

INTERNAL TOOLING BRIEF · COWORK / CLAUDE PLUGIN

ECHO Memory

Persistent, cross-session memory for Claude and CoWork. The assistant loads your context, decisions, and projects at the start of every session. No re-explaining.

v1.0.0 · Schema 4

Owner: Jason Stedwell

Stack: Pure Python · Obsidian REST API

Platforms: Windows · macOS · Linux

THE ONE-LINER

Claude forgets everything when a chat closes. ECHO fixes that. It gives the assistant a structured Obsidian vault that it reads at session start and writes to as you work. The result: **memory you can trust and retrieve**, running identically on every machine.

WHY IT MATTERS

The problem it solves

Every session starts cold. You re-explain who you are, what you're building, and what was decided last week. Notes scatter, duplicate, or land in the wrong place. Worse, writes fail silently — you think a note saved when it didn't. ECHO closes all four gaps.



Remembers across sessions

Profile, active scope, recent sessions, and open projects load automatically at session start. No re-briefing.



Routes itself

One `capture` call files a note, stamps it, indexes it, and cross-links it. No manual filing.



Recalls neighbourhoods

Ask about a topic. Get the note plus everything linked to it — people, decisions, meetings. Not one fragment.

GETTING STARTED

How to use it

Most of the time you do nothing. The skill fires on phrases like *"remember that,"* *"what do you know about me,"* and *"log this decision,"* and loads your context at the start of real work. For direct control, use the eight slash commands:

/echo-load

Cold-start read — profile, scope, recent sessions, inbox.

/echo-save

Route and persist a memory to its correct home in one call.

/echo-recall

Pull a topic plus its linked neighbourhood.

/echo-reflect

Extract durable takeaways from the session for one-tap capture.

/echo-triage

Drain the inbox — route aging captures to their homes.

/echo-health

Run the linter — surface drift, broken links, orphans.

/echo-sweep

Bring an upgraded vault up to spec — index plus links.

/echo-doctor

Readiness check — Python, reachability, auth, schema.

A typical session

1**Start working**

ECHO loads your profile, active scope, last session, today's note, and inbox. It confirms scope before doing anything.

2**Just talk**

Mention a decision, a project, or a preference. Say *"remember that"* and it's filed and cross-linked.

3**Recall on demand**

"What do we know about the ALABAMA wISP rollout?" returns the project note and its linked people, decisions, and meetings.

4**Session ends**

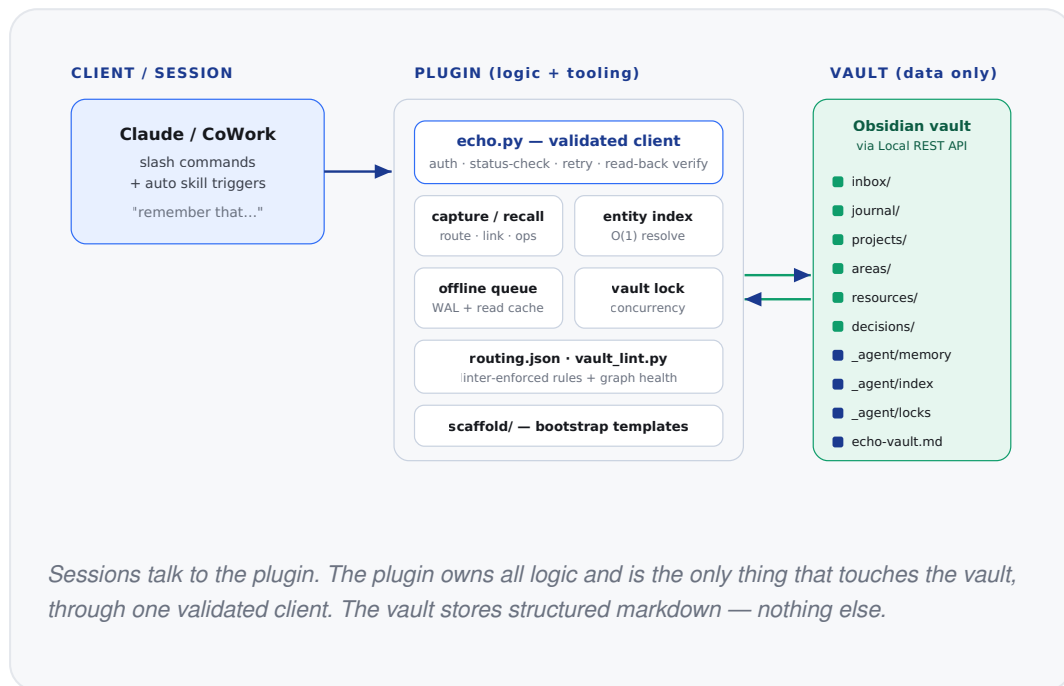
ECHO writes a one-line Agent-Log entry and a session log, then drops a heartbeat pointer to resume from next time.

Setup is two steps: run Obsidian with the Local REST API plugin, and set an API key (`echo.py write-key <token>` or the env var). Point the plugin at an empty vault and it builds the full structure itself.

UNDER THE HOOD

Architecture

One principle drives the design: **the plugin is the single source of truth**. All behavior — bootstrap logic, the operating contract, taxonomy, routing rules, and note templates — ships *inside* the plugin. The vault holds **data only**. That makes the system self-bootstrapping, updatable in one place, and portable across machines.



