Samvid Memory Studio - User Manual

Revision History

Revision	Date	Author	Description
1.0	June 2026	Anand Dey	Initial release

Table of Contents

About This Manual

Who Should Read This Manual

This manual is written for end users of Samvid Memory Studio. It assumes basic familiarity with using a computer, a web browser, and a terminal (command prompt).

The chapters are structured so you can read them in order or jump directly to the topic you need:

Chapter	Title	Intended For
1	Introduction	All users — understand what Samvid does and why
2	Installation	All users — get Samvid running on your machine
3	The Desktop Application	All users — learn the visual interface
4	Command-Line Interface (CLI)	Power users and developers — script and automate memory operations
5	MCP Server — Connecting AI Agents	AI tool users — connect Samvid to your AI assistant
6	Memory Types	All users — reference for choosing memory categories
7	Knowledge Graph — Associations	All users — reference for linking memories together
8	Lifecycle Management	All users — keep your knowledge base clean and relevant
9	Troubleshooting and Support	All users — resolve common problems
10	Glossary	All users — definitions of key terms

Conventions Used in This Manual

The following typographic conventions are used throughout this document:

Convention	Appearance	Meaning
Bold text	**Bold**	User interface elements, button names, page names, and menu items
`Monospace`	`samvid search "query"`	Commands typed in a terminal, file paths, code snippets, parameter names, and tool names
Italic text	*first time*	Emphasis or the first occurrence of a key term

Command examples are shown as code blocks. The `$` at the beginning of a line represents the terminal prompt and should not be typed:

```
$ samvid stats
```

Where platform-specific commands differ, separate examples are shown with labels:

```
Windows:   samvid-mcp.bat --config cursor
  Linux/macOS:   ./samvid-mcp.sh --config cursor
```

Callout Boxes

Throughout this manual, the following callout styles highlight important information:

> ***NOTE***
> *Supplementary information that is helpful but not critical.*

> ***TIP***
> *A recommendation that can save time or improve results.*

> ***IMPORTANT***
> *Information that must be read to avoid mistakes or data loss.*

> ***WARNING***
> *Critical information about actions that could cause permanent problems if not followed carefully.*

---

1. Introduction

1.1 What is Samvid Memory Studio?

Samvid Memory Studio is a local memory system for your computer. It stores, organizes, and retrieves information using **semantic search** — meaning it understands the *idea* behind your words, not just the exact words themselves.

Samvid runs entirely on your machine. No data ever leaves your computer. No internet connection is required for normal operation.

Samvid provides three ways to interact with your memory store:

1. **A Desktop Application** — a cross-platform windowed application for browsing, searching, adding, and managing your memories visually.

2. **A Command-Line Interface (CLI)** — type commands in a terminal to add, search, update, or manage memories. Outputs JSON for scripting and automation.
3. **An MCP Server** — a local service that AI assistants (cursor, Claude Desktop, VS Code, Windsurf, Codex, and others) can connect to. This gives AI agents persistent memory across sessions.

All three interfaces read and write to the same database on your machine. A memory added through the desktop app is instantly available to your AI assistant. A fact stored by an AI agent appears in the desktop dashboard.

1.2 Why Use Samvid?

AI assistants today are powerful but forgetful. They start every conversation with no memory of who you are, what you decided, or what they learned in previous sessions. You must re-explain everything each time.

Samvid solves this by giving any AI agent a persistent, local memory. You also get a visual application to browse and manage everything your AIs have stored.

Key benefits:

- **AI agents remember across sessions.** Preferences, project decisions, research notes — everything persists.
- **Multiple AI tools can share one memory.** Connect Cursor, Claude Desktop, and VS Code to the same Samvid instance. They all know the same information.
- **Your data stays private.** Everything runs locally. No cloud, no accounts, no API keys.
- **No recurring costs.** Samvid is a single installation with no subscription fees.

1.3 Who Is It For?

User	How They Use Samvid
AI tool users	Connect Samvid to your AI editor or assistant. Every AI tool you use now shares a persistent, shared memory.
Developers and power users	Use the CLI for scripting and automation. Integrate Samvid into build pipelines, cron jobs, and custom workflows.
Researchers and writers	Use the desktop app as a searchable knowledge base. Add notes, papers, and ideas. Search semantically later.
Anyone who uses AI regularly	Stop re-explaining yourself. Your AI remembers your preferences, project context, and past decisions.

1.4 How the Pieces Fit Together

The typical workflow:

1. **Install** Samvid using the platform installer.
2. **Launch the desktop app.** It opens the dashboard and automatically starts the MCP server.

3. **Add memories** through the app — facts about yourself, project notes, preferences.
4. **Connect your AI agent** by adding a short configuration snippet to your AI tool's settings. Restart the AI tool.
5. **Your AI now has memory.** It can store what it learns, recall what it knows, and build context from past memories.
6. **Monitor** your growing knowledge base through the dashboard. Run periodic cleanup operations to keep it organized.

---

2. Installation

Samvid Memory Studio is distributed as a single installer for each platform. The installer bundles everything: the desktop application, the MCP server, the CLI, and the embedding AI. No separate dependency setup is required.

2.1 Windows

1. Download the Windows installer (`Samvid Memory Studio-1.0.0-win-x64.exe`) from the GitHub releases page.
2. Double-click the installer file.
3. Follow the on-screen instructions. You can choose a custom installation directory.
4. After installation, the desktop app launches automatically.

