

Protecting agent data and memory

Designing purpose, access, retention and deletion across the full information lifecycle.

An agent's information footprint extends beyond the conversation. Map the copies, derived records and retrieval paths that influence future actions, then connect each to purpose, access, retention and verifiable handling.

FOR Privacy engineers, data owners and agent platform teams

Softlander Consultancy FZE LLC | 2026-09-16 | v1.0

The lifecycle model and worked scenario are illustrative engineering guidance. Applicable legal obligations require assessment for the actual organization, processing and jurisdictions.

In this paper

Follow the information after the conversation	2
Separate the stores by purpose	3
Make access survive transformation	4
Decide what the agent is allowed to remember	5
Make retention and removal technically specific	6
Worked example: a support agent that remembers too much	7
Observe enough without collecting everything	8
Include external processors in the map	9
A lifecycle worksheet for the engineering review	10
References	11

Follow the information after the conversation

A user deletes a conversation and assumes the information is gone. The application may still hold a summary, a search index entry, a cached tool response, an evaluation sample or an operational trace. Whether those copies are appropriate depends on the purpose and conditions of processing. The engineering problem is to know which copies exist, how they are used and what a deletion or correction request actually changes.

Agent memory should therefore be treated as a collection of information products with different lifecycles, not a single feature toggle. Some state is needed only to complete a task. Some may support continuity across sessions. Some is retained to investigate an incident or demonstrate an authorized action. Combining these purposes in one unrestricted store makes access, correction and retention harder to reason about.

The NIST Privacy Framework treats privacy through enterprise risk management and is voluntary guidance rather than a statement of legal obligations. [1] This paper develops a practical engineering approach: trace information from collection through use, derivation, sharing, retention and removal, and identify the owner and enforceable rule at every transition.

The approach is particularly useful when an agent combines retrieval, tools and persistent memory. It also applies to simpler assistants, but the investment should match the

information and consequences involved. A public-document summarizer and an assistant handling employee case files should not inherit the same retention and logging defaults merely because they use the same model provider.

Begin with one realistic task. Follow a piece of information through the actual system, including error paths and diagnostics. Ask where it can reappear later. If the answer depends on an undocumented cache, a vendor default or a developer's local export, that is a concrete design gap. A privacy statement cannot resolve an information path the team has not identified.

Separate the stores by purpose

A useful first map distinguishes six stores: source records, transient working context, persistent user or task memory, retrieval indexes, operational evidence and development datasets. They may share infrastructure, but they should not silently share rules. A source record may be authoritative; a generated summary is a derived interpretation. An audit event may justify retention of a transaction fact while providing no reason to retain the full conversation.

Transient context should have a clear completion boundary. Determine what remains when the task finishes, fails or is abandoned. A job queue may retain payloads after the user closes the interface. A retry mechanism may duplicate them. A failed tool invocation may include sensitive arguments in an error message. These paths deserve the same design attention as successful responses.

Persistent memory needs an explicit purpose and an update rule. A preference such as a user's chosen report format is different from an inferred health concern or an assessment of an employee's behavior. Do not treat all statements in a conversation as suitable material for future personalization. Specify which information can be written, who can inspect it, how conflicts are resolved and when it expires.

A retrieval index is a separate information asset, even when it is built from an existing repository. OWASP identifies unauthorized access and cross-context leakage among vector and embedding risks. [2] The practical review should examine both the indexing process and the query-time access decision; possession of a source document at ingestion does not establish every future user's right to retrieve its contents.

Operational evidence should be designed around the question it must answer. If the question is whether an approved action occurred, a transaction reference, authorization outcome and timestamp may be enough. Retaining the entire prompt "just in case" can create a larger information exposure than the original task. Development datasets require their own purpose and controls rather than automatically inheriting permission from production use.

