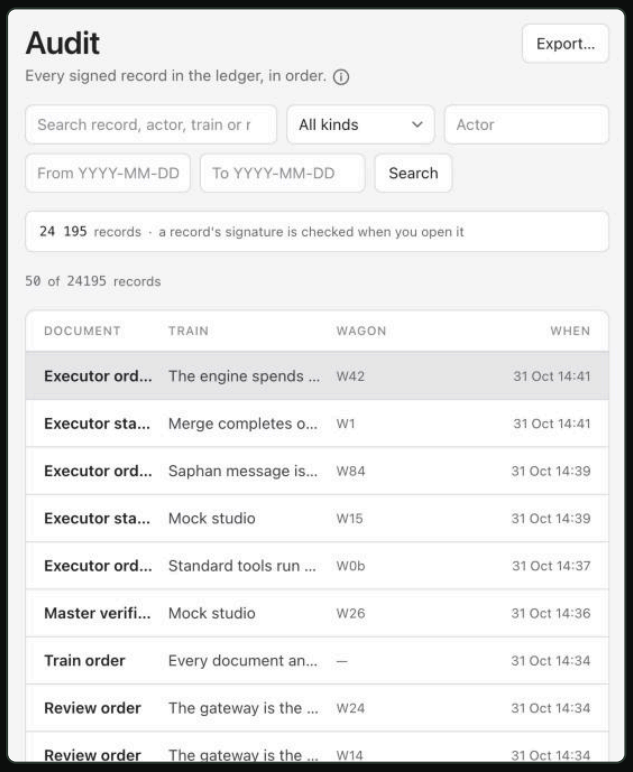
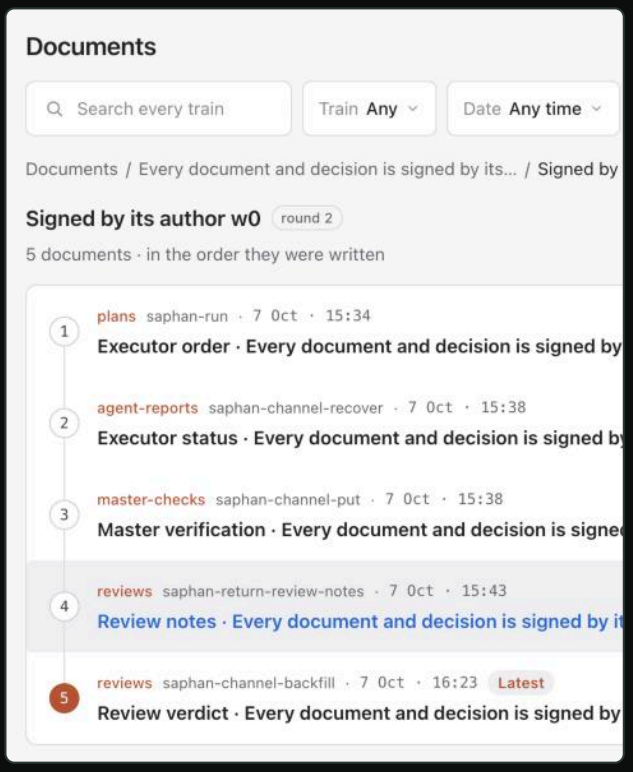
Auditing and observability with Saphan Studio

Who approved what, on what evidence, and how to check it yourself. Real screens from a working installation of AI agents under human approval.





24,195

signed records in the ledger on 31 October 2026

Every one

has its signature checked when it is opened

Read-only

export with signatures, for an auditor outside

The question an auditor asks

When AI agents write code and touch infrastructure, a supervisor, an auditor or a board will ask one question about any change: **who approved this, and on what evidence?**

Saphan Studio answers it from the record, not from memory and not from screenshots collected after the fact. For every change, the record holds:

