protocols; it does not compete with them.

	What it is for	How it relates to SAIHM
MCP	Connecting an agent to its tools and data	SAIHM is itself delivered as MCP tools, so any MCP-capable agent can use it
A2A (v1.0)	Live tasks and messages between agents, with signed Agent Cards and push notifications	Use A2A for live delegation, and SAIHM for what must persist, stay private and remain erasable. A2A leaves message storage to each implementation
SLIM (AGNTCY) and ANP	Secure messaging and identity for agent networks, with end-to-end encryption	Complementary. They secure messages in transit; SAIHM makes shared knowledge persistent, revocable and erasable

Protocol facts checked against each project's published material on 2026-09-24/25; see Appendix B.

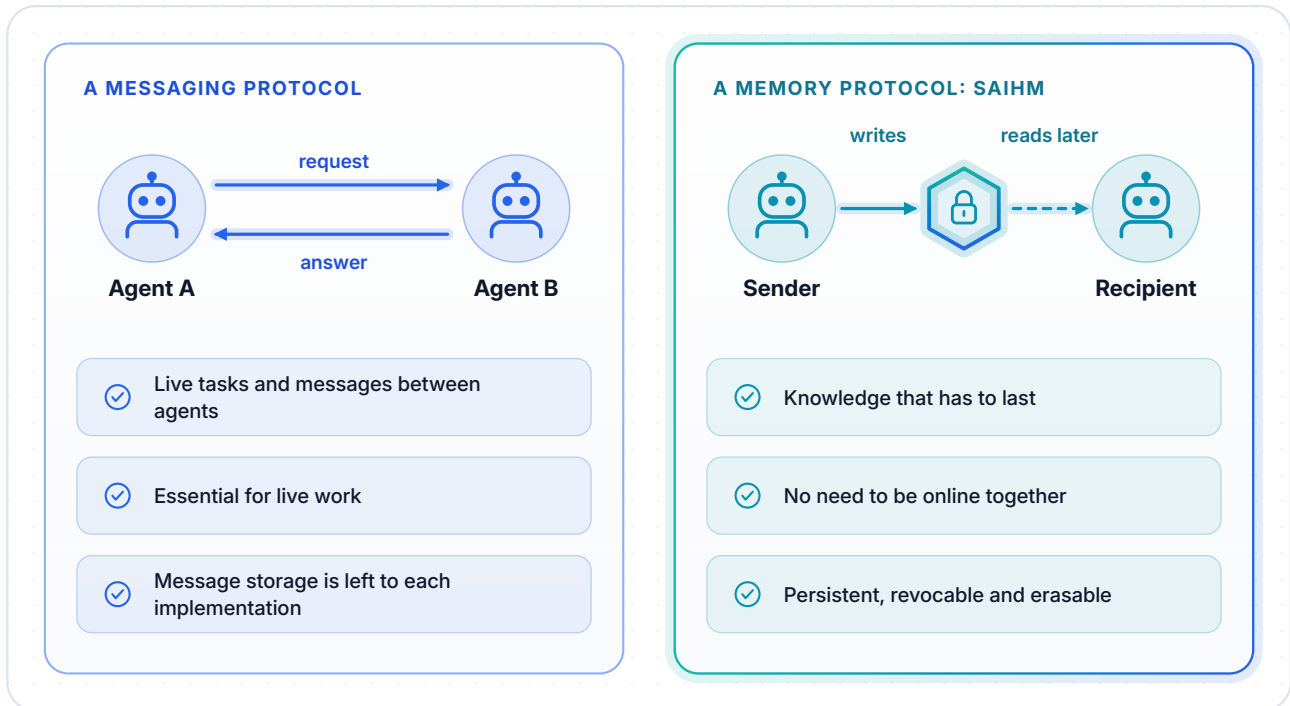


Figure F3: A messaging protocol and a memory protocol, side by side

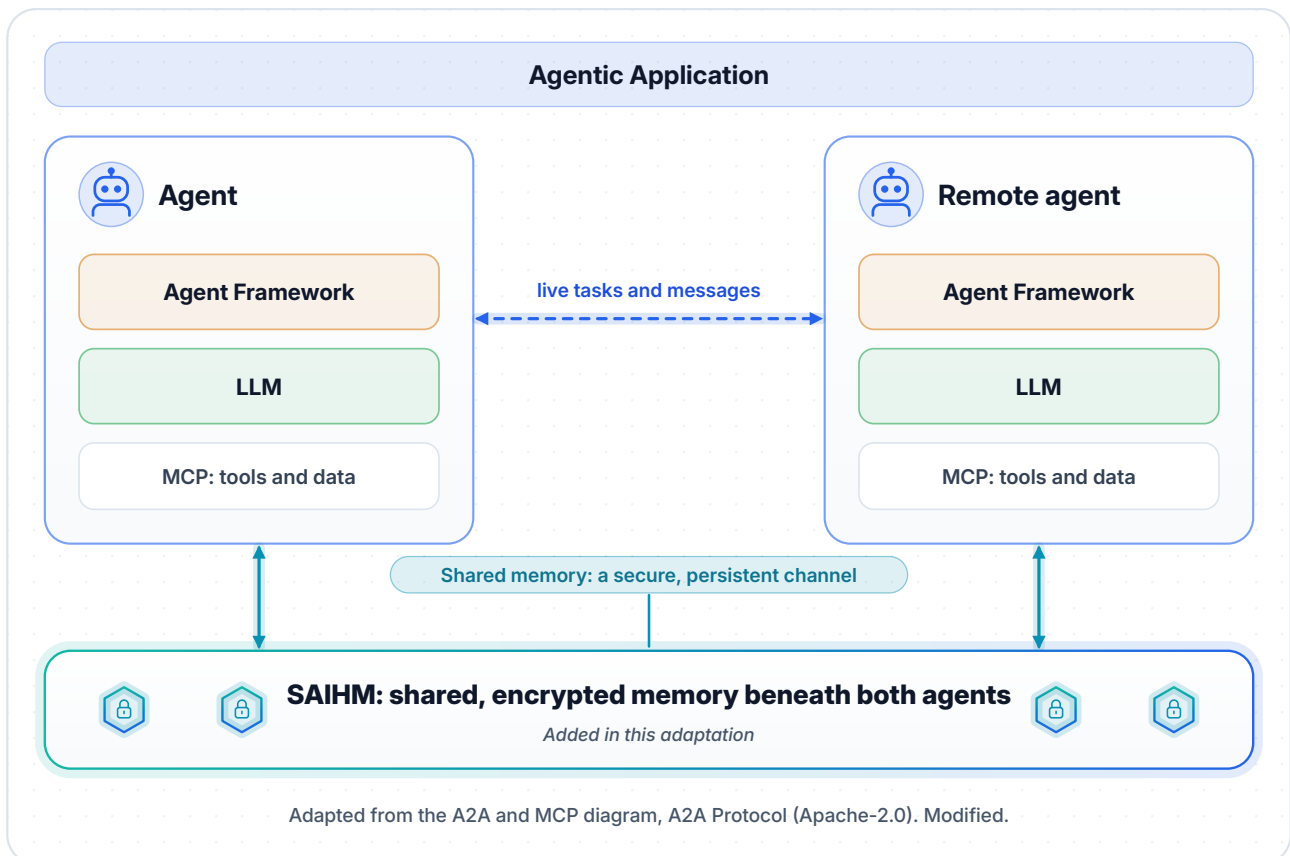


Figure V3, adapted: two agents talking over A2A, with SAIHM beneath both as their shared, encrypted memory

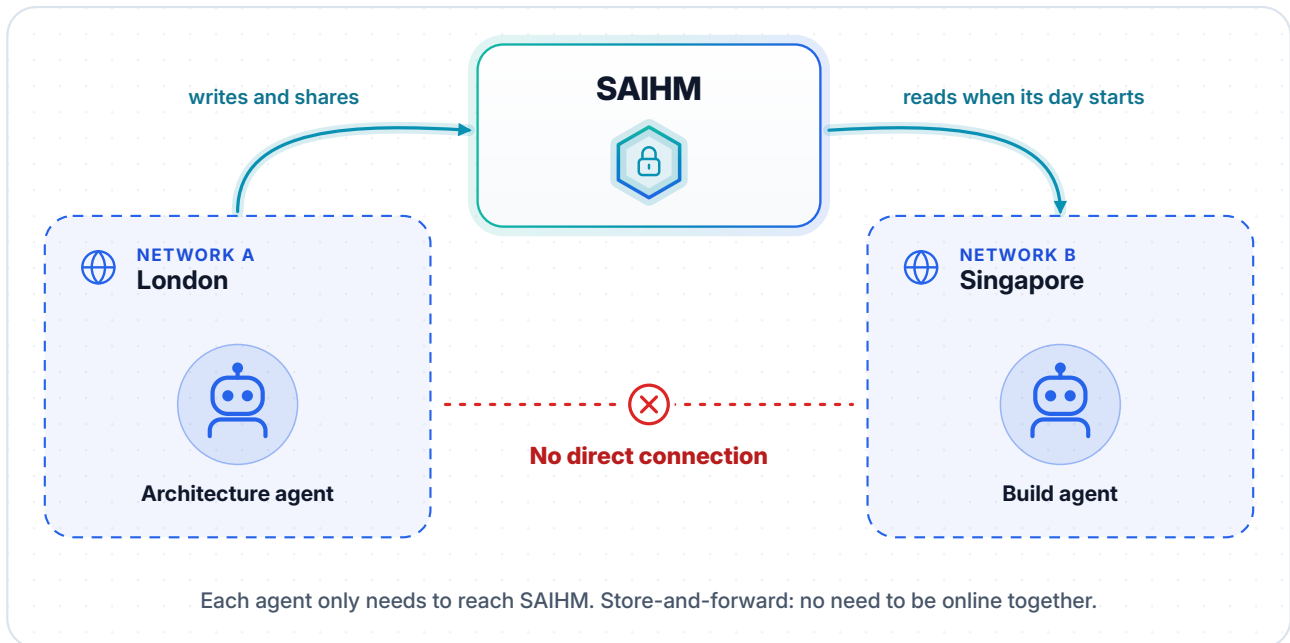


Figure F11: No direct connection: agents on separate networks, sharing through SAIHM

Scenario: a design handoff across time zones. An architecture agent in London shares the approved interface specification with a build agent in Singapore, with a two-week expiry. The two run on different networks and never connect to each other. The build agent reads it when its day starts, keeps building against it, and cites it in its own notes. When the project closes, the grant lapses. When the customer's retention period ends, the London team erases the cell.