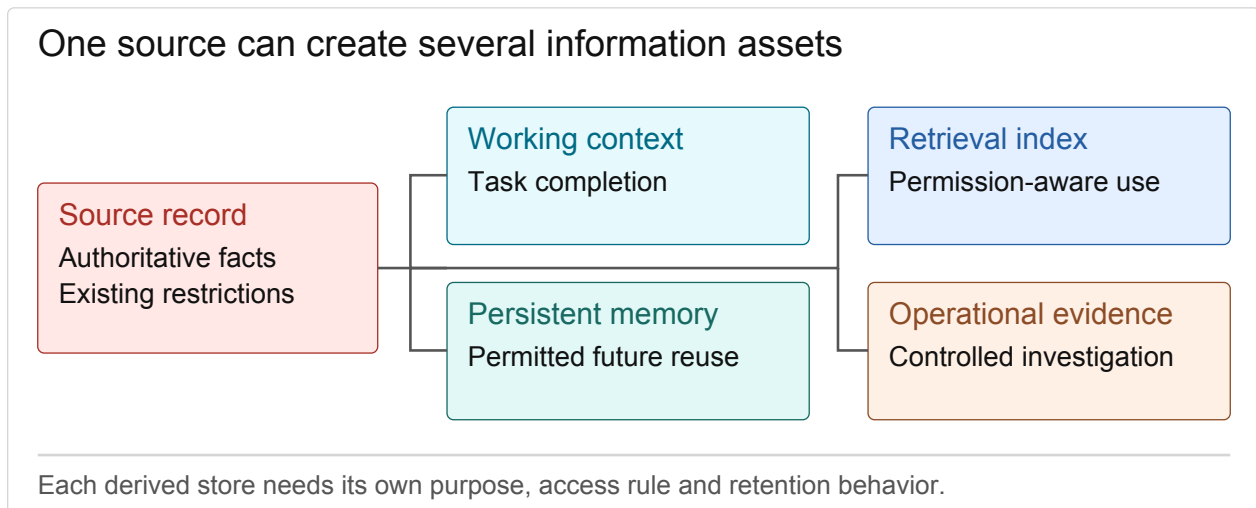


Figure 1. An agent's information footprint extends beyond the visible conversation. Original Softlander conceptual diagram, 2026-09-16.

Make access survive transformation

Information often loses its access context when it is transformed. A source document may carry a tenant, owner and classification, while an extracted paragraph or summary arrives in another store as plain text. The new representation can then be retrieved or displayed without the restrictions that protected the original. Preserve enough provenance and access context to make the downstream decision correctly.

For retrieval, consider the full sequence: candidate selection, permission filtering, ranking, context assembly and response generation. A design that filters only after sensitive text reaches the model has already crossed one boundary. Where feasible, enforce access before assembling the model context and verify the final response does not include material from excluded sources. The precise implementation depends on the retrieval platform and identity model.

Derived information can combine sources with different restrictions. A summary of several documents should not automatically become as accessible as the least restricted document. Define a combination rule and test it. If the system cannot preserve the necessary restrictions, retain the summary only within the narrower scope or avoid persisting it. Convenience does not determine the appropriate audience.

OWASP's sensitive-information-disclosure guidance identifies the risk of exposing personal, proprietary or other protected information through LLM applications. [3] For an agent, that exposure may occur through a tool argument or stored memory as well as the final answer. Review outbound calls, notifications and diagnostic records, not just the text shown in the chat window.

Authorization changes also need propagation. If a person leaves a team, their future access to derived summaries and cached results should reflect that change. A cache key that includes only the query text can be unsafe in a multi-user system if it allows another user to receive a result assembled under broader permissions. Include the relevant access context or avoid sharing such results across contexts.

Test these boundaries with deliberately similar records from separate users or tenants. The test should make an unauthorized result recognizable without using real personal information. Inspect retrieval candidates, assembled context and output separately. A clean final response can conceal an earlier unauthorized access; a blocked retrieval can coexist with a leak from persistent memory populated during an earlier task.

Decide what the agent is allowed to remember

Memory policy should answer three questions: what may be recorded, what may be inferred and what may be reused. A user may explicitly provide a delivery preference. The agent may infer a preference from behavior. Those are not equivalent records, and neither should automatically become a permanent fact about the person. Preserve the source and uncertainty of an inference when it is necessary to retain it.

For each memory type, define a write condition. Examples include an explicit user request to remember a preference, completion of an approved workflow step or a verified update from an authoritative system. Avoid allowing any retrieved document to write instructions into persistent memory. Material encountered during a task is evidence to interpret, not a new source of authority over future tasks.

Define a conflict rule as well. Suppose a remembered address differs from the current account record. The agent should know which source is authoritative for the operation. It may ask for clarification or refresh the record, but it should not choose whichever value makes completion easiest. A memory entry should carry enough context to determine whether it is stale, disputed or superseded.

NIST's Generative AI Profile discusses data privacy and information-integrity concerns among its risk areas. [4] The proposed memory ledger here makes those concerns operational by recording purpose, provenance, permitted reuse, review conditions and expiry for each class of retained information. It is an engineering pattern to evaluate, not a prescribed NIST data model.