2.2 macOS

1. Download the macOS disk image (`Samvid Memory Studio-1.0.0-arm64.dmg` for Apple Silicon, or `...-x64.dmg` for Intel).
2. Open the DMG file by double-clicking it.
3. Drag the "Samvid Memory Studio" icon into the "Applications" folder.
4. Launch the app from your Applications folder or Launchpad.

The first time you open the app, macOS may show a security warning because the application is not notarized and is normal. To proceed, go to **System Settings → Privacy & Security** and click "Open Anyway."

2.3 Linux

1. Download the appropriate package for your distribution:
   - **Debian / Ubuntu:** `.deb` package
   - **Other distributions:** `.AppImage` file
2. For the `.deb` package, open a terminal and run:

```
sudo dpkg -i Samvid-Memory-Studio-1.0.0-amd64.deb
```

1. For the `.AppImage`, make it executable and run:

```
chmod +x Samvid-Memory-Studio-1.0.0-x86_64.AppImage
  ./Samvid-Memory-Studio-1.0.0-x86_64.AppImage
```

1. Launch from your applications menu or run `samvid` in a terminal.

2.4 Verifying Installation

Open a terminal and run:

```
samvid stats
```

You should see a JSON output showing database statistics (initially zeros or empty values). This confirms the CLI is working.

To verify the MCP server can generate configuration snippets (without starting the server), run:

Windows:

```
samvid-mcp.bat --config cursor
```

macOS / Linux:

```
./samvid-mcp.sh --config cursor
```

This prints a JSON configuration snippet and exits. If you see the JSON output, the MCP server component is ready to use.

2.5 Where Your Data Is Stored

All your memory data lives in a single file called `samvid.db`. Its default location depends on your operating system:

Platform	Database Location
Windows	`%APPDATA%\Samvid Memory Studio\samvid.db`
macOS	`~/Library/Application Support/Samvid Memory Studio/samvid.db`
Linux	`~/.config/Samvid Memory Studio/samvid.db`

Note: Uninstalling the desktop application does not delete this file. Your memories remain on disk. Reinstalling later picks up your existing data automatically.

> ***IMPORTANT***
> *To permanently erase all your data, you must manually delete the database file from the location listed above.*

---

3. The Desktop Application

3.1 Overview

The desktop application is a cross-platform windowed interface with a sidebar navigation. It provides nine pages that cover all aspects of managing your memory store.

When the app launches, it automatically starts the MCP server. Any connected AI agents can immediately begin using the memory tools.

The app can be minimized to the system tray. Closing the window does not quit the application — the MCP server continues running. To fully exit, use the **Quit** option in the system tray menu.

The app supports dark and light visual themes.

> ***TIP***
> *Minimize the app to the system tray after setup. The MCP server stays running and your AI agents stay connected, even when the window is closed.*

3.2 Dashboard

The Dashboard is the home screen. It shows at-a-glance statistics about your memory store:

- **Total Memories** — the total number of stored items
- **Associations** — how many typed links exist between memories
- **DB Size** — disk space used by the database
- **Embedding Dim** — always 1024 (the dimension of the vector representation)
- **Avg Importance** — the average importance score across all memories
- **Expired** — memories ready for removal (TTL expired)
- **Total Searches** — cumulative search count
- **Avg Score** — average relevance score of search results

Three charts are displayed:

- **Namespace Density** — a pie chart showing how memories are distributed across namespaces
- **Importance Distribution** — an area chart showing how importance scores are spread
- **Daily Memory Activity** — a bar chart of daily creation and access counts

When no real distribution data is available yet, the charts display placeholder data.

3.3 Memories

The Memories page is the main working surface for browsing and managing individual memories.

To search for memories:

1. Type your query in the search box at the top of the page.
2. Select a search mode: **Semantic** (meaning-based) or **Hybrid** (combined meaning + keyword). Hybrid is recommended for general use.
3. Enable **Adaptive** mode to let the system adjust the threshold automatically, or use the slider to set a fixed threshold.
4. Press **Enter** or click the **Search** button.
5. Results appear below, ranked by relevance score.

> ***TIP***
> *Keep Adaptive mode enabled. It automatically lowers the threshold until useful results are found, so you do not need to fine-tune the slider.*

To browse memories:

1. Use the pagination controls at the bottom of the list to move between pages.
2. Select a page size from the dropdown: 25, 50, 100, or 200 memories per page.
3. Optionally, use the filters above the list to narrow results by namespace or memory type.

To add a memory:

1. Locate the Add Memory option at the top of the page.
2. Enter the text content of your memory.
3. Enter the **Namespace** name (keep "default" or type a new name to create one).
4. Select a **Memory Type** from the dropdown.
5. Set the **Importance** using the number field (0.0 = trivial, 1.0 = critical).
6. Click **Save Memory**. The new memory appears at the top of the list.

To view a memory:

1. Click any memory row in the list to open its detail panel.
2. The panel displays: full content, type label, namespace, importance bar, creation date, last updated date, access count, TTL, tags, and any raw metadata.
3. Press the **Escape** key, click the **X** button, or click outside the panel to close it.