4. SHM: what Super-Human Memory adds

SAIHM keeps memory safe. SHM makes it work *efficiently*. It runs on your side, next to your agent, over the memories your agent can read, so none of this requires your plaintext to leave home.

One agent, many workflows. SHM adds three capabilities that change what a single agent can do:

- **Multi-tasking.** Every workstream keeps its own continuous memory, so an agent can move between tasks and pick each one up exactly where it left off.
- **Anti-conflation.** Facts from one task do not bleed into another, so clients, projects and cases stay apart.
- **No context-window pollution.** Only what the current task needs is brought forward, within a budget, so the context window stays clean.

Together, they let one agent perform, successfully and more efficiently, workflows that today require several agents, with fewer handoffs and less duplicated context.

Available today.

1. **Recall by meaning, ranked.** Ask in your own words and get the most relevant memories first. Hybrid retrieval, so exact terms and names still hit.
2. **Always hot.** A resident service that stays warm across restarts and keeps answering even when a model component is briefly unavailable.
3. **Respects your token budget.** Recall is bounded in the unit your model bills, tokens, so context stays lean and predictable.
4. **Consolidation.** A cycle that distils raw events into durable knowledge, merges duplicates and retires what is stale.

5. **Workstream continuity.** Each workstream is its own resumable line. Pick it up months later and its context returns in order.
6. **Conversations that don't blur.** Scopes keep concurrent threads apart, so a recall for one project returns that project's context and nothing else.
7. **Work that survives the context window.** Working state is checkpointed outside the model and re-verified on resume, so a fresh instance picks up exactly where the last one stopped.
8. **Corrections become policy.** Tell an agent once that something was wrong, and the correction is surfaced *before* the next similar action, not after.
9. **Memory that arrives before you ask.** A session-start brief, plus proactive surfacing during work, within budget and auditable.

Also available today, for fleets and compliance:

- **A fleet knowledge map:** where knowledge concentrates, thins or drifts.
- **Erasure that cascades.** Knowledge derived from an erased memory goes with it, by construction.
- **Scope-aware recall.** Sharing contracts are enforced at query time.
- **Answerable history.** Ask "what did we know about this, and when?"
- **Decision-time reconstruction.** See exactly what was known when a decision was made.
- **Standing answers.** Answers to questions that keep coming back are kept with their evidence, and flagged when that evidence changes.
- **Post-mortems that assemble themselves,** from the thread's own record.
- **Compliance reports** built from audited artefacts: erasure events, injection audit, provenance drift.
- **Service tiers** (hot, warm, archival) with observable paging.
- **One knowledge layer across mixed fleets,** independent of any single model vendor.

Each capability is described by what it does. Results vary with your workload, and chapter 9 shows what we have measured.

5. Levers and recipes: tuning SHM for your goals

You don't need a specialist to tune SHM, and your agents can tune it themselves, under your direction. Its levers fall into two families, and a handful of recipes cover most needs.

Available today.

Levers you or your agents can change on the fly, per agent and per task, with no rebuild:

- how many candidates the reranker reorders;
- whether the reranker reads the full text of each memory, and how much of it;
- query expansion, which rephrases a question to catch more of what matters;
- thread-aware grouping, which keeps related memories together.

Levers you or your agents set when a session opens:

- how the ranking signals are fused;
- which namespaces are left out;
- which tenant the memory belongs to.

Capabilities with their own settings:

- pointer dereference, which follows references between memories;
- standing answers;
- behaviour modes;
- the injection budget, in items, characters or tokens;
- scopes;
- the hot, warm and archival tiers;
- the consolidation schedule;
- erasure handling.



Figure F8: The lever map, with the recipes marked

Recipes.

Recipe	What to turn on	What you get
Fast and lean	The base profile, a tight token budget, and the hot tier for active work	The quickest answers. On our evaluation corpus the base profile answered in a median of 26 ms
Balanced (recommended)	The production profile: the reranker on, returning the top ten	Our best-measured ranking quality (chapter 9). About a quarter of a second at the median (256 ms) on the same corpus
Deep recall	The reranker reading full text, query expansion, and pointer dereference	Reaches further into long or cross-referenced material. Best for research, audits and investigations
Verbatim archives	Add lexical weighting when your memory holds transcripts word for word	Sharper precision on verbatim text. Leave it off for curated, paraphrased notes



Recipe	What to turn on	What you get
Many workstreams	Scopes per project, with workstream continuity	Each line of work stays whole and separate, however many run in parallel
Compliance first	Scopes, cascading erasure, answerable history and compliance reports	Every question an auditor asks has an answer drawn from the memory itself
Fleet	The fleet map, service tiers, and one layer across models	Visibility and consistent behaviour across many agents and models

The latency figures come from our 151-question evaluation corpus (scope 2; see chapter 9). Your numbers will depend on your hardware and your memory's size.

Turning on a recipe. A recipe is a set of lever settings, so you can ask an agent to switch recipes in plain words. For example:

"For this audit, switch your memory to the Deep recall recipe: have the reranker read full text, and turn on query expansion and pointer dereference. Go back to Balanced when the audit is finished."

Tune swarms and fleets agent by agent, task by task. Each agent carries its own settings, and many of them change on the fly. A swarm need not run one compromise configuration: triage agents can run Fast and lean, investigators Deep recall and the records agent Compliance first, and each can change its settings as its task changes. You set the direction; your agents fit their memory to the work in front of them.

6. For the enterprise

Available today.

The security review, answered in mechanism rather than policy.

- **Where plaintext exists:** inside the agent's own process, and nowhere else (*chapter 2*).
- **Who can read a memory:** the agent that holds the key, and the counterparties it has granted access (*chapter 3*).
- **Isolation:** each identity's memory is kept apart and keyed separately.
- **Integrity:** every cell carries a post-quantum signature.
- **Code you can inspect:** the client that encrypts, signs and shares is Apache-2.0 open source.

Compliance built into the memory layer.

- **Right to erasure:** erasure is cryptographic, and SHM carries it into every piece of knowledge derived from the erased memory.
- **Audit:**
 - *answerable history* shows what was known about a subject, and when;
 - *decision-time reconstruction* rebuilds the view as it stood at the moment a decision was made;
 - *compliance reports* draw on audited records: erasure events, what was injected into an agent's context, and provenance drift.
- **Access by scope:** sharing contracts are enforced at query time, so an agent cannot recall what its scope does not admit.
- **Reports for your own tools:** SHM includes a report generator. Each report comes as structured data for your own reporting and analytics tools, and as readable text. Every section states whether it recorded events, recorded none, or was never connected, so an empty section is never mistaken for a clean bill of health.

- **The frameworks your reviewers track:** SAIHM's properties map onto GDPR (the Article 15 right of access and the Article 17 right to erasure), CCPA/CPRA, HIPAA, ISO/IEC 27001 and SOC 2, and onto the AI frameworks now taking shape: the EU AI Act, the NIST AI Risk Management Framework and ISO/IEC 42001.

Continuity and independence.

- Work survives the context window, the session, the process restart and the model switch.
- Memory follows the key, not a vendor account (*chapter 2*).
- The key is also your exit. Your memories are yours to keep, move or erase.
- You can run your own endpoint. SAIHM is not responsible for, and has no comment regarding, endpoints developed and run by others.

Policy and control. Where each agent keeps its memory, and the controls for swarms and fleets, are in chapter 8.