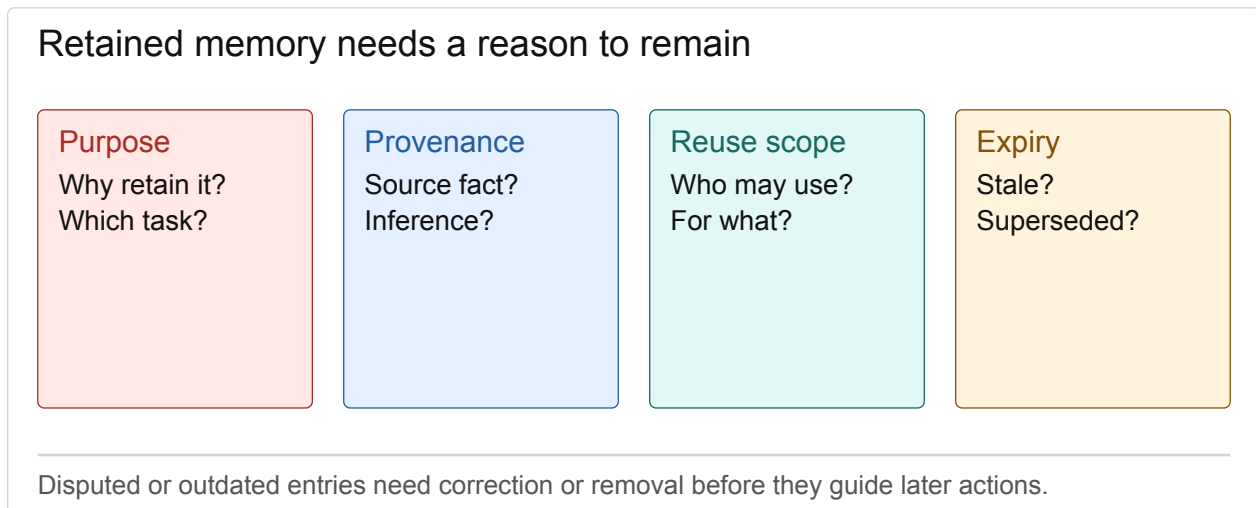


Figure 2. Four questions distinguish appropriate memory from indiscriminate persistence. Original Softlander conceptual diagram, 2026-09-16.

Make correction visible to the person who depends on the result. A user should not have to guess whether editing a profile also updates a stored summary. Define whether a correction changes the source, the derived memory, both or neither. If historical evidence must be retained, separate the historical record from the current value used for action and restrict access appropriately.

Some organizations will reasonably choose no cross-session memory for a sensitive workflow. Others may accept carefully bounded preferences or task state. Evaluate the utility against the handling burden. "More context" is not a sufficient reason to persist information indefinitely, especially when the system cannot explain why a remembered fact is still relevant.

Make retention and removal technically specific

Retention should be defined per information class and purpose. A single organization-wide number may conceal important differences between transient task data, business records, security evidence and backups. The authorized retention schedule should come from the organization's requirements; engineers should translate it into deletion, expiry, archival and access behavior that can be tested.

A deletion request needs a scope statement. Does it remove the visible conversation, prevent future retrieval, erase a persistent memory entry, remove development copies or trigger a request to a processor? State what happens immediately and what follows a separate lifecycle. Avoid presenting a successful user-interface action as proof that every downstream copy has been removed.

Build a removal map using the information footprint. Identify the authoritative record, derived stores, caches, queued jobs and external recipients. For each, record the handling

mechanism and evidence of completion. Where the system cannot delete an individual item from a backup immediately, document the applicable lifecycle and prevent inappropriate restoration into active use. Do not invent a guarantee that the platform cannot support.

NIST SP 800-88 Revision 2 addresses media sanitization. [5] That is relevant when storage media is cleared or retired, but it is not a substitute for application-level deletion design. Removing a database row, expiring an index entry and sanitizing a storage device are different operations with different evidence.

Deletion can also conflict with an authorized preservation requirement. Engineers should not make that legal or business determination implicitly. Route the conflict to the responsible privacy, legal or records owner, restrict the preserved copy and stop using it for unrelated purposes. The application should distinguish “removed from active use” from “retained under a controlled exception” when that distinction matters to the user or operator.