> ***NOTE***
> *The detail panel is read-only. To modify a memory, use the `samvid update` CLI command (see Section 4.2.6) or the `update_memory` MCP tool (available to connected AI agents).*

To delete a memory:

1. Click the delete icon on the memory row.
2. The memory is removed from the list immediately. If the operation fails on the server, it is automatically restored.

> ***IMPORTANT***
> *Deleting a memory is permanent. Its vector embedding, search index, and all associations to other memories are also removed. Review before deleting.*

3.4 Sessions

The Sessions page groups memories by their session ID, showing episodic chains.

To work with sessions:

1. Each row in the table represents one session, showing its ID and the number of memories it contains.
2. Click a session row to expand it and view all memories inside, listed in chronological order.
3. Click the **Replay** button to open a modal window showing the entire session reconstructed as a formatted, chronological transcript — suitable for reading or copying.

3.5 Lifecycle

The Lifecycle page provides three maintenance operations for keeping your knowledge base healthy.

> ***IMPORTANT***
> *Before applying any lifecycle operation, review the preview results. Use the **Dry Run** toggle for Consolidate to preview merges without making changes.*

Operation 1 — Compact:

1. Click the **Compact** button.
2. The system removes all memories whose TTL has expired.
3. A database defragmentation runs automatically to reclaim disk space.
4. Review the result showing how much space was reclaimed.

Operation 2 — Consolidate:

1. Set the **Cosine Distance Threshold** (default: 0.15). Lower values mean only very similar memories are merged.
2. Enable the **Dry Run** toggle to preview merges without applying them.
3. Click **Consolidate**.
4. If Dry Run is enabled, review the candidate pairs shown in the results.
5. Disable Dry Run and click **Consolidate** again to apply the merges.

> ***TIP***
> *Always run with Dry Run enabled first. Review the proposed merges before applying them permanently.*

Operation 3 — Forget:

1. Set the **Importance Threshold** (default: 0.1). Memories with effective importance below this value are deleted.
2. Optionally, enter a **Namespace** to limit the operation to one group.
3. Click **Forget**.
4. Review the count of deleted memories in the result.

> ***NOTE***
> *Memories with a base importance of 0.9 or higher are always preserved, regardless of their effective importance.*

3.6 Summarization

The Summarization page provides a three-step pipeline to compress many raw memories into concise summaries.

Step 1 — Prepare Batches:

1. Type the **Namespace** name you want to summarize.
2. Set the **Memory Threshold** (default: 100) — the minimum number of raw memories needed before summarization is triggered.
3. Set the **Batch Size** (default: 20) — how many memories go into each summarization batch.
4. Click **Prepare Batches**.
5. If the namespace has enough memories, batches appear below the button.

Step 2 — Review Batches:

1. Click a batch row to expand it and see the memories inside.
2. Click the copy button next to a batch to copy all the memory texts to your clipboard.

Step 3 — Store Summary:

1. Paste the batch text into any AI assistant (Claude, ChatGPT, etc.) and ask it to summarize.
2. Copy the generated summary text.
3. Paste the summary into the text field on the Samvid page.
4. Click **Store Summary**.

> ***NOTE***
> *When you store a summary, the source memories are automatically downgraded: their importance is reduced to 30% of its original value and their type is changed to "summarized."*

3.7 Namespaces

The Namespaces page displays a table of all namespaces in your database, sorted by memory count — most populous first. Each namespace name is color-coded by its rank.

This page is read-only. Namespaces are created automatically when you add a memory and type a new namespace name.

3.8 Export

The Export page lets you save your memories to a JSON file on your computer.

To export memories:

1. Optionally, type a **Namespace** name in the input field to export only that namespace. Leave the field blank to export all namespaces.
2. Click the **Export** button.
3. A native file save dialog opens. Navigate to your desired location and enter a filename.
4. Click **Save**.
5. A confirmation message shows the number of memories exported and the file path.

> ***NOTE***
> *The exported file includes all memories and their associations in a versioned JSON format. Import is available through the CLI and MCP server tools. The desktop application itself does not provide an import page.*

3.9 MCP Server Status

The MCP Server page shows the state of the background MCP service.

Status Indicators:

- **Connection Status** — Connected, Disconnected, Reconnecting, or Failed
- **Port** — the HTTP port (default: 8258)
- **Server Name** — the name sent during MCP handshake
- **Tools Registered** — the count of MCP tools currently available
- **MCP Endpoint** — the address agents connect to
- **Session ID** — the current MCP session identifier

Server Controls:

1. **Stop Server** — click to disconnect the MCP client. Visible only when connected.
2. **Start Server** — click to reconnect. Visible only when disconnected or in a failed state.
3. **Restart** — click to disconnect and reconnect after a short delay.

Registered Tools:

- Displayed as a grid of labeled badges, showing every tool currently registered by the server.

Client Configuration:

- The page includes ready-to-paste configuration snippets for four MCP client categories:
  - **Codex** (TOML format)
  - **Cursor / Claude / Windsurf** (JSON with `mcpServers` key)

- **VS Code** (JSON with `servers` key)
- **OpenClaw / Hermes Agent** (YAML format)

- Click a client tab to view its snippet. Each snippet shows the config file location and includes a copy-to-clipboard button.

> ***TIP***
> *Use this page's client configuration snippets instead of typing them manually. Select your client tab, click the copy button, and paste directly into your AI tool's MCP config file.*

3.10 Settings

The Settings page has two sections.