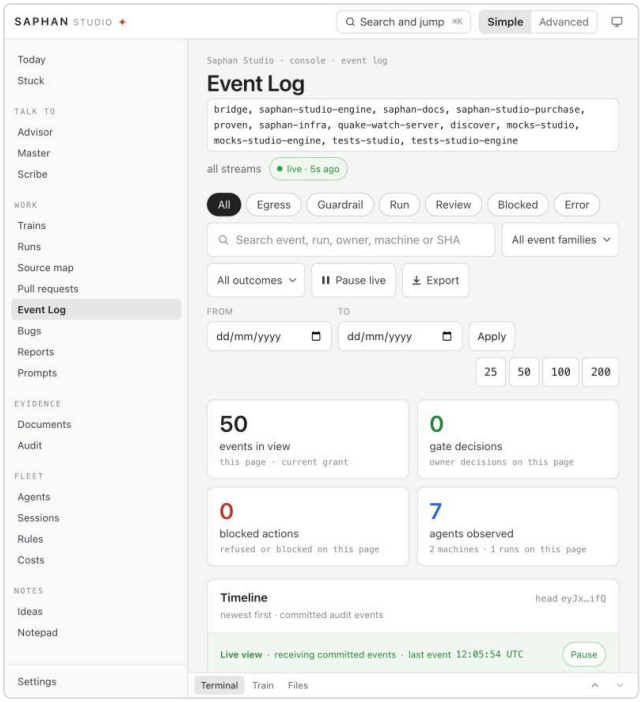
- **the order:** what was asked, by whom, within which boundaries;
- **the work:** which agent did it, in which run, and what it reported;
- **the checks:** the verification, the review notes and the review verdict;
- **the decision:** the person who approved it, and when;
- **everything that was refused** along the way, with the rule that refused it.

Every document in that chain is signed, and its signature is checked when anyone opens it. This paper walks through the screens an auditor and an operations team use, with real screenshots from Saphan's own installation, taken on 31 October 2026. Machine names are blurred; nothing else is.

You want to...	Use
watch what agents are doing, live	**Event Log** (section 1)
read one agent's session, and every tool it was refused	**Sessions** (section 2)
see every signed record, in order, and check one	**Audit** (section 3)
follow one change from the order to the verdict	**Documents** (section 4)
hand the evidence to someone outside	**Export** (section 5)
feed your own monitoring	**Metrics export** (section 6)

1. Event Log: what is happening, live

The Event Log is the operations view: every committed event from every repository the agents work on, newest first, as it happens.



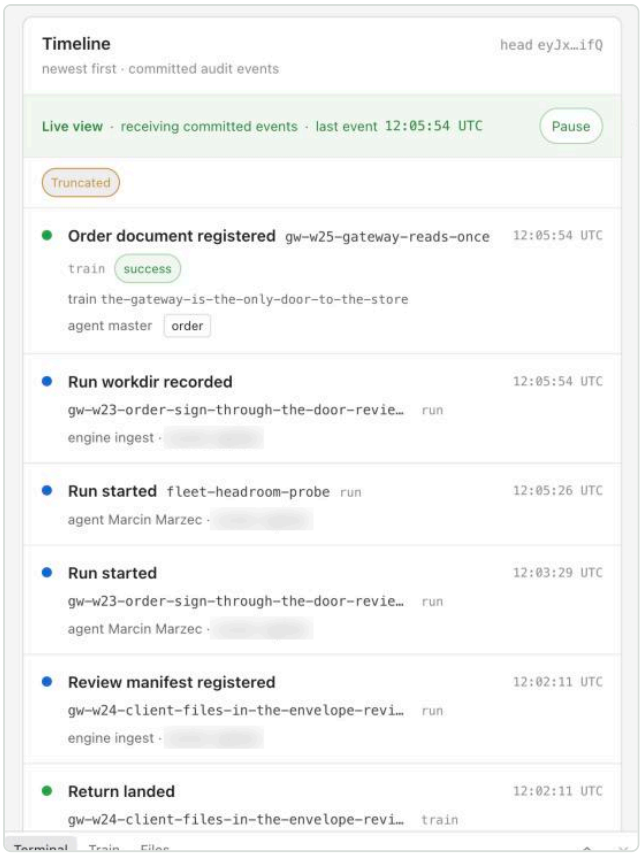
Event Log across all repositories, live. Filters by family (Egress, Guardrail, Run, Review, Blocked, Error), by outcome and by date. Counters show events in view, gate decisions by people, blocked actions and the agents observed.