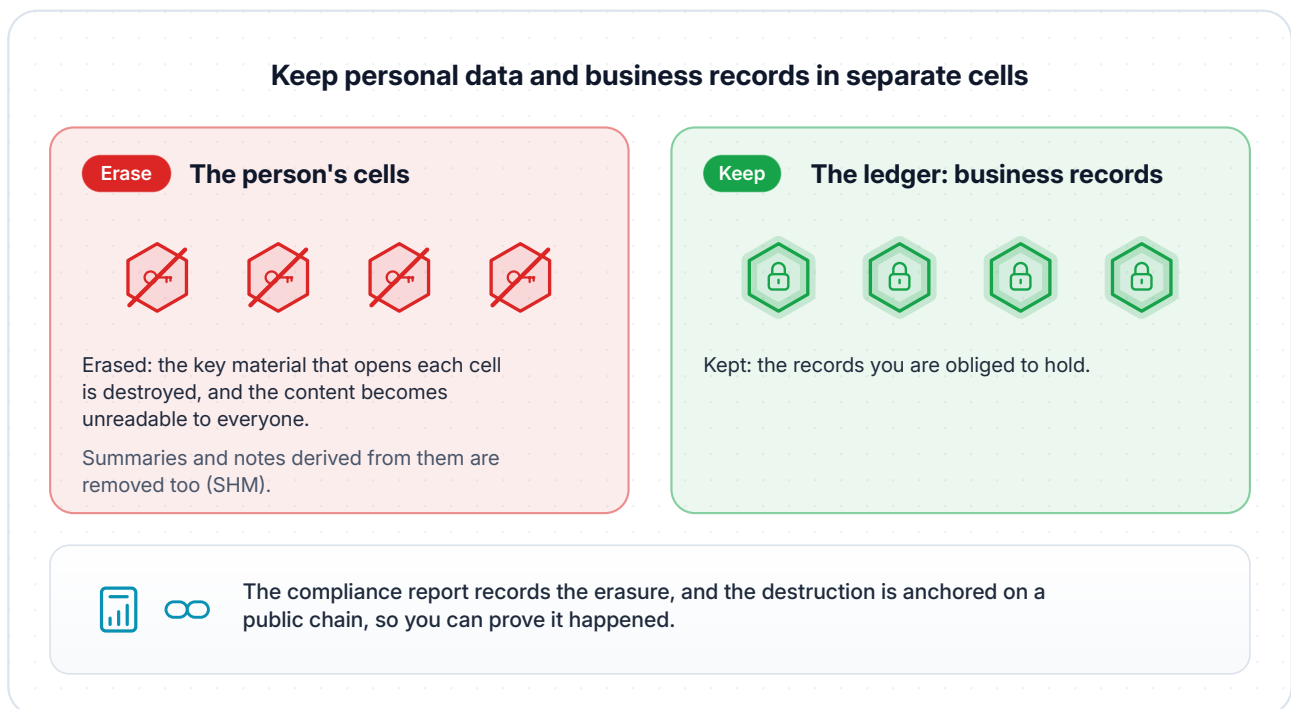


Figure F4: Erase the person, keep the ledger

Scenario: a regulated service desk.

1. A bank's support agents remember each customer's standing instructions and past resolutions, scoped per customer.
2. When a customer closes the account and asks to be forgotten, erasing their cells also removes every summary and note derived from them.
3. The compliance report records the erasure.
4. A year later, an auditor asks what the agent knew when it approved a request. Decision-time reconstruction answers from the record.

7. Between organisations: trade and partnership

When two organisations work together, their agents need to exchange knowledge that is confidential, must last, and must stop being shared when the relationship changes. SAIHM's sharing was designed for exactly that.

Available today. Everything in chapter 3 works across company lines: share exactly what a partnership needs, with an expiry that matches the engagement, and revoke at any time, with no direct connection between the two organisations' systems. Two things matter most between organisations:

- Every shared cell is signed, so both sides can later verify exactly what was shared, by whom and when.
- Each organisation keeps its own private memory of the relationship. No central party scores anyone.

Best practice. Share only what the partnership needs, set an expiry, and keep the signed record of every share.

Coming next, with MKT. Licensing enforcement for shared memory, and memory monetisation, so what your agents know can be licensed and sold on terms MKT enforces. (*Appendix A*)

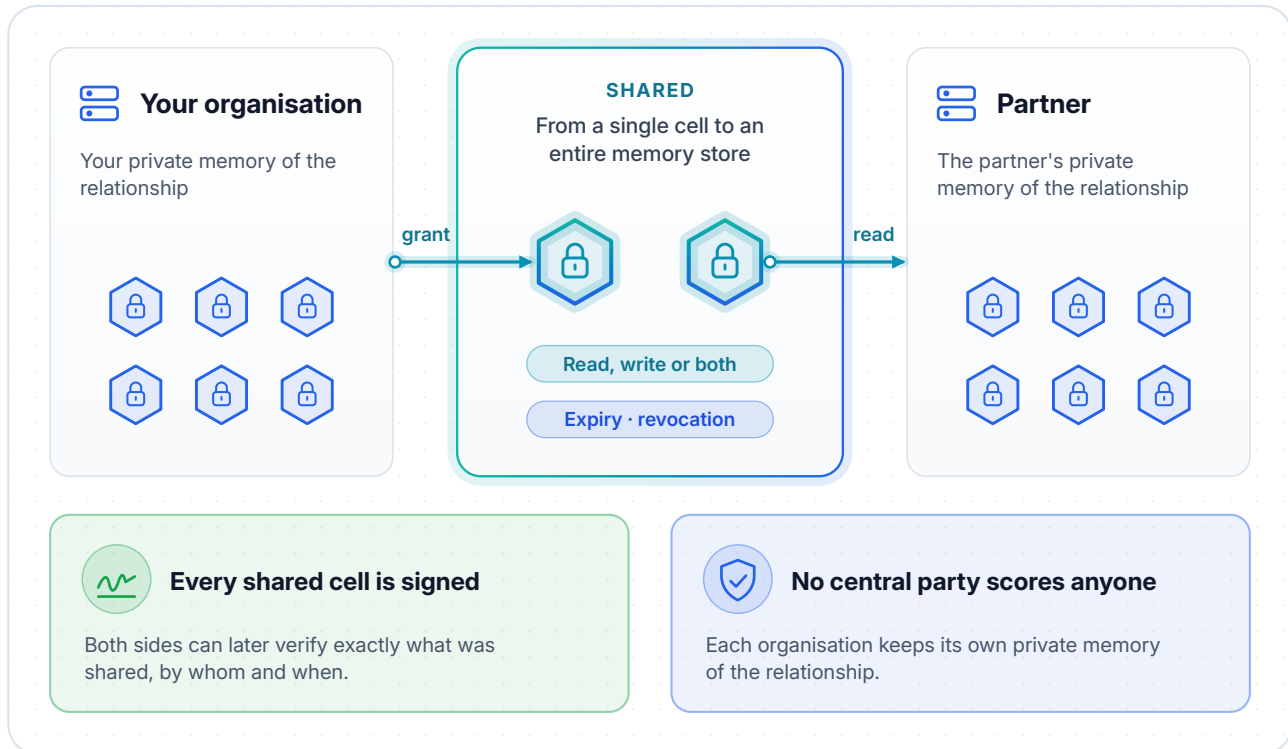


Figure F2: A deal as cells: who holds what, and what is shared with whom

Scenarios.

- **Supplier collaboration.** A manufacturer's procurement agent shares a drawing revision with three approved suppliers' agents for the length of the tender. When the tender closes, the grants lapse. The signed record of what each supplier saw stays with the manufacturer.
- **Joint venture.** Two firms' research agents share findings cell by cell. Each firm's SHM ranks the partner's findings alongside its own. When the venture ends, each side revokes its grants and keeps its own history.
- **Consortium.** A trade consortium distributes the same signed terms to every member's agent. Each member verifies the terms against the consortium's known identity.

8. Swarms, fleets and robots

Agents rarely work alone for long. As fleets grow, from dozens of software agents to thousands of machines, memory becomes the thing that keeps them coordinated without a central controller.

Available today. The capabilities in chapters 2 to 5 work across a whole fleet: one memory layer across models and vendors, the fleet knowledge map, service tiers, and store-and-forward sharing, so a member that was offline catches up when it next checks in. On top of them sit the policies and controls that run a fleet.

Where each agent's memory lives, set by policy. SHM works with three stores: the agent's local store, your organisation's central store, and SAIHM. Each agent has a store policy that names its default store, the stores it may write to, and what happens when a store is out of reach. The policy is set per agent in one of three ways:

- **by the agent itself**, for a standalone agent or a developer's machine;
- **centrally, by your organisation**: the policy is itself a SAIHM polymorphic cell, issued by your organisation's identity and shared to the agent, which verifies it before use and cannot change it;
- **hybrid**: your organisation decides which stores are allowed, and the agent picks its default among them.

Built to keep a fleet's memory whole.

- **Fail-safe.** If an agent cannot read or verify its central policy, it keeps reading from its local store and writes nothing to the shared stores until the policy checks out, so a fleet's shared memory never quietly splits.
- **Redundancy for continuity.** Local copies keep each agent recalling when the central store is out of reach. The policy chooses whether the agent keeps reading locally while shared writes pause, or pauses entirely.
- **Stamped provenance.** Every piece of derived knowledge is stamped with the agent that wrote it and the memories it came from, so it can be verified later and erased along with its sources.
- **Knowledge that stays with the organisation.** Knowledge an agent derives can be shared to your organisation as it is written, so what a fleet learns outlives any single agent.

Swarm and fleet controls at a glance.