Startup Preferences:

1. Toggle the **Auto-launch at login** switch on or off.
2. When enabled, the app starts automatically when you log into your computer.

Check for Updates:

1. Click **Check for Updates** to query the update server.
2. If a newer version is available, the app begins downloading it. A progress bar shows the download speed.
3. After the download completes, click **Install & Restart** to apply the update.

---

4. Command-Line Interface (CLI)

4.1 General Usage

The CLI is available as the `samvid` command (on all platforms). It provides 18 subcommands for memory operations. Every command outputs **JSON** with at least a `"success"` field (`true` or `false`). This makes the CLI suitable for scripting and automation.

Memories are identified by UUIDs (e.g., `"a1b2c3d4-e5f6-7890-abcd-ef1234567890"`). You obtain IDs from the output of `add`, `search`, `list`, or `get` commands.

4.2 Command Reference

4.2.1 `add` — Add a Memory

Stores a new piece of information. The content is automatically converted into a vector embedding for future semantic search.

```
samvid add "Your memory text here"
```

Parameters:

Parameter	Description	Default
`content`	The text to store (positional; can be omitted if using `--file`)	—
`--file`	Read content from a file path instead of typing inline	—
`--type`	Memory type	`general`
`--importance`	How important (0.0 = trivial, 1.0 = critical)	`0.5`
`--namespace`	Group/folder name	`default`
`--tags`	Space-separated keywords	—
`--source`	Where this information came from	—
`--metadata`	JSON object with extra data	—
`--session-id`	Session ID to group related memories	—
`--prev`	Previous memory ID for chain linking	—
`--decay-rate`	How fast importance fades (higher = faster)	`0.01`
`--ttl`	Days before automatic expiration	—

Examples:
Add a simple fact:

```
samvid add "The user prefers dark mode" --type preference --importance 0.9
```

Add from a file:

```
samvid add --file meeting-notes.txt --type event --namespace work
```

Add with tags:

```
samvid add "Postgres selected for payments" --type fact --tags database decision
```

Add a self-expiring memory:

```
samvid add "Temporary sprint note" --ttl 7
```

Add linked into a session chain:

```
samvid add "Step two: write migrations" --session-id ses_abc --prev mem_xyz
```

4.2.2 `search` — Semantic Search

Searches by meaning, not exact words. Returns results ranked by relevance score.

```
samvid search "your query"
```

Parameters:

Parameter	Description	Default
`query`	The search text (positional, required)	—
`--namespace`	Limit to one namespace	All
`--type`	Limit to one memory type	All
`--limit`	Maximum results	`10`
`--threshold`	Minimum relevance score (0.0–1.0)	`0.15`
`--progressive`	Auto-relax threshold if no results (default)	Enabled
`--no-progressive`	Use strict threshold only	—

Examples:

```
samvid search "what database do we use"
  samvid search "auth infrastructure" --namespace project --limit 5
  samvid search "api design" --no-progressive --threshold 0.25
```

Progressive search (enabled by default): If no results are found at the given threshold, the system automatically tries lower thresholds (stepping from the given value down to 0.10 in increments of 0.05) until results are found. This means you get useful output without having to fine-tune the threshold parameter. Disable with `--no-progressive` to enforce a strict cutoff.

4.2.3 `hybrid-search` — Combined Search

Combines semantic (meaning-based) search with keyword (exact word) search using Reciprocal Rank Fusion. This is the recommended search mode for most queries.

```
samvid hybrid-search "your query"
```

Parameters:

Parameter	Description	Default
`query`	The search text (positional, required)	—
`--namespace`	Limit to one namespace	All
`--limit`	Maximum results	`10`

Example:

```
samvid hybrid-search "postgres migration"
  samvid hybrid-search "deployment steps" --namespace work --limit 5
```

4.2.4 `list` — List Memories

Displays memories as a paginated list with optional filters.

```
samvid list
```

Parameters:

Parameter	Description	Default
`--namespace`	Only from this namespace	All
`--type`	Only of this type	All
`--limit`	Maximum per page	`50`
`--offset`	Skip this many (for pagination)	`0`

Examples:

```
samvid list --namespace work
  samvid list --type fact --limit 20
  samvid list --limit 10 --offset 30
```

4.2.5 `get` — Retrieve a Memory

Fetches a single memory by its UUID. The access count and last-accessed timestamp are updated.

```
samvid get <memory-id>
```

Example:

```
samvid get a1b2c3d4-e5f6-7890-abcd-ef1234567890
```

4.2.6 `update` — Modify a Memory

Updates one or more fields of an existing memory. If content is changed, the vector embedding is regenerated automatically.

```
samvid update <memory-id> --content "New text" --importance 0.8
```

Parameters:

Parameter	Description
`id`	The memory UUID (positional, required)
`--content`	New text content
`--importance`	New importance (0.0–1.0)
`--type`	New memory type
`--tags`	New space-separated tags (replaces existing)
`--metadata`	New JSON metadata (replaces existing)

Examples:

```
samvid update a1b2c3d4... --importance 0.9
  samvid update a1b2c3d4... --content "Revised project architecture" --type plan
  samvid update a1b2c3d4... --tags urgent priority-high
```

4.2.7 `delete` — Remove a Memory

Permanently deletes a memory and all associated data — its vector embedding, full-text index entries, and any associations.

```
samvid delete <memory-id>
```