What you can do here

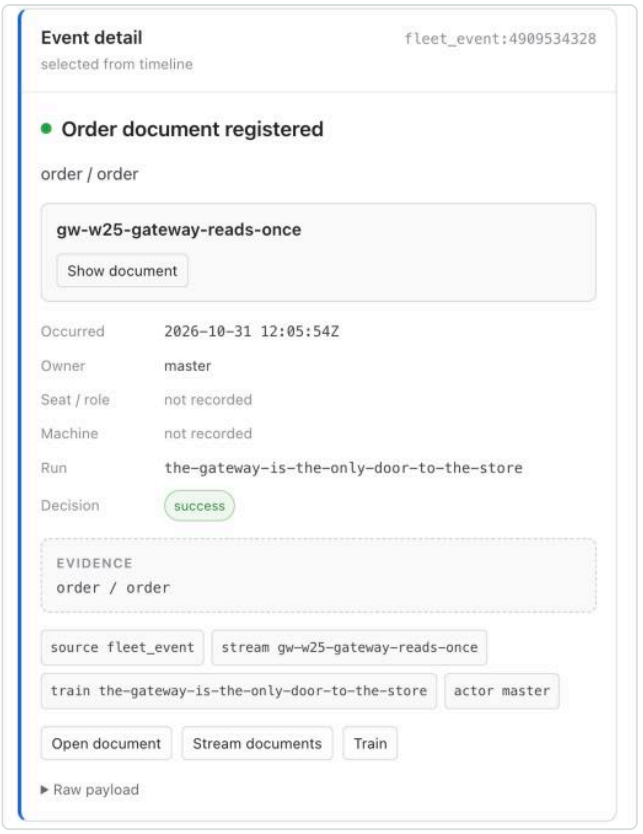
- **Filter by family.** *Egress* shows network attempts, *Guardrail* the content-safety checks, *Blocked* every action that was refused.
- **Search** by event, run, owner, machine or commit.
- **Pause the live view** to read without the list moving.
- **Export** what is in view.

The counters only count what they measured. When a number is not known, the console shows nothing, never a zero it did not measure.

The timeline's *head* (top right of the timeline) identifies the newest committed event, so two people looking at the log can tell whether they are looking at the same state.



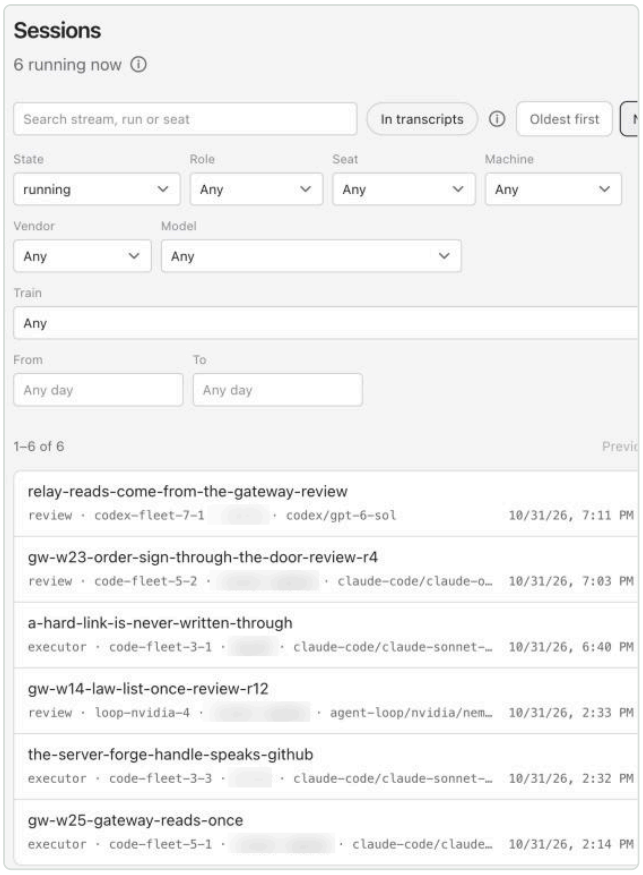
The timeline: an order registered by the coordinating agent, runs started by a person, a review manifest registered by the engine. Each row names the run or train, who acted, and where.



One event opened: when it occurred, who owns it, the run and the decision, the evidence it points to, and links to the document, the other documents of the same stream, and the train.

2. Sessions: watching an agent work

Every agent session, running or finished, can be opened and read like a flight recorder: which work it was doing, in which role, on which seat and machine, with which vendor and model, and every tool call it made, with the output.