Key components

COMPONENT	WHAT IT DOES
echo.py	The validated client and CLI. Every read and write injects auth, checks HTTP status, retries transient errors, and read-back-verifies the result. A failed write fails loud, not silent.
capture / recall / resolve / link	The do-it-for-me layer. <code>capture</code> routes, stamps, indexes, auto-links, and logs in one call. <code>recall</code> returns a topic's linked neighbourhood.
Entity index	A slug→{path, kind, aliases} registry. Turns routing into an O(1), alias-aware lookup instead of a fuzzy search.
Hybrid recall	Local BM25 over note bodies, fused with decayed graph expansion. Keyword relevance and link structure, together.
Offline queue + cache	Vault unreachable? Writes queue to a local write-ahead log and replay idempotently later. Reads fall back to a last-known-good cache.
routing.json + linter	A machine-readable manifest of what writes where, enforced by a read-only checker — drift, broken links, orphans, frontmatter integrity.
scaffold/ + bootstrap	Templates and seeds that stand up an empty vault deterministically and idempotently.

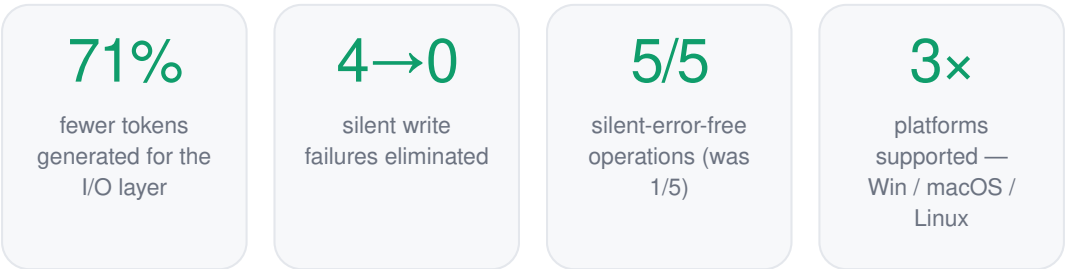
How memory is organized

ECHO mirrors a working memory model: **working** (transient state), **episodic** (what happened, when), **semantic** (durable facts and preferences), and **context** (the active focus). Daily content lives in plain folders — `projects/`, `journal/`, `decisions/`, `resources/`. The agent's bookkeeping sits under `_agent/`.

DOES IT ACTUALLY HELP?

Efficiency and reliability metrics

A credential-free A/B harness compares the plugin **before** (hand-built REST calls from the old skill docs) and **after** (the shipped validated client) across six representative operations, with deterministic fault injection. It measures the plumbing: token cost of the I/O layer and the rate of silent write failures.



METRIC	BEFORE (0.6)	AFTER (0.7+)	DELTA
Generated tokens (I/O layer)	608	174	-71%
Silent failures	4	0	-4
Duplicate lines written	1	0	-1
Silent-error-free ops	1 / 5	5 / 5	+4
Effective tokens (incl. recovery)	6,608	334	-95%

Source: [eval/run_eval.py](#) (mock OLRAPI, chars/token = 4.0). Generated-token counts are a proxy for the I/O layer only. "Effective tokens" applies a tunable recovery penalty on top of the hard silent-failure count.

What this measures: deterministic mechanics, not model reasoning (routing choices or prose quality). The "4 → 0 silent failures" figure is a hard count from ground truth. The recovery-weighted number is a model on top of it. CI runs the offline, reflect, and patch-semantics suites across Windows/macOS/Linux × Python 3.10/3.12.

Why "silent failures" are the real win

A failed write that looks like success is the worst outcome for a memory system. You trust a note that never saved. The old approach hid four across six operations. The validated client surfaces every one — non-zero exit, retry, read-back verify — so the assistant recovers instead of moving on. That is the line between memory and memory you can trust.

BUILT TO BE SAFE

Reliability and safety guarantees

Search-first writes

Before creating a note, ECHO searches the whole vault for the slug and title. No duplicates scattered across folders.

Idempotent appends

Reads before appending, skips on an exact whole-line match. A retry never duplicates an entry.

Concurrency-safe

A cooperative lock plus re-read-merge closes the race when Claude Code and CoWork write at once.

Never fabricates

The operating contract forbids inventing facts or decisions, mass restructuring, and storing secrets in notes.

Graceful offline

Vault down? ECHO says so once, queues writes, and proceeds. No retry storms.

Self-healing schema

Deterministic bootstrap, migration, and sweep scripts keep any vault on the current spec.

HOW IT GOT HERE

Maturity at a glance

VERSION	MILESTONE
0.3–0.4	Source promoted to a tracked tree. Search-first writes, project lifecycle, scope switching, monthly vault-health.
0.5	Self-bootstrap. The plugin becomes the single source of truth; vault migrated to data-only.
0.7	Validated client (echo.py), routing manifest plus linter, A/B eval harness: –71% I/O tokens, 4→0 silent failures.
0.8	Toolchain ported to pure Python. Runs identically on Windows, macOS, Linux.
0.9	Entity index, hybrid recall, bidirectional auto-linking, graph-health checks.
1.0	"Memory you can trust and retrieve." Hybrid BM25+graph recall, offline durability, concurrency hardening, reflection capture, secret hardening. CI across 3 OSes × 2 Python versions.

echo-memory v1.0.0 · Internal stakeholder brief · Generated June 22, 2026

Personal plugin built and maintained by Jason Stedwell (MPM / ALABAMA wISP). Endpoint and credentials are private. Not for external distribution.