Example:

```
samvid delete a1b2c3d4-e5f6-7890-abcd-ef1234567890
```

4.2.8 `context` — Build AI Context

Searches semantically for the most relevant memories and formats them into a clean text block suitable for pasting into an AI conversation as background context.

```
samvid context "your query" --limit 5
```

Parameters:

Parameter	Description	Default
`query`	What to build context about (positional, required)	—
`--limit`	Number of memories to include	`5`
`--namespace`	Search within one namespace	All

Examples:

```
samvid context "database decisions for payments" --limit 5
  samvid context "auth approach" --namespace project
```

4.2.9 `associate` — Link Two Memories

Creates a typed relationship between two memories to build a knowledge graph.

```
samvid associate <source-id> <target-id> --relation supports --strength 0.8
```

Parameters:

Parameter	Description	Default
`source`	Source memory UUID (positional, required)	—
`target`	Target memory UUID (positional, required)	—
`--relation`	Relationship type	`related_to`
`--strength`	Confidence strength (0.0–1.0)	`1.0`

The CLI supports four relation types: `supports`, `contradicts`, `caused_by`, `related_to`. (The MCP server supports all 13 relation types — see Section 7.)

Examples:

```
samvid associate mem_a mem_b --relation supports --strength 0.9
  samvid associate new_plan old_plan --relation supersedes --strength 1.0
  samvid associate bug_report bug_root --relation caused_by --strength 0.85
```

4.2.10 `associations` — View Memory Links

Lists all outgoing associations from a memory to other memories.

```
samvid associations <memory-id>
```

Example:

```
samvid associations a1b2c3d4-e5f6-7890-abcd-ef1234567890
```

4.2.11 `replay` — Replay a Session

Reconstructs all memories belonging to an episodic session as a formatted chronological transcript.

```
samvid replay <session-id>
```

Example:

```
samvid replay ses_7f3a9b
```

4.2.12 `summarize-prep` — Prepare for Summarization

Checks if a namespace has enough non-summary memories to trigger summarization, and groups them into batches for processing by an AI.

```
samvid summarize-prep --namespace work
```

Parameters:

Parameter	Description	Default
`--namespace`	Namespace to check	`default`
`--threshold`	Minimum raw memory count before summarization is suggested	`100`
`--batch-size`	Memories per summarization batch	`20`

Example:

```
samvid summarize-prep --namespace research --threshold 50 --batch-size 10
```

4.2.13 `store-summary` — Store a Summary

Stores a summary memory and automatically downgrades the source memories (importance reduced to 30%, type changed to "summarized").

```
samvid store-summary "Summary text" <source-id-1> <source-id-2> ...
```

Parameters:

Parameter	Description	Default
`content`	The summary text (positional, required)	—
`sources`	One or more source memory UUIDs (positional, required)	—
`--namespace`	Namespace for the summary	`default`
`--importance`	Importance of the summary	`1.0`

Example:

```
samvid store-summary "Auth decisions: JWT with refresh tokens in httpOnly cookies" abc1
  23 def456 --namespace project --importance 1.0
```

4.2.14 `consolidate` — Merge Similar Memories

Finds semantically similar memories (near-duplicates) and merges them. Always preview first with `--dry-run`.

> ***IMPORTANT***
> *Always run with `--dry-run` first. Review the proposed merges carefully before applying them permanently.*

```
samvid consolidate --dry-run
```

Parameters:

Parameter	Description	Default
`--namespace`	Namespace to consolidate	All
`--threshold`	Cosine distance threshold for similarity (0 = identical, 2 = completely different)	`0.15`
`--dry-run`	Preview merges without applying	Off

Examples:
Preview:

```
samvid consolidate --dry-run --namespace work
```

Apply after review:

```
samvid consolidate --namespace work
```

4.2.15 `forget` — Remove Stale Memories

Deletes memories whose effective importance (after decay over time) has dropped below a threshold. Memories with base importance of 0.9 or higher are always preserved.

```
samvid forget --threshold 0.1
```

Parameters:

Parameter	Description	Default
`--namespace`	Limit to one namespace	All
`--threshold`	Effective importance below this value gets deleted	`0.1`

Examples:

```
samvid forget
  samvid forget --namespace personal --threshold 0.15
```

4.2.16 `compact` — Clean Up Database

Removes memories whose TTL has expired and runs database defragmentation (VACUUM) to reclaim disk space.

```
samvid compact
```

No parameters required.

4.2.17 `export` — Export to JSON

Saves memories (and their associations) to a JSON file.

```
samvid export <file-path>
```

Parameters:

Parameter	Description
`output`	Destination file path (positional, required)
`--namespace`	Export only this namespace (optional)

Examples:

```
samvid export ./backup.json
  samvid export ./work-only.json --namespace work
```

4.2.18 `import` — Import from JSON

Loads memories from a previously exported JSON file. Vector embeddings are regenerated. Duplicate memories (by UUID) are skipped by default.

```
samvid import <file-path>
```

Parameters:

Parameter	Description
`input`	Source JSON file path (positional, required)
`--force`	Do not skip duplicates

Examples:

```
samvid import ./backup.json
  samvid import ./backup.json --force
```

4.2.19 `stats` — Database Statistics

Displays comprehensive analytics about your database.

```
samvid stats
```

Output includes: total memory count, association count, database file size, per-namespace breakdown, importance distribution, average importance, average relevance score, total searches, expired memory count, and other metrics.