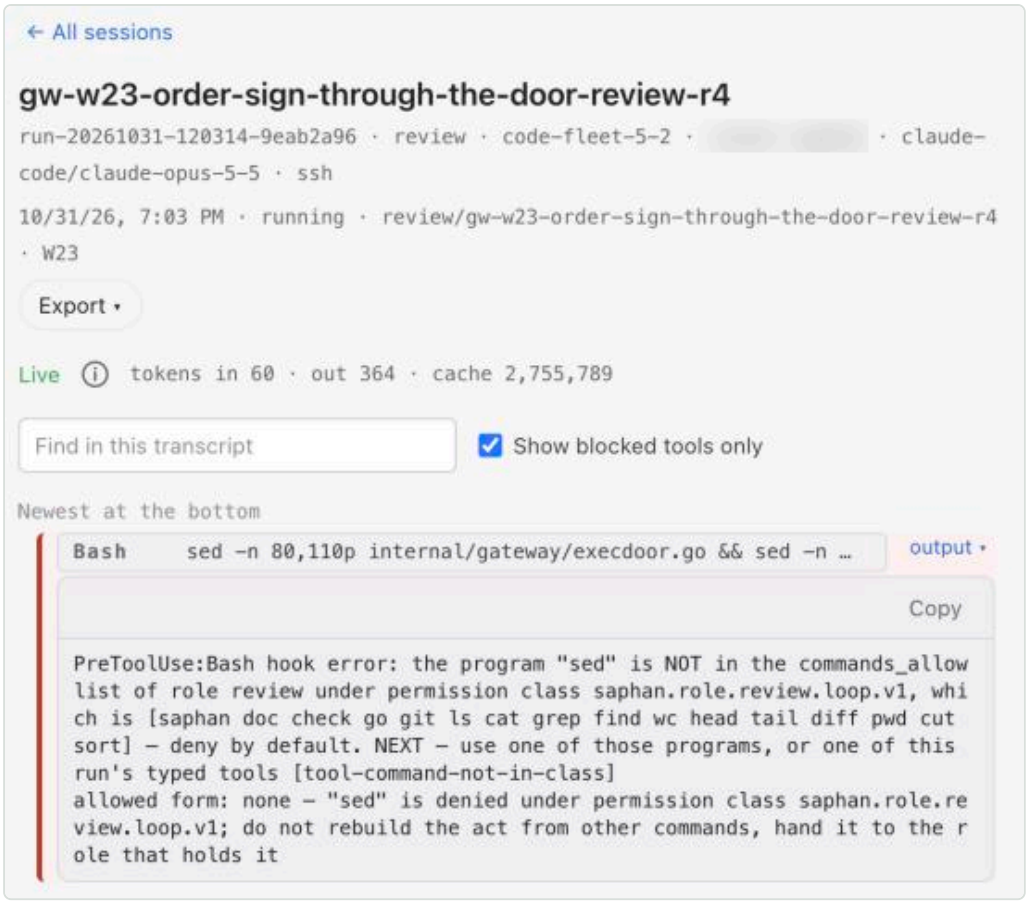


Six sessions running at once, from three vendors: Codex, Claude Code and an NVIDIA Nemotron agent loop. Each row names the work, the role (executor or reviewer), the seat, the machine and the model. Filters by state, role, seat, machine, vendor, model, train and date, and a search inside the transcripts.

What a session shows

- **The work it belongs to:** the train, the wagon and the round, so a session is never an anonymous chat.
- **Who did it:** the role, the seat (a named agent identity with its own profile) and the machine.
- **Which model:** the vendor and the exact model, because "the AI did it" is not an answer an auditor accepts.
- **Every tool call,** in order, with its output, and the agent's own notes on what it is doing and why.
- **Usage:** tokens in, out and from cache, live.
- **Export** of the whole transcript.

A refusal, as the agent saw it



"Show blocked tools only": one click, and the session shows only the calls that were refused. Here a reviewer agent (Claude Code on Claude Opus) reached for `sed`. Its role may run only a short list of programs, and everything else is denied by default.

