Event ends in 17 hours 40 minutes.

Event dashboard > AgentCore Memory

AgentCore Memory

What is Amazon Bedrock AgentCore Memory?

AgentCore Memory lets your AI agents deliver intelligent, context-aware, and personalized interactions

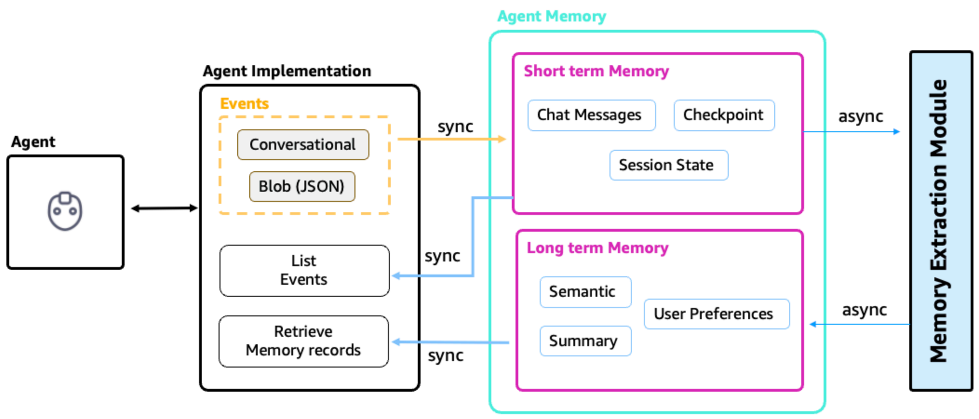
- **Short-term memory**: Stores conversations to keep track of immediate context.
- **Long-term memory**: Stores extracted insights - such as user preferences, semantic facts, and summaries - for knowledge retention across sessions.

Why AgentCore Memory?

Traditional AI agents face several critical challenges that AgentCore Memory addresses:

- **Infrastructure management**: AgentCore Memory provides a serverless solution with a single API that automatically manages the underlying infrastructure.
- **Context window limits**: AgentCore Memory persists the interaction history once and lets the agent selectively retrieve only what is relevant.
- **Cross-session continuity**: AgentCore Memory's long-term tier retains user preferences, semantic facts, and summaries so returning users experience continuity without repeating themselves.
- **Reliable consistency**: AgentCore Memory provides a managed schema (events and extracted records), asynchronous consolidation, namespace scoping, and role-based access controls out of the box.
- **Namespace isolation**: Enterprise deployments require per-user isolation, PII protection, and time-bound retention. AgentCore Memory supports hierarchical namespaces (`/strategy/{strategyId}/actor/{actorId}/session/{sessionId}`), encrypted storage with customer-managed KMS keys, event expiry settings up to 365 days, and IAM context keys for least-privilege policies.
- **Information processing and persistence**: Built-in and custom strategies intelligently capture main concepts from interactions and persists them. These strategy configurations help extract and persist relevant information without requiring custom parsing code, while supporting both immediate context and processed information across sessions.

Learn more about AgentCore Memory



Key Components

Memory Resource

A top-level logical container / resource that encapsulates configuration (name, description, encryption key, event retention period) and the set of extraction strategies. It's created only once to return a Unique Identifier **MemoryId** that will be used to store and retrieve - memories or raw events.

Short-Term Memory (Raw Events)

Short-term memories are immutable, time-stamped records of raw interactions/messages, or structured payloads (for Agent-state checkpointing) scoped by **actor ID** and **session ID**. It supports branching to represent edits or alternative paths. Learn more about short-term memory in AgentCore Memory

Long-Term Memory (Extracted Records)

Long-term memory stores information extracted from raw agent interactions, which is retained across multiple sessions. Instead of saving all raw conversation data, long-term memory preserves only the key insights such as summaries of the conversations, facts and knowledge (semantic memory), or user preferences. The records created are immutable; consolidation operations add, update or skip existing items to avoid duplication.

Learn more about long-term memory in AgentCore Memory

Memory Strategies

Extraction pipelines attached to a Memory resource.

- **Built-in**: `SemanticMemoryStrategy`, `SummaryMemoryStrategy`, `UserPreferenceMemoryStrategy`. Fully managed, no additional IAM policies required. Learn more about pricing in AgentCore Memory
- **Custom**: `CustomMemoryStrategy` where you supply a prompt, choose the model, and optionally override consolidation behaviour. Learn more about implementing memory strategies in AgentCore Memory

Namespaces

Logical partitions defining where records live. Support variable substitution (`{actorId}`, `{sessionId}`, `{strategyId}`) for automated organisation and IAM scoping.

Encryption Configuration

Data encrypted at rest with either AWS-managed keys or a customer-managed KMS key referenced during memory creation.

How it Works (General Flow)

1. **Memory Creation** - The builder invokes `CreateMemory`, supplying optional strategies, KMS key, retention period, and description. The control plane provisions the store. If no strategy is provided, only short-term memory is available (milliseconds). When long-term strategies are specified, they enter an `ACTIVE` state after deployment (≈ 2–3 min). Learn more about memory creation in AgentCore Memory
2. **Event Ingestion** - During a live conversation the AI agent calls `CreateEvent`, passing actor ID, session ID, and payload. Events are committed synchronously, guaranteeing durability before the agent responds.
3. **Asynchronous Extraction** - A background service batches recent events and applies each configured strategy (only when a long-term strategy is specified):

   - **Extraction** generates candidate records via the chosen model.
   - **Consolidation** adds, updates, or skips based on the strategy's consolidation prompt, preventing duplication and contradiction.
4. **Retrieval and Semantic Search** - At inference time, the AI agent invokes `ListEvents` to reconstruct recent context and `RetrieveMemoryRecords` to pull long-term knowledge relevant to a query. The semantic search operates within the specified namespace, and can be filtered by strategy ID. Learn more about long-term memory retrieval in AgentCore Memory
5. **Governance and Lifecycle**

   - Raw events age out automatically per `eventExpiryDuration`.
   - Developers can delete specific events (`DeleteEvent`) or records (`DeleteMemoryRecord`) for right-to-be-forgotten compliance.
   - Updating a Memory (`UpdateMemory`) allows adding or disabling strategies without data loss; new records appear only for events stored after the strategy becomes active.
   - The entire Memory can be permanently removed via `DeleteMemory`.

Labs:

In the following labs, you'll learn:

Lab: Short-Term Memory Fundamentals

- Implement short-term memory for conversation context
- Handle multi-turn interactions with context preservation
- Explore event storage and retrieval patterns
- Work with actor and session based scoping

Lab: Long-Term Memory Fundamentals

- Configure long-term memory strategies
- Use memory hooks for automatic knowledge extraction
- Learn about hierarchical namespace organization
- Handle complex multi-session workflows

Previous Next