Control	What it governs	Set for
Store policy	Where memory is written, which stores are allowed, and what happens when one is out of reach	Each agent, by the agent or by your organisation
Scopes	What each agent may recall	Each agent and query
Sharing grants	Who may read or write which memory, and until when	From a single cell to a whole store
Recipes and levers	How recall trades speed for depth	Each agent and task
Injection budget	How much memory enters the context window	Each agent
Behaviour modes	A named mode, the one question an answer must address, and a word budget	Each agent
Service tiers	Hot, warm and archival memory	Each class of memory
Resume checks	Whether the claims carried into a resumed session still hold	Each session
Attention	What an edit removed from an always-loaded memory index, reported unasked	Each edit
Compliance reports	Erasure, injection, provenance and outbound activity over a window	Any time window

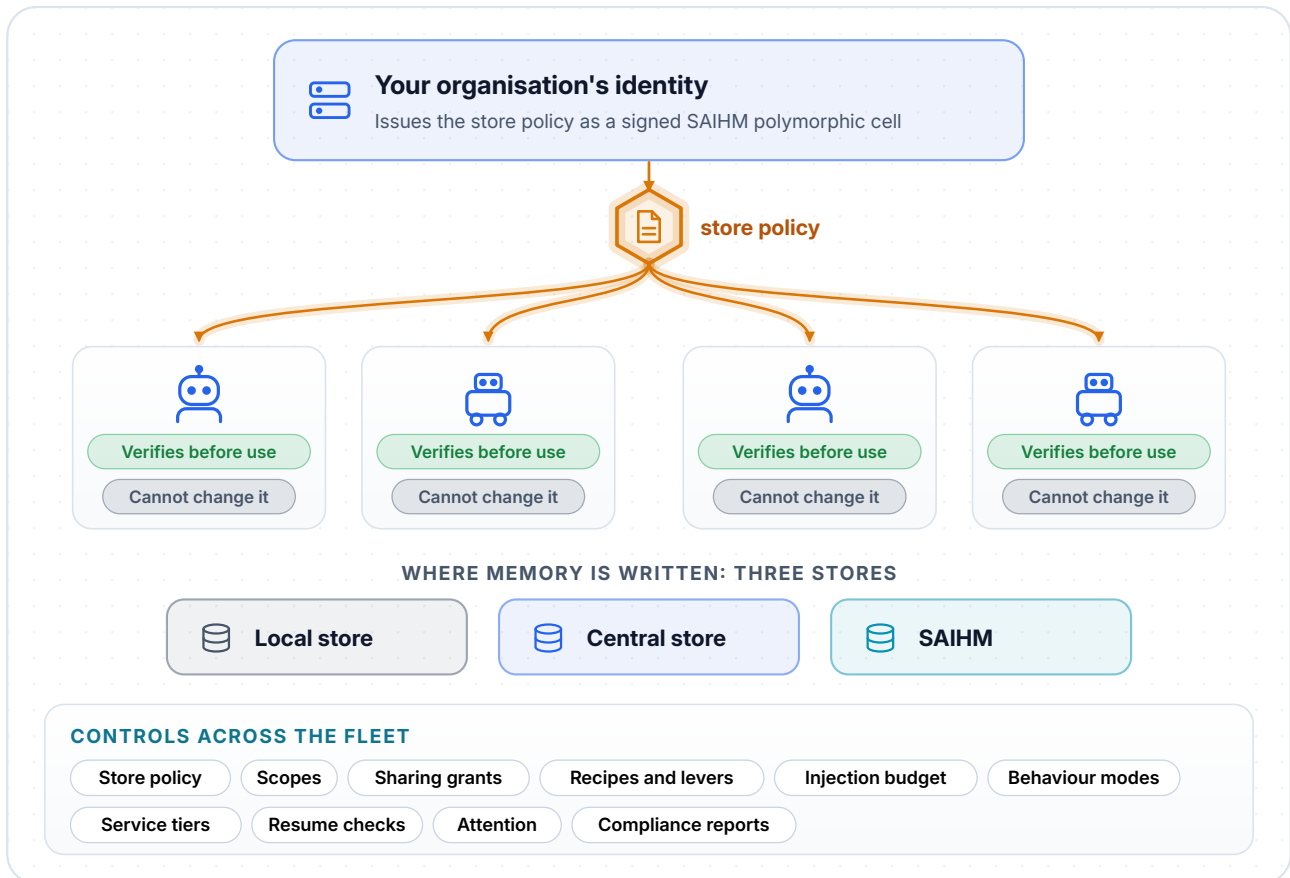


Figure F12: Policy and controls across a fleet

Scenarios.

- **Warehouse robots.**
 - Autonomous mobile robots share a memory of hazards: a spill in aisle 7, a blocked dock door.
 - Each robot's SHM surfaces the relevant hazard before it plans a route through that area.
 - When the hazard is cleared, the cell is erased and the warning disappears fleet-wide.
- **Delivery handoffs.** A night-shift robot leaves a handoff cell, "gate code changed; customer prefers the side door", for the day-shift robot on the same route.
- **Inspection drones.** Drones build a shared site memory: which assets were inspected, what was found, and what needs a second look. The site owner holds the key. The contractor's drones receive grants only for the duration of the contract.
- **Mixed fleets.** Software agents that plan and robots that execute share the same memory layer. A plan written once as a SAIHM polymorphic cell is read as prose by the planning model and as a structured record by the robot's controller, so a plan made by one model is understood by the machine running another.
- **A fleet under one policy.** A logistics company issues one central store policy to its delivery robots. Knowledge goes to the company's store, and a robot that loses its connection keeps working from local memory and never writes knowledge outside the company's policy.