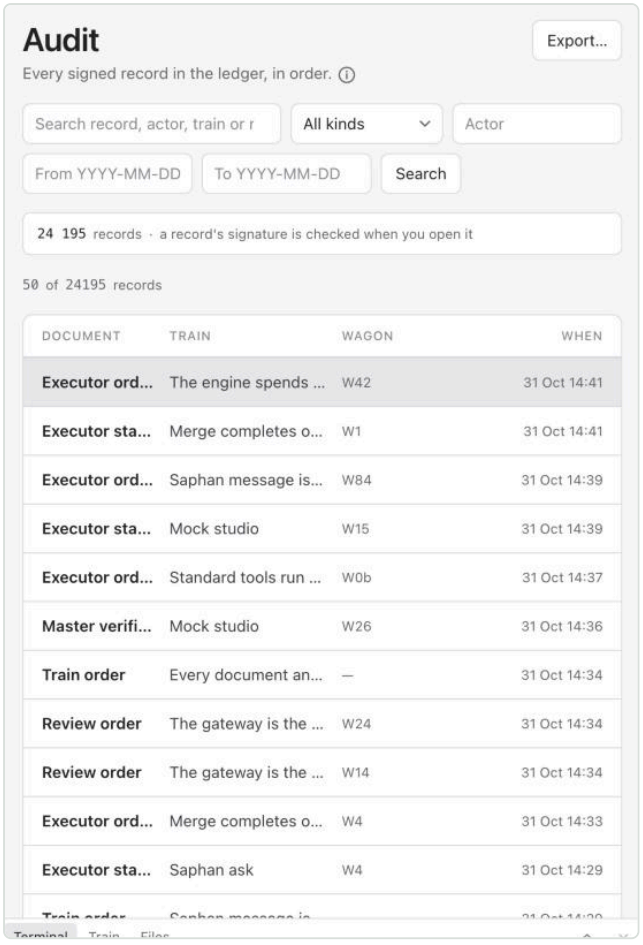
The reviewer was doing nothing wrong. It wanted to read a few lines of a file, and reached for a tool that was not on its role's list. The refusal is a good example of what zero trust for agents looks like in practice:

- **Deny by default.** A reviewer may run a fixed list of programs for reading code. Anything not on the list is refused, however harmless it looks.
- **The rule is named.** The refusal says which role, which rule, and what the role is allowed instead, so neither the agent nor the person reading it later has to guess.
- **No workaround.** The agent is told not to rebuild the same act from other commands, but to hand it to the role that holds it.
- **It is on the record.** The refusal stays in the session and in the Event Log, with the run it belongs to.

Not even the tools are trusted by default, including the trusted vendor tools.

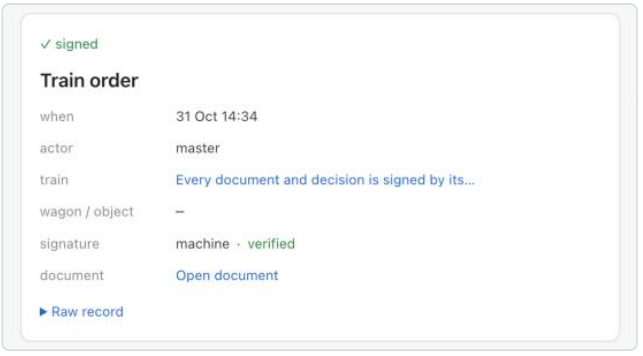
3. Audit: every signed record, in order

The Audit screen is the ledger itself: every signed record, in the order it was written. It is read-only. Nothing on this screen can change a record.



24,195 signed records on 31 October 2026. Executor orders and statuses, master verifications, review orders, train orders, each with its train, its wagon and its time. "A record's signature is checked when you open it."

Reading one record



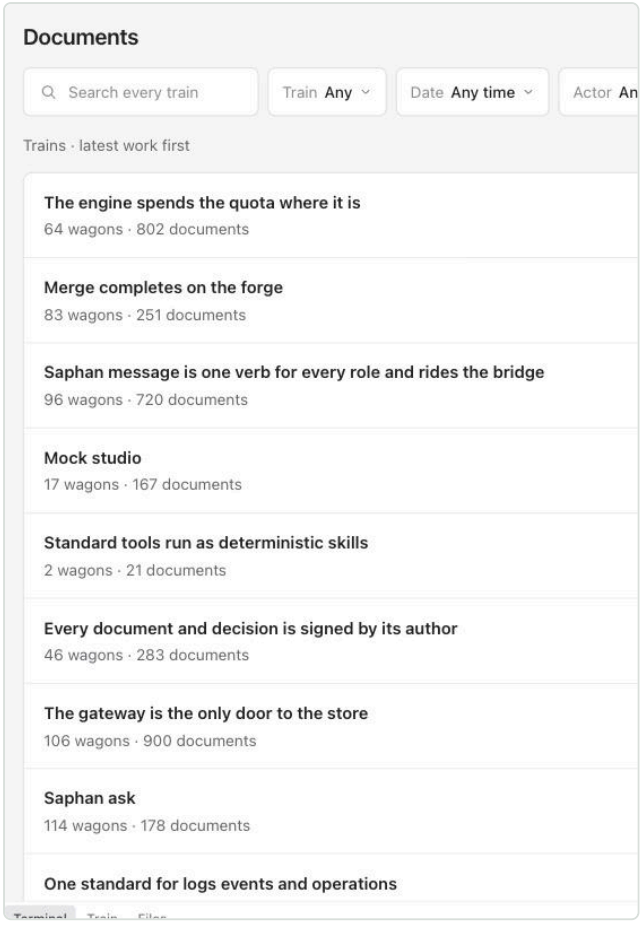
A train order, opened: signed, by whom, for which train, with its signature verified and a link to the full document and the raw record.