---

5. MCP Server — Connecting AI Agents

5.1 What is the MCP Server?

The MCP (Model Context Protocol) server is a local HTTP service that exposes Samvid's memory operations as 20 tools for AI agents. Any MCP-compatible AI client can connect to it.

The server listens on `http://127.0.0.1:8258/mcp` by default.

The desktop application automatically starts the MCP server when it launches. You do not need to start it separately if you are using the desktop app.

> ***IMPORTANT***
> *The MCP server must be running before any AI agent can use Samvid's tools. If you are using the desktop application, it starts automatically. If running the server standalone, keep the terminal window open — closing it stops the server.*

5.2 Starting the Server

If you need to start the server without the desktop app:

Windows:

```
samvid-mcp.bat
```

macOS / Linux:

```
./samvid-mcp.sh
```

The server starts and prints:

```
Samvid Memory Studio MCP Server starting (transport=http, db=samvid.db, dimension=1024)
  MCP endpoint (Streamable HTTP): http://127.0.0.1:8258/mcp
  Press Ctrl+C to stop
```

It runs until you press `Ctrl+C` or close the terminal.

The server uses **Streamable HTTP** transport by default (the current MCP standard). It also supports SSE (legacy) and stdio transports.

5.3 Server Command Options

Flag	Description	Default
`--port <number>`	HTTP port	`8258`
`--host <address>`	Bind address	`127.0.0.1`
`--endpoint <path>`	HTTP endpoint path	`/mcp`
`--db <path>`	Database file path	`./samvid.db`
`--model <path>`	Model directory	`./models/embedder`
`--sse`	Use legacy SSE transport	—
`--stdio`	Use stdio transport	—
`--config <client>`	Print config snippet for a client and exit	—
`--help`	Show all options	—

Examples:

```
samvid-mcp.bat --port 9000
  ./samvid-mcp.sh --host 0.0.0.0
  ./samvid-mcp.sh --db /home/user/my-db.db
```

5.4 Connecting Specific AI Clients

Each AI client needs a small configuration snippet so it knows how to reach the Samvid MCP server. The table below shows where each client stores its MCP configuration:

AI Client	Config File Location
Cursor	`.cursor/mcp.json` (project root)
Claude Desktop	`%APPDATA%\Claude\claude_desktop_config.json` (Windows)
	`~/Library/Application Support/Claude/claude_desktop_config.json` (macOS)
	`~/.config/Claude/claude_desktop_config.json` (Linux)
VS Code	`.vscode/mcp.json` (project root)
Windsurf	`~/.codeium/windsurf/mcp_config.json`
Codex	`~/.codex/config.toml`
OpenCode	`opencode.json` (project root) or `~/.config/opencode/opencode.json`
OpenClaw	OpenClaw configuration (varies)
Hermes Agent	`~/.hermes/config.yaml`

The configuration snippets assume the server runs on the default port (8258) at localhost (127.0.0.1).

5.4.1 Cursor IDE

Add to `.cursor/mcp.json`:

```
{
    "mcpServers": {
      "samvid": {
        "type": "http",
        "url": "http://127.0.0.1:8258/mcp"
      }
    }
  }
```

Then restart Cursor.

5.4.2 Claude Desktop

Add to the Claude Desktop config file (see table above):

```
{
    "mcpServers": {
      "samvid": {
        "type": "http",
        "url": "http://127.0.0.1:8258/mcp"
      }
    }
  }
```

Then restart Claude Desktop.

5.4.3 VS Code with Copilot

Add to `.vscode/mcp.json`:

```
{
    "servers": {
      "samvid": {
        "type": "http",
        "url": "http://127.0.0.1:8258/mcp"
      }
    }
  }
```

Then restart VS Code.

5.4.4 Windsurf

Add to `~/.codeium/windsurf/mcp_config.json`:

```
{
    "mcpServers": {
      "samvid": {
        "type": "http",
        "url": "http://127.0.0.1:8258/mcp"
      }
    }
  }
```

Then restart Windsurf.

5.4.5 Codex

Add to `~/.codex/config.toml`:

```
[mcp_servers.samvid]
  url = "http://127.0.0.1:8258/mcp"
```

Then restart Codex.

5.4.6 OpenCode

Add to `opencode.json` (project root or global config):

```
{
    "mcp": {
      "samvid": {
        "type": "http",
        "url": "http://127.0.0.1:8258/mcp"
      }
    }
  }
```

Then restart OpenCode.

5.4.7 OpenClaw

Add to your OpenClaw configuration:

```
mcp:
    servers:
      samvid:
        url: "http://127.0.0.1:8258/mcp"
```

Then restart OpenClaw.

5.4.8 Hermes Agent

Add to `~/.hermes/config.yaml`:

```
mcp_servers:
    samvid:
      url: "http://127.0.0.1:8258/mcp"
```

Then restart Hermes.

5.5 Generating Config Snippets Automatically

Instead of writing the configuration manually, you can ask the server to print the exact snippet:

```
samvid-mcp.bat --config cursor        (Windows)
  ./samvid-mcp.sh --config claude       (macOS / Linux)
  ./samvid-mcp.sh --config vscode
  ./samvid-mcp.sh --config windsurf
  ./samvid-mcp.sh --config codex
  ./samvid-mcp.sh --config opencode
  ./samvid-mcp.sh --config openclaw
  ./samvid-mcp.sh --config hermes
```

To see all configurations at once:

```
./samvid-mcp.sh --config all
```

This prints the snippets and exits — the server does not start.