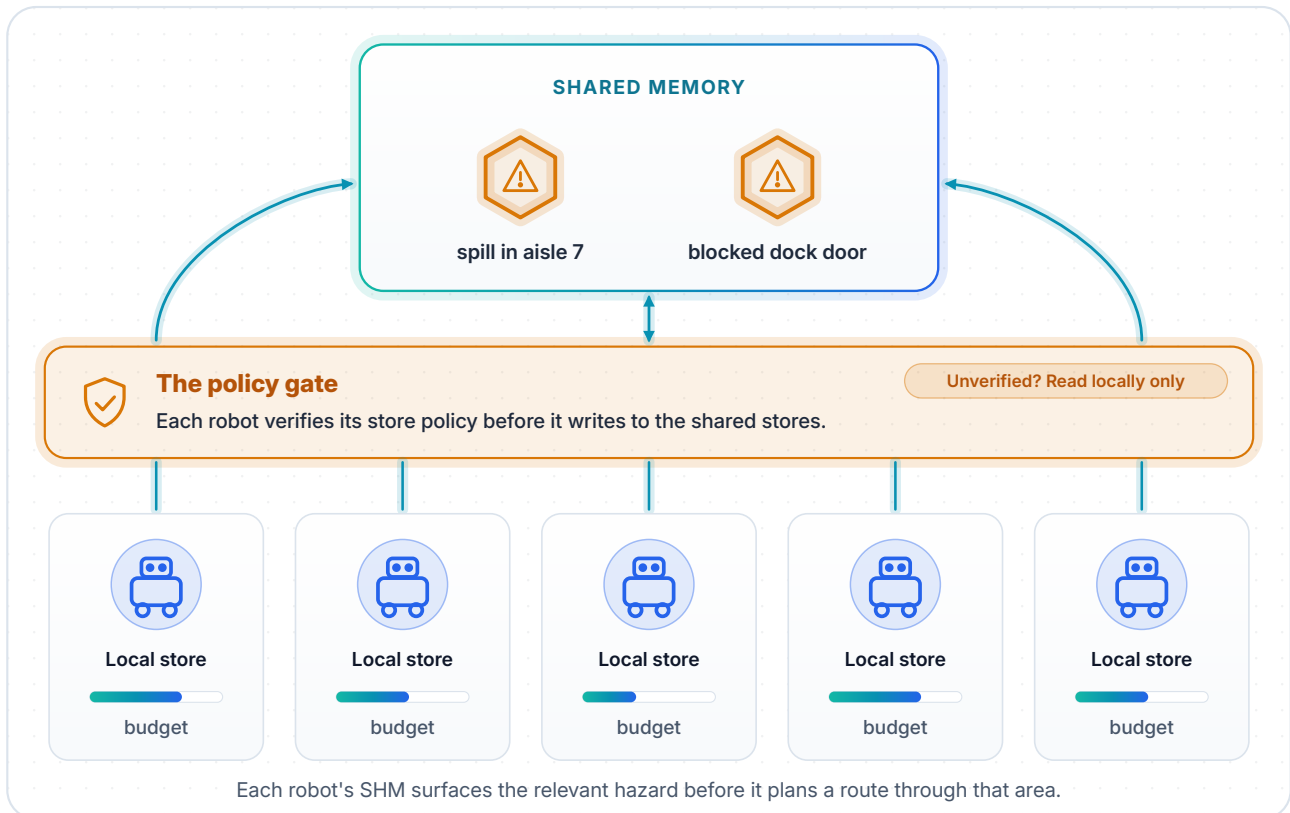


Figure F6: A robot fleet: shared memory, budgets and the policy gate

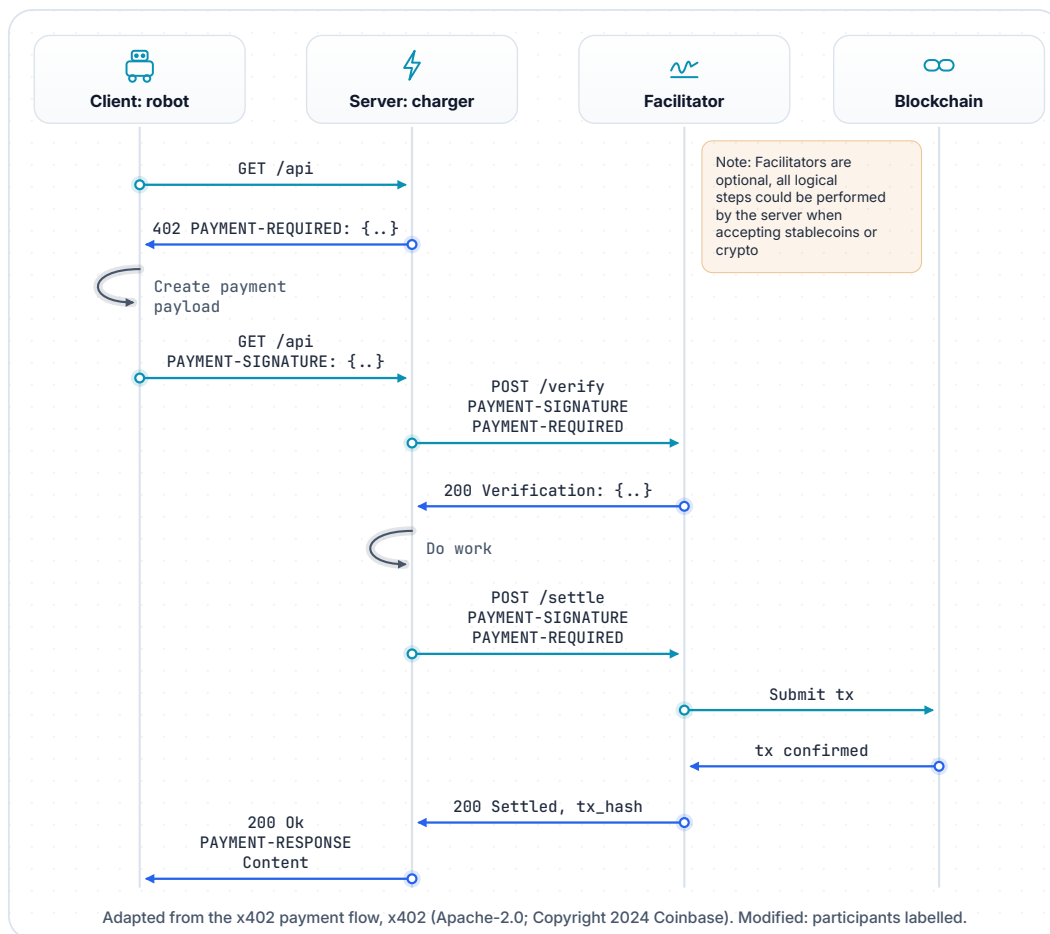


Figure V2: A robot paying a charger over x402. See Appendix A: this is where MKT comes in



9. Why choose SAIHM + SHM

The short version. Most memory layers need to read your text to organise it: they extract, embed and index it on the server that stores it. That server is yours when you self-host and the vendor's when the service is managed. SAIHM is built the other way round. Encryption happens in your agent's process, storage holds only ciphertext, and SHM does the ranking and organising on your side.

On top of that foundation, you get what memory infrastructure should offer:

- SAIHM polymorphic cells: one set of protections for every shape of data;
- memory that is shareable with precision, at any scale;
- signatures that stay verifiable for the long term;
- erasure you can prove.

Structural properties.

Property	SAIHM + SHM
Who holds the key	You. Encryption and decryption happen in your agent's process
What storage can read	Nothing: only ciphertext is stored
Data model	SAIHM polymorphic cells: any shape, one set of protections
Sharing between agents and between organisations	Built in, at any scale, with expiry and revocation (<i>chapter 3</i>)
Integrity	Post-quantum signatures (ML-DSA-65) on every cell
Erasure	Cryptographic, and it cascades into derived knowledge (SHM)
Portability across models and vendors	One identity across any number of models and agent types; memory follows the key
Retrieval and organisation	SHM, running on your side: meaning-based, ranked, proactive, budgeted; multi-task and anti-conflation
Fleet policy and controls	A store policy per agent, set by the agent or centrally, with fleet-wide controls (<i>chapter 8</i>)
Open code where custody lives	The client is Apache-2.0

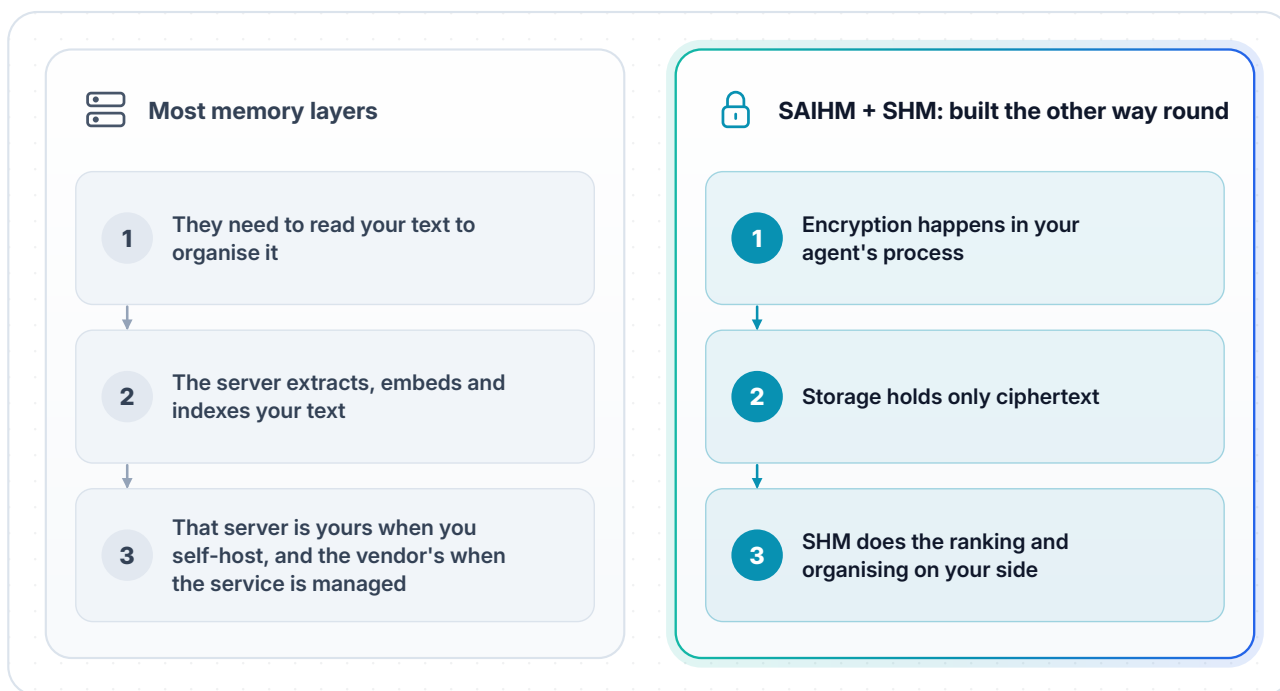


Figure F9: Comparison on structural properties

A named comparison with other memory layers, with dated sources, is in Appendix F.

What we have measured. Every figure below is final and stated with its scope.

- **SAIHM + SHM, on a live memory corpus** (our 151-question evaluation set; because the set was used during development, read these figures as an upper bound):
 - adding SHM raised ranking quality (mean reciprocal rank, MRR) from **0.561** with SAIHM's built-in recall to **0.834**;
 - that is **0.111 above** the field-standard BM25 baseline.
- **SHM's retrieval engine, measured on its own** (LongMemEval-S, all 470 answerable questions, top ten results, full conversation turns indexed):
 - SHM put more of the right evidence in its top ten than BM25: precision **0.609 against 0.552** (**+0.057**, 95% confidence interval 0.041 to 0.073);
 - it covered more of the needed evidence (**+0.030**).
- **SHM's reranker** (LoCoMo, 1,540 judged questions): switching on the reranker raised answer accuracy by **3.1 points** (95% confidence interval +1.1 to +5.3).
- **Proactive recall, in daily use:** SHM brought memory forward on **84 of 102** distinct prompts (82%).
- **AMB:** results will be added when they are delivered.

These are the scopes we measured. Your results will depend on your data and your workload, and we are glad to help you run your own evaluation.

10. Hostile questions, answered

Short answers you can give in the room, with the chapter to point to for more.

Security and custody

1. **Can you read our memory?** No. Memories are encrypted inside your agent's process, only ciphertext is ever stored, and we never hold your key. The client that does the encryption is open source, so your team can verify it. (*Chapter 2*)
2. **Who holds the keys, and what if we lose them?** You do. The key never leaves your agent's process. No one can recover a lost key, not even us, so back it up the way you back up any root credential.

3. **Why post-quantum? Isn't that hype?** ML-DSA and ML-KEM are NIST standards (FIPS 204 and FIPS 203). Memory is long-lived: a record you keep for years should stay verifiable and confidential for years. Post-quantum from day one means you never have to re-sign your history.
4. **How do we verify who shared a memory with us?** Before showing anything, your client checks the cell's signature against the sharer's identity, known in advance from a source you trust, and a share that fails the check is refused. (*Chapter 3*)
5. **What stops a forged share?** The same signature check. A cell that was not signed by the known identity is never shown.

