VAAS-X SDK · AI INFRASTRUCTURE

Agent Memory That Survives the Session

How the vaas-x SDK gives AI agents outcome-grounded episodic memory across sessions — validated in a live 35,000+ episode autonomous loop, not a benchmark demo.

35,000+	14ms	0	1
EPISODES, LIVE LOOP	RECALL, 1.18M EPISODES	GPU REQUIRED	SDK, ANY FRAMEWORK

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

The problem: every agent framework has the same gap

Context windows aren't memory. An agent can carry a long conversation within a single session, but the moment that session ends, everything about what worked, what failed, and why is gone. The next run starts cold — no matter how good the last one's outcome was. Bolting a vector database on top gives you similarity search, not memory: it can find text that looks like your query, but it has no concept of whether the thing it's returning actually led anywhere good.

Outcome-grounded recall, not just similarity

The vaas-x SDK stores every agent action as a State-Action-Outcome episode — what the agent observed, what it did, and what happened as a result — and by default weights query results toward episodes with a recorded successful outcome (`prefer_success=True`). That's the difference between "this looks similar to what you're asking" and "this is what actually worked last time."

```
from vaasx import Bootstrap

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

hits = brain.query("customer requesting refund, order not yet shipped", k=5)
for hit in hits:
    print(hit["score"], hit["episode"])

# Close the loop: tell the system whether the action taken actually
# worked. Future queries surface what worked, not just what matched.
brain.outcome(episode_id=hits[0]["id"], success=True)
```

Rate-limit and quota errors return a machine-readable 4xx via `VaasxAPIError` rather than an ambiguous failure, so agent code can distinguish "nothing matched" from "you're over your tier's limit" and handle each correctly.

```
from vaasx import VaasxAPIError

try:
    brain.query("...")
except VaasxAPIError as e:
    print(e.status_code, e.detail)
```

Validated: 35,000+ episodes in a live autonomous loop

> **IN PRODUCTION, NOT A DEMO**
>
> A live deployment has accumulated more than 35,000 episodes through an autonomous agent loop — ingest, query, outcome, repeat — running continuously rather than as a one-off benchmark. Retrieval stays fast at that scale: it returns the closest match from over a million stored episodes in roughly 14ms on standard CPU hardware, with no GPU in the path.

Multi-agent fleets: HiveMind

Agent deployments rarely stay single-instance. HiveMind fans the same authenticated query out across every registered device or agent instance on your account and merges the ranked results, so a fleet of agent workers shares one memory rather than each starting cold.

```
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("your query", k=10)
```

No per-device integration work beyond registering the device IDs — available from the Developer tier upward.

What ships client-side, and what doesn't

The installed vaas-x package ships Bootstrap's local pipeline — profiling, classification, episode accumulation — so an agent's raw interaction data is processed on your own infrastructure before anything reaches the network. The retrieval index, the ranking logic behind `query()`, and encrypted-search deployments all run server-side against your account and are never present in the installed package. That's a deliberate boundary, not a limitation: it's what lets the same client run from a lightweight worker process to an enterprise server without shipping the substrate's retrieval logic to every machine that calls it.

> **SCOPE, HONESTLY**
>
> Text-based outcome-grounded memory — everything shown above — is production today, including at the 35,000+ episode scale cited. Multimodal encoding via ArtificialCognition

> (vision, audio, IMU, depth, thermal) is early access — useful for agents operating on more than text, but those specific encoder models are still in active training. Encrypted search is a roadmap item for custom Enterprise deployments, not yet part of the standard hosted service.

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