5.6 Legacy SSE Transport

Some older MCP clients do not support the modern Streamable HTTP transport. For these, use SSE mode:

> ***NOTE***
> *SSE is a legacy transport. Use Streamable HTTP (the default) unless your client explicitly requires SSE.*

```
samvid-mcp.bat --sse
```

The endpoint becomes `http://127.0.0.1:8258/samvid/sse`.

Generate SSE config snippets:

```
samvid-mcp.bat --sse --config cursor
```

In SSE mode, the `type` field in the generated config changes to `"remote"` (or `"sse"` for VS Code), and the URL changes to the SSE endpoint.

5.7 MCP Troubleshooting

"The AI agent does not see Samvid tools"

- Verify the MCP server is running. Check that `samvid stats` works from the terminal.
- Confirm the configuration file is in the correct location for your client (see table in Section 5.4).
- Verify the URL in the configuration matches exactly: `http://127.0.0.1:8258/mcp`.
- Restart the AI client after adding the configuration.

"Port 8258 already in use"
Start the server on a different port:

```
samvid-mcp.bat --port 9000
```

Then update the URL in your client's configuration to `http://127.0.0.1:9000/mcp`. The `--config` flag with the custom port does not automatically include the port change in the output, so edit the URL in the snippet manually.

"Client doesn't support Streamable HTTP"

Use legacy SSE mode (see Section 5.6). Start the server with `--sse` and update the URL and `type` field in your config accordingly.

---

6. Memory Types

Every memory has a type that indicates what kind of information it represents. Choosing the right type improves search quality and helps AI agents reason about the knowledge.

Type	Purpose	Example
fact	Verifiable information	"Postgres is the primary database"
event	Something that happened at a point in time	"User requested auth refactor on June 5"
preference	A user or agent preference	"Prefers concise responses"
procedure	Step-by-step instructions	"To deploy: run tests, build, push, restart"
context	Environmental or situational information	"Currently working on the auth migration"
general	Catch-all for unstructured information	Any general note
summary	A condensed version of multiple memories	"Auth decisions summary: JWT, key rotation"
summarized	A memory that has been condensed into a summary	(Set by the summarization pipeline)
entity	A thing, person, or object	"Stripe", "Alice Chen"
goal	An objective to work toward	"Ship v2 by end of quarter"
plan	Concrete steps toward a goal	"Step 1: refactor auth, Step 2: add tests"
rule	A hard constraint that must be followed	"Never commit directly to main branch"
heuristic	A soft guideline from experience	"When the user says 'quick', they mean one-liner"
feedback	A correction or rating from the user	"That answer was too long, be more concise"
critique	Self-critique or review	"This approach scales poorly past 10k records"
concept	Abstract idea or definition	"Eventual consistency means..."

---

7. Knowledge Graph — Associations

Memories can be linked together with typed relationships. These associations form a knowledge graph that lets AI agents trace connections between facts, track the evolution of decisions, and detect conflicts.

All 13 relationship types (available via MCP tools):

Relation	Meaning	Example
`supports`	A corroborates B	"Test passes" `supports` "Code is correct"
`contradicts`	A conflicts with B	"Error log" `contradicts` "System is healthy"
`caused_by`	A was caused by B	"Bug fix" `caused_by` "Bug report"
`related_to`	Generic connection	Catch-all link
`is_a`	A is a type of B	"Postgres" `is_a` "database"
`instance_of`	A is an instance of B	"Our prod DB" `instance_of` "Postgres"
`part_of`	A is a component of B	"Auth module" `part_of` "System"
`contains`	A contains B	"System" `contains` "Auth module"
`supersedes`	A replaces B	"New plan" `supersedes` "Old plan"
`implies`	A logically implies B	"Zero downtime" `implies` "Blue-green deploy"
`entails`	A necessarily entails B	"Compliance req" `entails` "Audit trail"
`prevents`	A prevents B	"Validation" `prevents` "Invalid data"
`blocks`	A blocks B	"Firewall rule" `blocks` "External access"

Each association has a **strength** score (0.0–1.0) indicating confidence.
The CLI supports 4 of these 13 types via the `associate` command: `supports`, `contradicts`, `caused_by`, and `related_to`. The MCP server supports all 13.

---

8. Lifecycle Management

Memory bases grow over time. Samvid provides four lifecycle operations to keep the knowledge base clean and relevant.

8.1 Importance and Decay

Every memory has two properties that control how it ages:

- **Importance** (0.0–1.0) — how valuable the memory is. Set at creation and can be updated.
- **Decay rate** (default: 0.01) — how fast importance fades over time.

The effective importance at any moment is:

```
effective_importance = importance × e^(-decay_rate × age_in_days)
```

This model means:

- High-importance memories (0.9+) persist for a very long time.
- Medium-importance memories (0.5) fade gradually.

- Low-importance memories (0.1) become stale quickly.

Protection: Memories with base importance 0.9 or higher are never deleted by the `forget` operation, regardless of their effective importance.

8.2 Time-to-Live (TTL)

A memory can be assigned a TTL (time-to-live) in days. After that many days, the memory is eligible for removal during the next `compact` operation.