Test restoration and re-ingestion. A removed document can reappear if a nightly indexing job reads an unchanged source, or if a restored backup repopulates the active store without applying the removal record. Verification therefore needs to include the next lifecycle event, not only an immediate query after deletion. A controlled tombstone or equivalent suppression mechanism may be appropriate, but its own retention and access should be justified.

Worked example: a support agent that remembers too much

Illustrative scenario. A software provider wants an agent to assist account-support staff. The agent reads support tickets and account documentation, drafts replies and remembers recurring technical preferences. A customer mentions sensitive personal circumstances in a ticket to explain why they need more time. The scenario is constructed; it contains no real customer data or observed deployment results.

The initial design stores a summary of every resolved ticket as long-term account memory. That makes later conversations easier to assemble, but it also copies incidental personal details into a store used for future tasks. The information may be unnecessary for technical support, difficult for the customer to inspect and accessible to a broader staff group than the original ticket.

The team changes the memory policy before introducing the feature. It permits a narrow set of technical preferences, such as an explicitly requested file format or supported integration version. It excludes free-form personal circumstances from persistent account memory. The ticket itself follows the organization's records policy; the agent does not decide that all source material should disappear simply because it should not become memory.

A memory-write step now produces a structured candidate with the permitted field, source reference and reason for reuse. Unrecognized fields are rejected. A reviewer can inspect

the candidate when the context is ambiguous. The system tests whether sensitive material in a ticket can bypass this step through a generated summary, an error log or a tool response. Filtering only the visible memory screen would miss those paths.

The retrieval design also separates account-scoped technical material from individual tickets with narrower access. A staff member assigned to another account cannot obtain either the original ticket or a derived summary through a similar query. The test uses synthetic accounts with distinct canary phrases, then checks candidate retrieval, assembled context and the final response.

The customer later changes the preferred file format. The authoritative preference updates, the old memory is marked superseded and future tasks use the current value. Historical transaction evidence may retain what was sent previously, but it is not treated as the current preference. This distinction prevents a correction from rewriting business history while still correcting future behavior.

Finally, the team exercises removal. It removes a test memory entry, runs the next indexing cycle and restores a controlled test backup. The expected result is that the entry does not silently return to active use. Any failure identifies the specific lifecycle path to fix. The exercise demonstrates the kind of evidence to collect; it does not prove that every real-world removal request will be complete.

Observe enough without collecting everything

Debugging an agent often creates pressure to capture complete prompts, responses and tool payloads. That can help reproduce a problem, but it can also create a secondary repository of protected information. Decide which questions routine telemetry must answer and reserve deeper capture for explicitly authorized investigations with narrower access and retention.

A useful baseline may include task type, component version, policy outcome, tool name, duration, error category and a reference to controlled evidence. Whether that is sufficient depends on the workflow. Do not assume that removing names makes a trace anonymous: unusual combinations of facts or linked identifiers can still reveal a person or organization.

NIST SP 800-53 integrates security and privacy controls, including audit and access-control considerations. [6] Use that control vocabulary to review who can view traces, export them, change collection settings and delete or preserve evidence. The presence of an observability platform does not establish that those controls are configured appropriately.