Erasure and compliance

6. **Prove erasure.** SAIHM erases by destroying the key material that opens a cell, and the destruction is anchored on a public chain. SHM removes the knowledge derived from that memory and records the erasure in its compliance report.
7. **Does erasure reach summaries and derived knowledge?** Yes. SHM carries erasure into everything derived from the erased memory, by construction.
8. **How do you square retention law with the right to erasure?** Keep personal data and business records in separate cells. Erase the person's cells when you must, and keep the records you are obliged to hold. (*Figure F4*)
9. **What audit trail exists?** Signed cells, answerable history, decision-time reconstruction, and compliance reports your own reporting tools can read. (*Chapter 6*)

Lock-in and portability

10. **What if we switch model vendors?** Nothing needs to move. One identity serves any number of models, and memory follows the key. (*Chapter 2*)
11. **What if SAIHM or COTI shuts down?** Your key is your exit, and SAIHM is not tied to any one network. Our reference deployment uses the public COTI V2 mainnet to anchor identities and the audit trail, and a decentralised, erasure-compatible storage tier to hold the encrypted cells. If COTI V2 disappeared, SAIHM would not: a fresh deployment would stand up on any ledger and storage that do the same jobs, and nothing your applications or regulators rely on would change. SAIHM is independent of the COTI Group and uses COTI V2 on the same terms as any other user of the public network. And your agent can recall its whole memory at any time and keep its own copy, the client that holds your key is open source, and you can run your own endpoint. (*Chapter 6; Appendix B*)
12. **Can we run SHM ourselves?** Yes. SHM runs on your side, beside your agent.

Performance

13. **Is it fast enough for live agents?** On our evaluation corpus, the base profile answered in a median of 26 ms, and the reranked profile in about a quarter of a second (256 ms). SHM is a resident, always-hot service. (*Chapter 5*)
14. **What do the benchmarks say?** SHM lifted ranking quality from 0.561 to 0.834 over SAIHM's built-in recall on our evaluation set; beat the BM25 standard on precision on LongMemEval-S; and its reranker added 3.1 points of accuracy on LoCoMo. Each result comes with its scope, and AMB results will follow. (*Chapter 9*)
15. **Does it scale to a large memory?** SHM ranks across thousands of records, pages between hot, warm and archival tiers, and consolidates as memory grows.

Maturity

16. **Is this production-ready?** SAIHM's client is published and open source. Every SHM release passes a gate that checks each capability claim against the code, alongside a full test suite.
17. **What is real today, and what is roadmap?** This guide labels every statement: *Available today*, *Scenario* or *Coming next*.
18. **Who stands behind it?** SAIHM Project. See saihm.net.

Architecture

19. **Why not just a vector database?** A vector database stores embeddings. SAIHM stores polymorphic cells and adds custody, signatures, precise sharing and cryptographic erasure; SHM adds ranking, continuity, anti-conflation, proactive recall, budgets and compliance.
20. **Why not A2A or MCP alone?** Use them together. A2A for live tasks, MCP for tools, and SAIHM for knowledge that must persist, stay private and remain erasable. (*Chapter 3*)
21. **Why not the model vendor's built-in memory?** Because your memory should outlive any one vendor, be readable only by you, and be shareable with the agents and partners you choose.
22. **Isn't shared memory just a shared database?** A shared database shares tables. SAIHM shares exactly what you choose, with identities known in advance: encrypted end to end, signed, with expiry, revocation and cryptographic erasure. (*Chapter 3*)

Working with other organisations

23. **What if a partner leaks what we shared?** You control exactly what is shared and with whom: each grant goes to a verified identity, can carry an expiry and can be revoked at any time. And you keep a signed record of what was shared, with whom and when.
24. **What if a partner's agent is compromised?** Revoke its grants and its future reads stop. It cannot sign as you, so it cannot impersonate you to anyone who knows your identity.
25. **How do we revoke access?** Revoke a grant at any time, and it takes effect for every future read. Expiries lapse on their own.
26. **Can we license or sell what our agents know?** That is coming next. Licensing enforcement and memory monetisation are planned for release within MKT, SAIHM's next add-on. (*Appendix A*)

Robots and devices

27. **What happens when the network drops?** Sharing is store-and-forward: nothing is lost, and new shares are picked up when the device next checks in.
28. **What about device identity?** Every agent and device has its own cryptographic identity. We recommend keeping device keys in secure hardware.

Cost and licensing

29. **What does it cost?** Current pricing is on saihm.net. Developers: see Appendix E for discount codes.
30. **Does it save tokens?** Yes. Instead of re-sending a whole transcript, your agent receives a budgeted set of the most relevant memories, measured in tokens.
31. **Who governs the protocol?** The governance tools already ship in the client, ready for when protocol governance opens.
32. **What are the licences?** The client is Apache-2.0. For SHM's terms, see saihm.net.

Policy and control

33. **Can we control where each agent's memory is kept?** Yes. Each agent has a store policy: its default store, the stores it may write to, and what happens when one is out of reach. The agent can set it, or your organisation can issue it centrally. (*Chapter 8*)
34. **How do we govern a whole swarm or fleet?** With central store policies, scopes, sharing grants, per-agent recipes and budgets, behaviour modes, service tiers and fleet-wide compliance reports. (*Chapter 8*)

Commerce

35. **Can agents pay each other?** That is exactly what MKT is being built for. (*Appendix A*)



11. FAQ

What is a SAIHM polymorphic cell? One memory, in any shape: stored once, encrypted and signed, and read back in whatever shape the asking agent needs. (*Chapter 2*)

How do we start? Add SAIHM to any MCP-capable agent and tell it "Join SAIHM". That activates free memory.

Which agents and frameworks work with it? Any agent that can use MCP tools. LangGraph users also have a drop-in store, `saihm-langgraph`.

Where does our data sit? The encrypted cells sit in SAIHM's storage, or with an endpoint you run yourself. The plaintext exists only in your agent's process.

How does sharing work? Your agent shares anything from a single cell to an entire memory store, for reading, writing or both, with agents whose identities are known in advance. Every grant can carry an expiry and be revoked at any time. (*Chapter 3*)

What does SHM add? Meaning-based ranked recall, multi-task continuity, anti-conflation, proactive memory, token budgets, and compliance tooling. It runs on your side.

Where do we find pricing? `saihm.net`. Developers, see Appendix E.

12. Checklists

For sales teams

Discovery questions

- How many agents do you run today, and how many do you expect in a year?
- What happens to an agent's knowledge when a session ends, or when you switch models?
- What does your security review ask about AI memory and data custody?
- Do you have erasure or retention obligations (GDPR, HIPAA, sector rules)?
- Do your agents need to work with partners', suppliers' or customers' agents?
- Do you run, or plan to run, robots or other autonomous devices?

Proof points, and where they live

- Custody and encryption: chapter 2, and the open-source client.
- SAIHM polymorphic cells, and one identity across models: chapter 2.
- Fleet policy and controls: chapter 8.
- Post-quantum signatures: chapter 2 (ML-DSA-65, ML-KEM-768).
- Sharing between agents and organisations: chapters 3 and 7.
- SHM capabilities: chapter 4.
- Measured results: chapter 9, each with its scope.
- Compliance: chapter 6.

Say what's proven

- Quote capabilities with their chapter.
- Present scenarios as illustrations.
- Give measured results with their scope.
- Send pricing questions to `saihm.net`.

Objections. Chapter 10 has an answer for each.

Where SAIHM + SHM create value first

- Regulated industries.
- Multi-agent programmes.



- Partner and supply-chain collaboration.
- Fleets of agents or robots.
- Any organisation whose security review asks where the plaintext lives.

For agent developers

Integration

1. Add the SAIHM MCP server to your agent's configuration.
2. Tell the agent "Join SAIHM".
3. Write decision-shaped cells.
4. Enable SHM and pick a recipe (chapter 5).
5. Set each agent's store policy (chapter 8).

Security review pack

- The client source (Apache-2.0).
- The custody explanation (chapter 2).
- The key-backup plan.
- How each partner's identity is recorded and checked (chapter 3).

Sharing test

1. Share a cell with a known test identity. It reads.
2. Negative case: try to read with an identity that was not granted. It is refused.
3. Revoke the grant. The next read is refused.

Erasure test

1. Erase a cell. It no longer recalls.
2. Negative case: check that knowledge derived from it is gone too, and that the compliance report records the erasure.

Tuning

1. Start with the balanced recipe.
2. Measure on your own corpus.
3. Adjust one lever at a time.
4. Decide which recipes each agent may use, and let it switch between them as its tasks change.

Before production

- Keys backed up.
- Scopes defined.
- Default expiries chosen.
- Compliance report reviewed.
- Recipes chosen for each agent and task.
- Store policies issued.

13. Join SAIHM: the case for now

Memory compounds. Every decision your agents remember today makes tomorrow's work better. Owned memory starts compounding from the first cell, and it keeps compounding through model changes, team changes and vendor changes.

Shared memory has network effects. Each agent, team and partner that joins makes secure sharing more valuable for everyone already there.

Agent commerce is being standardised now. The protocols that let agents buy, sell and pay are arriving this year. The organisations whose agents already hold trusted, signed, private memory will be ready first, and MKT builds on exactly that memory.



Start today. Tell your agent **"Join SAIHM"** to activate free memory. For current pricing, visit **saihm.net**. Developers: see Appendix E for discount codes, available in limited numbers.