Search by record, actor, train or date range. Filter by kind of record: orders, statuses, verifications, review notes, verdicts, decisions.

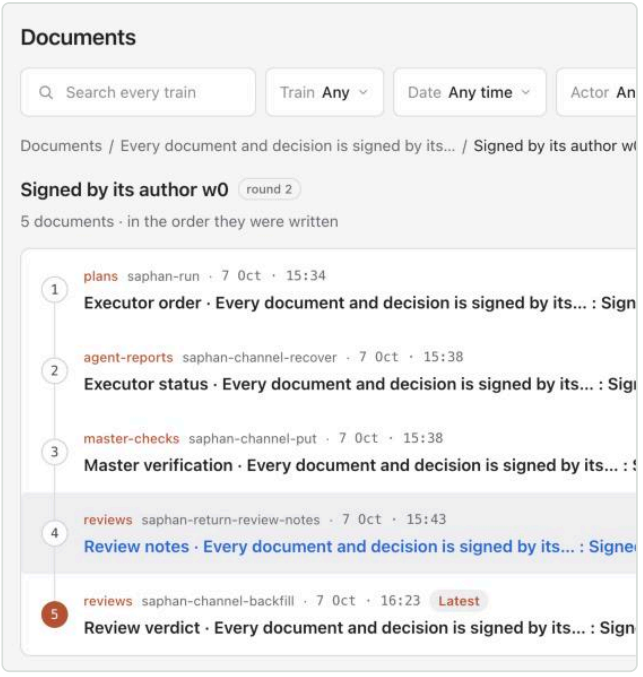
Every record links to the document it signs, so the auditor never has to trust a summary: the full text is one click away.

4. Documents: one change, from the order to the verdict

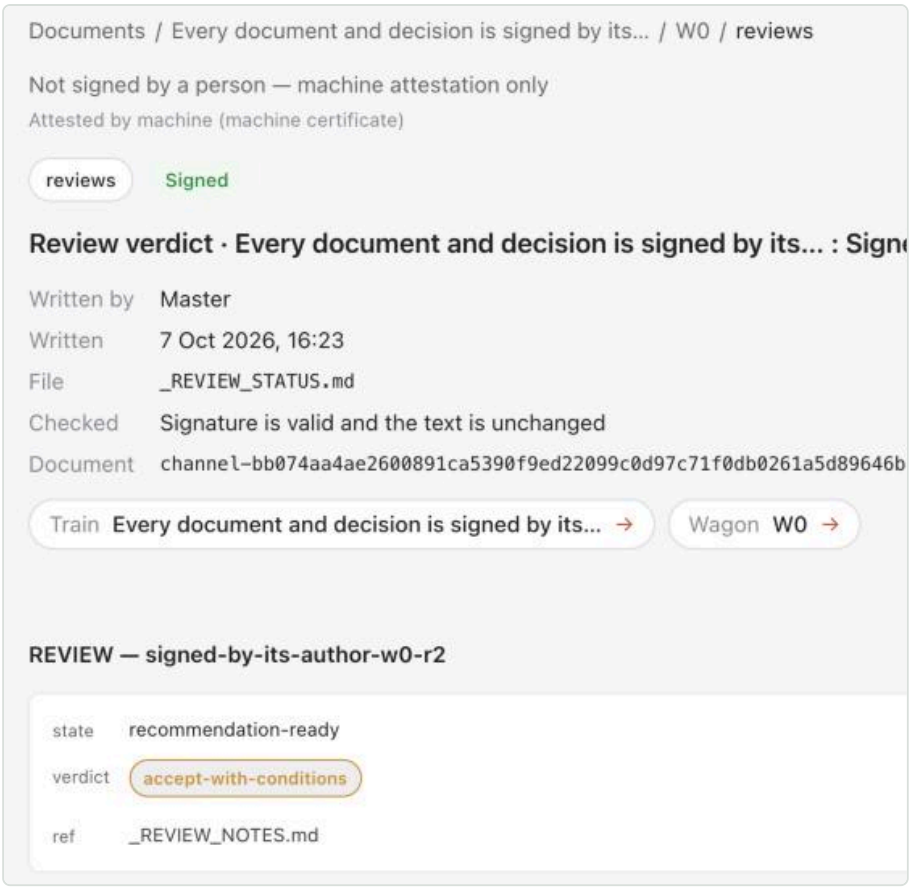
Work in Saphan Studio is organised in trains (a planned piece of work) and wagons (its steps). The Documents screen shows, for each wagon and each round, every document in the order it was written.