Use TTL for temporary information: sprint notes, reminders for this week, debugging context.

8.3 Forgetting

The `forget` operation (CLI: `samvid forget`, MCP: `forget`) deletes memories whose effective importance has fallen below a threshold.

> ***IMPORTANT***
> *Memories with a base importance of 0.9 or higher are **always preserved** by the forget operation, regardless of their age or effective importance. Mark your critical memories with importance ≥ 0.9 to protect them from automated pruning.*

Run this periodically — monthly is a good cadence — to prevent the knowledge base from accumulating irrelevant old notes.

8.4 Compacting

The `compact` operation (CLI: `samvid compact`) does two things:

1. Removes all memories whose TTL has expired.
2. Runs SQLite VACUUM to physically reclaim disk space.

Run this after significant deletion activity or as part of routine maintenance.

8.5 Consolidating

The `consolidate` operation finds memories that are semantically nearly identical (cosine distance below a threshold) and merges them. The older memory absorbs the newer; the higher importance value is kept.

> ***WARNING***
> *Consolidation is irreversible. Always preview with `--dry-run` first and review the proposed merges before applying them permanently.*

Preview:

```
samvid consolidate --dry-run
```

Then apply after review:

```
samvid consolidate
```

8.6 Summarizing

The summarization pipeline compresses many raw memories into a single summary:

1. Use `summarize-prep` (CLI) or `prepare_summarization` (MCP) to check if a namespace has enough raw memories to justify summarization and to create batches.
2. For each batch, generate a summary (using any AI assistant).
3. Store the summary with `store-summary` (CLI) or `store_summary` (MCP). The source memories are automatically downgraded: importance reduced to 30% and type changed to `summarized`.

---

9. Troubleshooting and Support

Issue: Desktop App Starts But Shows No Data

If you previously uninstalled and then reinstalled, your old database may still exist at the platform-specific location.

To resolve:

1. Verify the database file exists at your platform's default location (see Section 2.5).
2. The app automatically loads any existing database found there. No additional action is required.

Issue: MCP Server Starts But AI Agents Cannot Connect

To resolve:

1. Verify the server is running: open a terminal and run `samvid stats`. If this command works, the server and database are functioning.
2. Confirm the AI client's configuration file is in the correct location and matches the format shown in Section 5.4.
3. Verify the URL in the configuration is exactly `http://127.0.0.1:8258/mcp`.
4. Ensure no firewall or security software is blocking connections to localhost port 8258.
5. Restart the AI client after making any configuration changes.

Issue: Search Returns No Results

To resolve:

1. Verify that memories exist in the database: run `samvid stats` and check the total memory count.

2. Try hybrid search instead of semantic search: `samvid hybrid-search "your query"`.
3. Ensure progressive search is enabled (it is on by default in both the CLI and MCP tools).
4. If using a fixed threshold, lower it: `samvid search "query" --no-progressive --threshold 0.5`.

Issue: Port 8258 Already in Use

To resolve:

1. Start the MCP server on a different port:

```
samvid-mcp.bat --port 9000
```

1. Update the URL in your AI client's configuration to `http://127.0.0.1:9000/mcp`.
2. Restart the AI client.

> ***NOTE***
> *If port 8258 is in use because a previous instance of the server did not shut down cleanly, restarting your computer releases the port.*

Support Resources

For additional support, contact: **ananddey.nic@gmail.com**

---

10. Glossary

Term	Definition
Memory	A single piece of stored information — text plus metadata (type, importance, namespace, tags).
Vector / Embedding	A 1024-dimensional numerical representation of a text's meaning.
Semantic Search	Searching by meaning rather than exact words.
Cosine Distance	A measure of vector dissimilarity. 0 = identical meaning, 2 = completely opposite.
Hybrid Search	Combines semantic search (meaning) with keyword search (exact words).
Progressive Search	Automatically relaxes the relevance threshold until results are returned.
RRF	Reciprocal Rank Fusion — a method for combining ranked results from different search systems.
Namespace	A logical group for memories (e.g., "work", "personal").
Session	A chain of related memories sharing a session ID.
Importance	A 0.0–1.0 score indicating memory value. Decays over time.
Decay Rate	Speed at which importance fades. Default 0.01.
TTL	Time-to-Live in days. Memory is removed after expiration.
Association	A typed link between two memories (e.g., "supports", "contradicts").
Knowledge Graph	The network of interconnected memories formed by associations.
MCP	Model Context Protocol — a standard for AI agents to call external tools.
MCP Server	A local HTTP service that exposes Samvid's tools to AI agents. Default: `http://127.0.0.1:8258/mcp`.
Streamable HTTP	The current MCP transport standard. All communication through a single endpoint.
SSE	Server-Sent Events — a legacy MCP transport.
stdio	A transport using standard input/output for local CLI-based agents.
Consolidate	Merge semantically similar memories into one.
Compact	Remove expired TTL memories and reclaim disk space.
Forget	Delete memories with low effective importance.
Summarize	Compress many raw memories into a single summary.
Deduplication	Automatic detection and merging of near-identical new memories.
Auto-Chunking	Splitting long content at sentence boundaries into linked memory chunks.
Episodic Chain	A sequentially linked series of memories replayable in chronological order.

Samvid Memory Studio — Version 1.0

Your data. Your machine. Your memory.