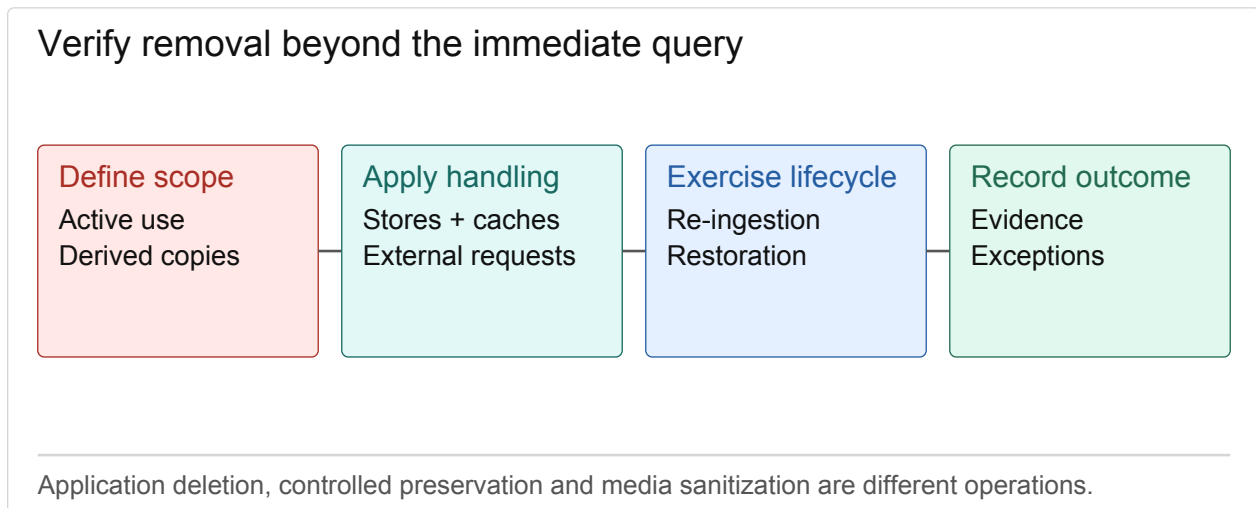


Figure 3. Verification includes scheduled jobs and restoration paths that could reintroduce information. Original Softlander conceptual diagram, 2026-09-16.

For incidents, preserve what is necessary to investigate while avoiding uncontrolled copying. An engineer should not need to paste a complete customer transcript into a general collaboration channel to explain a defect. Provide a sanitized summary and a controlled reference. If a vendor needs evidence, review the minimum data and authorized transfer route before sharing it.

Sampling is another design decision. A small sample of complete traces can still contain highly sensitive information, while a large sample of carefully chosen non-content events may answer the operational question. Evaluate usefulness and exposure together. Record what the sampling strategy cannot reveal, such as a rare failure that occurs only in a particular long-running context.

Include external processors in the map

An agent's information path may cross several suppliers: a model endpoint, a search service, a tracing platform and a support system. Record the actual configuration and agreed processing conditions for each. Do not infer that a setting in one service controls retention or secondary use in another. The architecture diagram should show these boundaries explicitly enough for procurement, privacy and engineering teams to ask the same questions.

For each transfer, identify the minimum payload, the receiving purpose, access controls, retention behavior and route for correction or removal. If an operational investigation requires a richer payload than routine processing, treat it as a separate handling decision. A general integration approval should not silently authorize every future troubleshooting export.

Where information about a supplier's handling is unavailable, preserve the uncertainty in the buying decision. Options include excluding the sensitive fields, using a different deployment arrangement, restricting the workflow or seeking clarification through the authorized commercial channel. Do not turn an unknown into an assurance statement merely because the supplier is well known. The appropriate boundary follows the actual processing, contract and configuration.

A lifecycle worksheet for the engineering review

For each store, answer eight questions: what information enters it; why it is needed; who owns that purpose; who may access it; how derived information retains restrictions; how it is corrected; when it leaves active use; and how the team verifies that behavior. Use concrete examples rather than "standard policy applies" when the implementation is uncertain.

Then walk one task across the stores. Include a successful run, a timeout, a retry, a support investigation and a later session. Ask whether the same handling rules still apply at each step. A temporary value can become persistent through an error path, and a tightly controlled source can become widely visible through a troubleshooting export.

Choose three tests that would most change the release decision. For a multi-tenant retrieval assistant, cross-account access may be the priority. For a personal memory feature, correction and reuse may matter most. For a regulated records workflow, preservation and removal boundaries may need specialist assessment. There is no useful universal score that replaces understanding the actual processing.

Record gaps as engineering work with observable completion conditions. "Improve privacy" is not a testable task. "Prevent a revoked user from retrieving an earlier cached result, including after a retry" is. Link the work to the relevant owner and evidence, and keep legal determinations with qualified advisers who understand the processing and jurisdictions involved.

A well-designed agent does not need to remember everything it encounters. It needs access to the right information for an authorized purpose, a clear distinction between current facts and historical evidence, and a lifecycle that remains understandable when the conversation is over. Start by tracing one piece of information through the system. The resulting map will usually reveal more useful design work than another general promise of privacy.

References

- [1] NIST. Privacy Framework, Version 1.0. January 2020; version explicitly cited here.
- [2] OWASP. LLM08:2025 Vector and Embedding Weaknesses. 2025 edition.
- [3] OWASP. LLM02:2025 Sensitive Information Disclosure. 2025 edition.
- [4] NIST. AI 600-1: Generative Artificial Intelligence Profile. July 2024.
- [5] NIST. SP 800-88 Revision 2: Guidelines for Media Sanitization. September 2025.
- [6] NIST. SP 800-53 Revision 5: Security and Privacy Controls. September 2020, with release updates listed by NIST.