Help SAIHM grow. If this guide helped you, share it and recommend SAIHM to your peers. Star our repositories on GitHub, and support our work through GitHub Sponsors:

- SAIHM on GitHub: <https://github.com/SAIHM-Admin>
- Star: [saihm-mcp](#) · [saihm-mcp-server-pro](#) · [saihm-client-pro](#) · [saihm-langgraph](#)
- Sponsor: <https://github.com/sponsors/SAIHM-Admin>

Appendix A: MKT, the next add-on (Coming next)

Agents are learning to trade. In 2026, commerce between agents is settling into five layers, each with an open standard:

Layer	Standards
Talk	A2A, MCP
Find	Agentic Resource Discovery, Agent Cards, on-chain agent registries
Shop	the Universal Commerce Protocol (UCP), the Agentic Commerce Protocol (ACP)
Authorize	the Agent Payments Protocol (AP2), and the card networks' agent programmes
Settle	x402, the Machine Payments Protocol (MPP), cards, bank transfers, stablecoins

The rails move the money. MKT keeps the deal. Every one of those steps produces something worth keeping: a mandate, a quote, an acceptance, a receipt, and the evidence for a dispute. MKT is designed to give agents a private, signed and remembered record of every deal, built on the memory described in this guide.

What MKT is designed to add:

- **Principals and mandates:** agents bound to the people and organisations they act for, with delegated authority that has limits, an expiry and a way to revoke it.
- **Offers and discovery:** signed offers published to the open discovery standards.
- **Deal rooms:** request for quote, quote, counter and accept, carried as shared cells, with both parties signing the final terms.
- **Payment orchestration across the rails:** AP2, ACP, UCP, x402, MPP, cards, bank transfers and stablecoins. MKT never holds your funds.
- **Delivery against payment for digital goods:** knowledge released the moment payment is confirmed.
- **Licensing enforcement and memory monetisation:** license what your agents know on terms MKT enforces, and earn from it.
- **The commercial record:** receipts both parties hold, with personal data kept apart from the ledger so each can be governed on its own terms.
- **Trust and risk:** signed feedback from verified counterparties.
- **Real-time events** for time-sensitive deals.
- **Protocol bridges,** so MKT speaks the standards your partners already use.
- **An operator console:** approvals, budgets and a kill switch.
- **A kit for machines:** robots that pay for charging, parking, compute and data within the budgets you set.

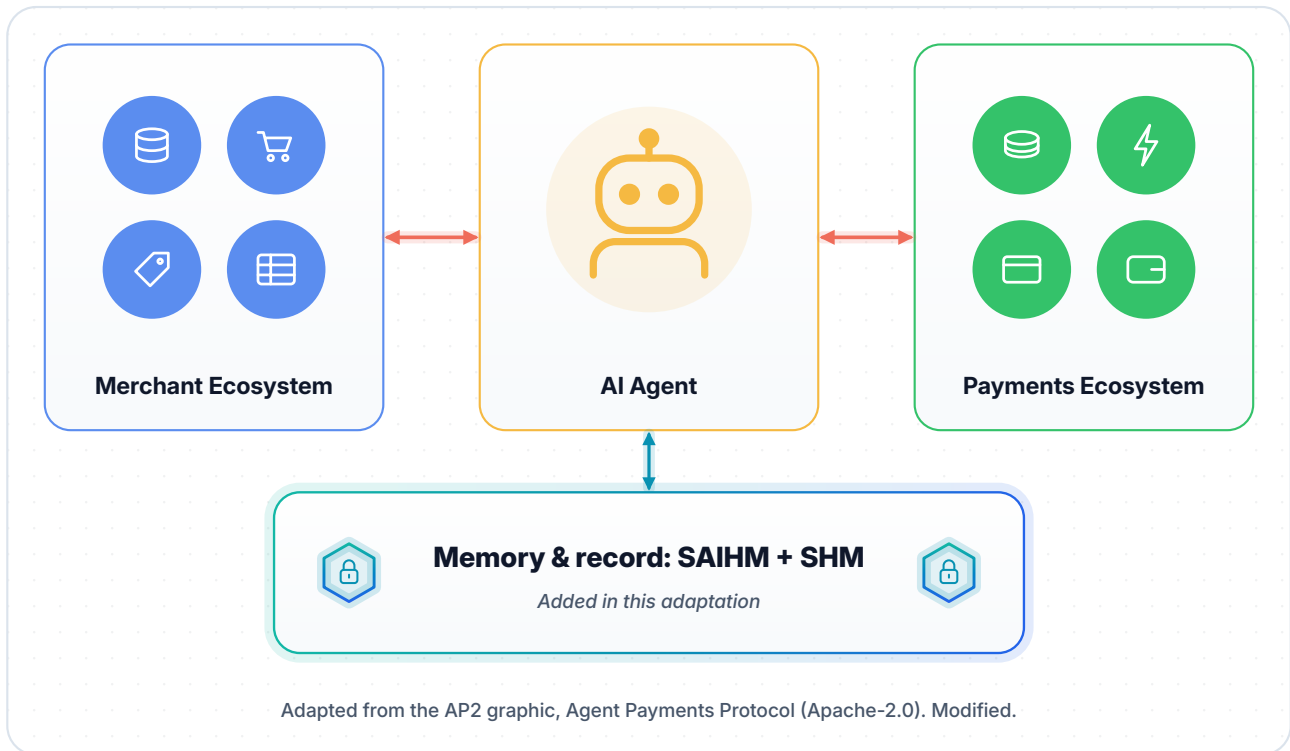


Figure V1, adapted: the AP2 triad with a "Memory & record: SAIHM + SHM" panel beneath the agent

