VAAS-X SDK · AGENT MEMORY · VALIDATION REPORT

Validated: Recall That Surfaces the Fix, Not Just a Guess

How outcome-grounded memory let one agent session recover a confirmed fix from a completely different session's history on live production infrastructure — then held up again at 3,586-episode independent scale with a statistically significant result.

100%	3,586	p≈0.0002	0
VALIDATED-POOL RETRIEVAL, ZERO UNFILTERED GUESSES	INDEPENDENT EPISODES, REAL DATA	STATISTICALLY SIGNIFICANT RESULT	SDK OR SERVER CHANGES REQUIRED

VAAS-X Ltd (SC873624) · July 2026 · vaasx.com

The problem: "similar" isn't the same as "worked"

An agent that acts, observes an outcome, and later faces a similar-looking situation has three questions to answer, not one: what happened before that looks like this, what was tried, and did it actually work. Standard vector memory answers the first question well and leaves the other two to the agent's own judgement — which in practice means re-deriving "did it work" from context that may no longer be available, or not checking at all.

What the `vaas-x` SDK actually does

Every stored episode carries an explicit outcome, not just a similarity fingerprint: a success/failure signal that biases future ranking by default, and a separate `resolved` tag written once the episode's real-world result is known — distinct from the success/failure stats, so an episode can carry feedback without ever being marked as resolved.

```
from vaasx import Bootstrap

brain = Bootstrap(api_key="YOUR_API_KEY", device_id="agent_worker_3")

hits = brain.query("checksums match but the running process still behaves like old
code", k=5)
for hit in hits:
    print(hit["score"], hit["outcome"])

# Ground the episode once you know what actually happened -- this is
# the field a plain vector store has no equivalent for.
brain.outcome(episode_id=hits[0]["id"], success=True,
resolved="fixed_deployment_drift")
```

Validated: cross-session recall, on live production infrastructure

Rather than a synthetic benchmark, this was tested against real diagnostic history: two agent sessions each ingested their own hypothesis/outcome record from genuine debugging engagements, then a third session — with zero history of its own — queried across both via `HiveMind` to see whether cross-session recall could surface a relevant precedent it had no memory of on its own.

> **RESULT**

> The zero-history session recovered a directly relevant, grounded precedent from a completely different session's history — something no single session's own context window could do once that session has ended. Every episode's text was the real diagnostic language used during the actual engagements, and every result shown was a live API response, not a mock.

```
from vaasx import HiveMind

fleet = HiveMind(api_key="YOUR_API_KEY")
fleet.add_shard("agent_worker_1")
fleet.add_shard("agent_worker_2")

hits = fleet.query("checksums match but the running process still behaves like old
code", k=5)
```

Trust-gated ranking: guaranteeing a confirmed fix outranks a guess

For callers that need a guarantee — never surface an unconfirmed lead ahead of a validated result — the same `query()` / `HiveMind.query()` interface supports a simple, additive client-side pattern: fetch a wider candidate set, partition it into episodes with a confirmed successful outcome and everything else, rank within the validated pool, and only fall back to the full unfiltered list — explicitly flagged as unvalidated — when nothing validated exists yet. It's a hard gate rather than a scoring blend, so it doesn't depend on how large any particular similarity gap happens to be, and it requires no server or SDK change: pure post-processing over the existing public interface.

Validated at scale: 3,586 independent episodes, not six

A live test against a handful of real episodes proves a mechanism works; it doesn't prove it generalises. The same gate-then-rank pattern was tested again against data built entirely independently of it — NASA's public CMAPSS FD001 turbofan degradation benchmark, mapped into 3,586 real episodes with a genuine, externally-grounded health label per window, using a fixed-in-advance heuristic claim layer and leave-one-out retrieval across every episode.

> **RESULT**
>
> Gating toward validated claims measurably improved whether the retrieved precedent's true label actually matched the query's — 53.7% to 58.1% top-1 accuracy, a statistically significant improvement ($p \approx 0.0002$) on a dataset built for an entirely different domain and never tuned for this test.

The two results are complementary: the live production test shows the mechanism resolves a sharp, specific recall case; the independent 3,586-episode benchmark shows the same mechanism holds up, at three orders of magnitude more scale, on data structurally unrelated to what it was first built against.

Deployment model

Bootstrap's profiling, classification, and episode accumulation run locally, on your own infrastructure. Cross-session and cross-device recall via `HiveMind` is available from the Developer tier upward, fanning a single authenticated query out across every registered device on your account and merging results by outcome score — no per-device integration work beyond registering device IDs.

> **SCOPE, HONESTLY**
>
> Outcome-grounded recall, `resolved` tagging, and `HiveMind` fan-out are production today, exactly as shown above. The trust-gated ranking pattern is a client-side recipe over the existing public interface, not a separate billed feature or a new SDK method — anyone on any tier that supports `HiveMind` can implement it directly against `query()`. The CMAPSS-scale result is a real, independently reproducible statistical finding, not a projection.

Get started

Free tier: instant API key, no credit card — `pip install vaas-x`, connect a source, query your first episode in under ten minutes.

Full guide: vaasx.com/docs/getting-started.html | Enterprise: hello@vaasx.com