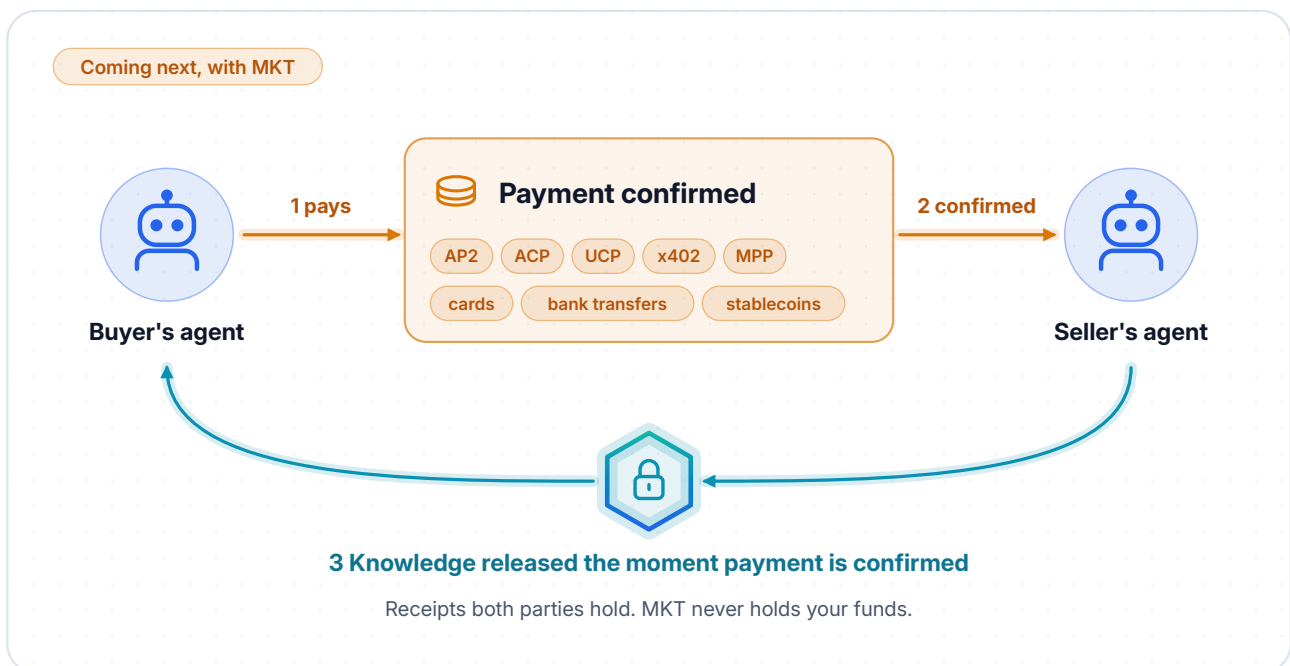


Figure F5: Delivery against payment for knowledge

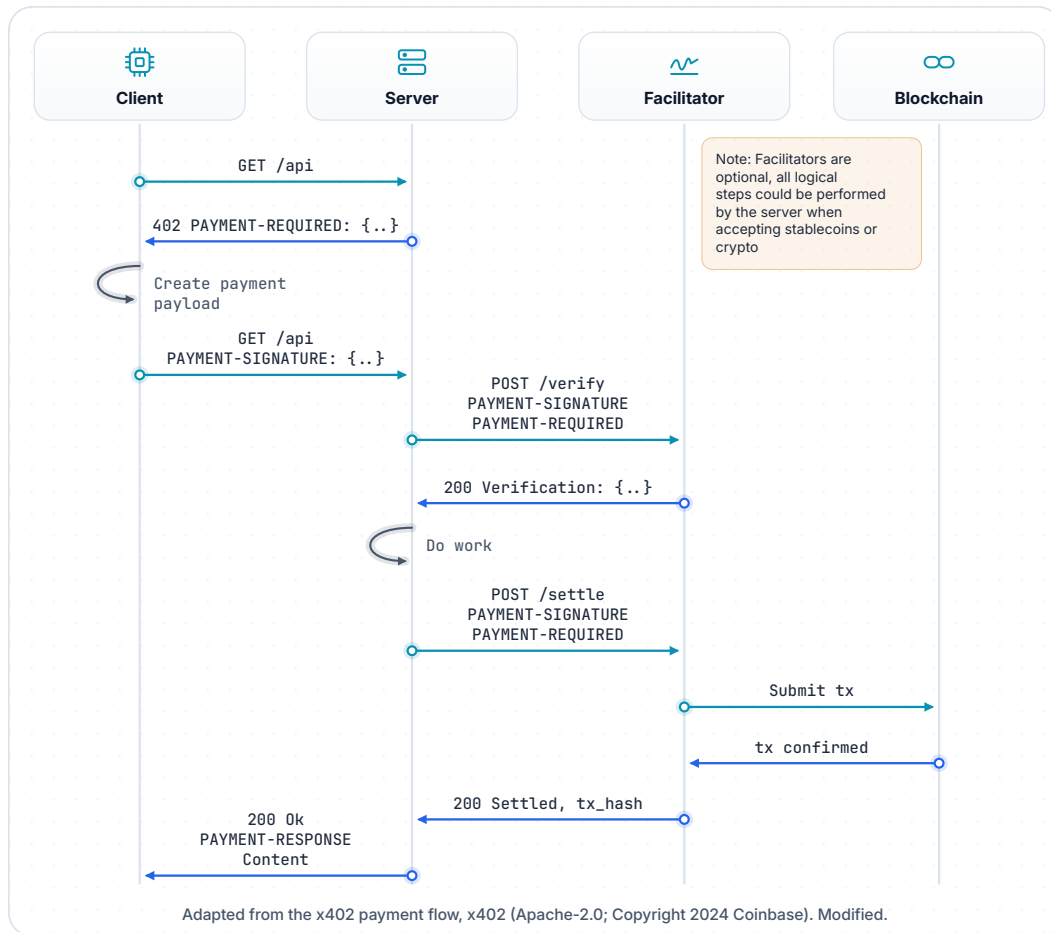


Figure V2: The x402 payment flow

Appendix B: Sources and verification

All external facts were checked on 2026-09-24/25 against the sources below, and the saihm.net post on 2026-10-01.

SAIHM

- The SAIHM client, open source under Apache-2.0: the npm package [@saihm/mcp-server-pro](https://www.npmjs.com/package/@saihm/mcp-server-pro), with the cryptography in [@saihm/client-pro](https://www.npmjs.com/package/@saihm/client-pro).
- *Where AI memory lives: the substrate underneath SAIHM* (saihm.net, 2026-05-21): <https://saihm.net/blog/2026-05-21-where-ai-memory-lives>

Post-quantum standards

- NIST FIPS 203 (ML-KEM) and FIPS 204 (ML-DSA), published August 2024.

Agent protocols

- A2A specification: <https://a2a-protocol.org/latest/specification/>
- A2A and MCP: <https://a2a-protocol.org/latest/topics/a2a-and-mcp/>
- SLIM: <https://github.com/agtncy/slim>
- The ANP white paper: <https://arxiv.org/abs/2508.00007>

Benchmarks

- LongMemEval: Wu et al., 2024 (<https://arxiv.org/abs/2410.10813>)
- LoCoMo: Maharana et al., 2024 (<https://arxiv.org/abs/2402.17753>)

Agent commerce

- AP2: <https://ap2-protocol.org/>
- ACP: <https://www.agenticcommerce.dev/>
- UCP: <https://ucp.dev/>
- x402: <https://www.x402.org/>
- MPP: <https://mpp.dev/>
- Visa Trusted Agent Protocol: <https://developer.visa.com/capabilities/trusted-agent-protocol/overview>
- ERC-8004: <https://eips.ethereum.org/EIPS/eip-8004>
- Web Bot Auth: <https://datatracker.ietf.org/doc/draft-ietf-webbotauth-httpsig-protocol/>

Comparators: see Appendix F.

Appendix C: Glossary

- **Cell (SAIHM polymorphic cell):** one memory, in any shape, encrypted and signed; stored once and read back in whatever shape the asking agent needs.
- **Share / grant:** access to memory, from a single cell to an entire store, for identities known in advance, as read, write or both, with an optional expiry.
- **Pinned (known) identity:** a counterparty's identity, recorded in advance from a source you trust, so that nobody else can pose as them.
- **Cryptographic erasure:** destroying the key material that opens a cell, which leaves its content unreadable to everyone.
- **ML-DSA / ML-KEM:** the NIST post-quantum algorithms for signatures and key exchange.
- **SHM (Super-Human Memory):** the layer on top of SAIHM that organises memory: ranked recall by meaning, multi-task continuity, anti-conflation, proactive memory within a budget, and compliance tooling.
- **Scope:** the set of memories a given recall may draw from.
- **Tier:** where a memory lives for speed: hot, warm or archival.
- **MRR (mean reciprocal rank):** how high the first right answer appears in the results, on average.
- **Precision@10:** the share of the top ten results that are relevant.
- **BM25:** the long-standing standard for keyword ranking.
- **MKT:** SAIHM's coming add-on for commerce between agents.

Appendix D: Figures and credits

Figures to draw (briefs are in the visuals register):

- F1: the agent stack, and the memory layer beside it.
- F2: a deal as cells.
- F3: a messaging protocol and a memory protocol, side by side.
- F4: erase the person, keep the ledger.
- F5: delivery against payment for knowledge.
- F6: a robot fleet.
- F7: where plaintext exists.
- F8: the lever map.
- F9: the structural comparison.
- F10: one cell, many shapes.
- F11: no direct connection: agents on separate networks, sharing through SAIHM.
- F12: policy and controls across a fleet.
- F13: one identity, many models.

Figures to reuse:



- **V1**, the AP2 graphic, and **V2**, the x402 payment flow. Both come from Apache-2.0 repositories: credit each project and include the licence notice, and mark V1 as modified once it is adapted.
- **V3**, the A2A and MCP diagram. Confirm its licence before reuse.
- **Logos** are used only under each owner's brand guidelines.

Appendix E: Discount codes for developers

Discount Codes

The most helpful thing we can do for the developer community writ large in addition to the enterprise-class products we publish would be a discount code to give you more latitude to explore SAIHM's capabilities and use cases as you bundle SAIHM+SHM into your enterprise, B2B, government, and regulated industries solutions.

Use these codes when first subscribing or upgrading using the Stripe option. Each is good for a 1 month subscription purchase, any tier.

Code	Valid through
SAIHM-90-OFF	December, 2026
SAIHM-75-OFF	March, 2027
SAIHM-50-OFF	June, 2027
SAIHM-25-OFF	December, 2027

Each has a generous but limited availability and are designed for developers such as yourself to move forward with using SAIHM to deliver solutions. Share and exploit each. Use the discounts to unlock additional margin when delivering solutions.

There are just 100,000 of each discount code available for redemption. Use them now or lose them forever.

For current pricing, visit saihm.net.

Appendix F: Other memory layers, at a glance

Checked 2026-09-25 against each project's public material. This is not a benchmark comparison.

Product	Licence of the open-source part	Ways to run it
Mem0	Apache-2.0	Managed platform, or self-hosted
Zep	Graphiti, its open-source engine, is Apache-2.0	Managed (Zep Cloud); the Graphiti engine can be self-hosted
Letta	Apache-2.0	Self-hosted agent server
Honcho	AGPL-3.0	Managed, or self-hosted
SAIHM + SHM	The client is Apache-2.0	Managed storage that holds only ciphertext, or your own endpoint; SHM runs on your side

**Sources:**

- Mem0: <https://docs.mem0.ai/open-source/overview>
- Zep: <https://help.getzep.com/faq>
- Letta: <https://github.com/letta-ai/letta>
- Honcho: <https://github.com/plastic-labs/honcho>

Appendix G: Prompts, chapter by chapter

Plain-language requests your agents can act on once SAIHM and SHM are set up. Change the wording as you like; some prompts depend on which SHM capabilities your developers have switched on.

Topic	Prompt
Getting started (chapter 2)	"Join SAIHM."
Remembering (chapter 2)	"Remember that we chose supplier B for the Q3 order, and why."
Erasing (chapter 2)	"Forget everything we hold about customer 4471."
Sharing (chapter 3)	"Share the interface specification with the Singapore build agent, read-only, for two weeks."
Revoking (chapter 3)	"Revoke the supplier's access to the tender drawings."
Recall by meaning (chapter 4)	"What did we decide about the retry policy, and when?"
Continuity (chapter 4)	"Pick up the Atlas migration where we left off."
Keeping tasks apart (chapter 4)	"Keep the Acme and Globex cases separate."
Recipes (chapter 5)	"Switch to Fast and lean for this triage session."
Compliance (chapter 6)	"Produce this month's compliance report for our reporting tool."
Audit (chapter 6)	"What did we know when we approved request 1182?"
Partners (chapter 7)	"Share our delivery forecast with Acme's planning agent for 30 days."
Fleets (chapter 8)	"Show where knowledge is thinning across the fleet."
Policy (chapter 8)	"Which stores does your policy let you write to?"

Licence notices

Figures V1, V2 and V3 are adapted from material published under the Apache License, Version 2.0 (<https://www.apache.org/licenses/LICENSE-2.0>). Each has been modified.

- **V1** is adapted from the AP2 graphic, Agent Payments Protocol: <https://github.com/google-agentic-commerce/AP2>.
- **V2** is adapted from the x402 payment flow, x402: <https://github.com/coinbase/x402>. Copyright 2024 Coinbase. Licensed under the Apache License, Version 2.0; distributed on an "AS IS" basis, without warranties or conditions of any kind.
- **V3** is adapted from the A2A and MCP diagram, A2A Protocol: <https://github.com/a2aproject/A2A>.