Trains, latest work first, with the number of wagons and documents behind each. The train "Every document and decision is signed by its author" has 24 wagons and 283 documents.



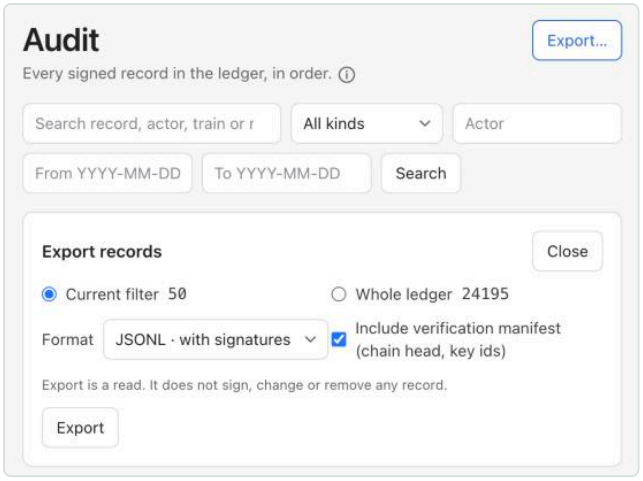
One round of one wagon: five documents, in the order they were written. The plan (executor order), the agent's report (executor status), the coordinator's check (master verification), the review notes, and the review verdict.



The review verdict, opened: who wrote it, when, which file, and the check: "Signature is valid and the text is unchanged." The screen says plainly what kind of signature it is: "Not signed by a person, machine attestation only." A decision that belongs to a person carries that person's own signature, and the screen tells the two apart.

This is the chain an auditor walks backward: from a merged change to the verdict, the review, the verification, the report, and the order that started it. The public documentation walks through it step by step: Tracing one change backward.

5. Export: evidence for someone outside



Export records: the current filter or the whole ledger, as JSONL with signatures, with an optional verification manifest. "Export is a read. It does not sign, change or remove any record."

An auditor should not have to log in to your console, or trust it.

The export gives them the records with their signatures, and a manifest with the chain head and key ids. Exporting is a read: it cannot sign, change or remove anything.

What a third party can check from your organisation's public key alone, in the documentation's own words:

1. the order, in full text, was signed by the person who issued it;
2. the execution status was signed by the agent seat that did the work;
3. the review verdict and notes were signed by the reviewer's seat;
4. the human decision at a gate was signed by the person's own key;
5. nothing was altered afterwards.

Source: What a third party can check, with the conditions under which each item holds.

6. Observability: feeding your own tools

The console is for people. Your monitoring team will want the numbers in the tools they already run.

Saphan publishes a small, deliberately narrow set of metrics from its record to any **OpenTelemetry (OTLP/HTTP)** collector you run: Prometheus behind the OpenTelemetry Collector, Grafana Alloy, Datadog's OTLP intake, or anything else that speaks the protocol.

Metric	What it tells you
`saphan.run.cost_usd`	the quoted and the actual cost of each run
`saphan.review.rounds`	how many review rounds a change needed
`saphan.account.usage_pct`	how much of each agent account's session and weekly allowance is used
`saphan.run.wall_clock_cap`	the time limit each run was given (what it was allowed, not how long it took)

Three rules govern the export: it **never sends recorded content**, it **never runs on its own** (you run it when you choose), and it is **never on the run path**, so an export cannot slow an agent down. Each metric exists only because a column in the record answers it directly; the exporter reads and never computes. Details: Metrics export: what you get.

What is coming

Agent events delivered to the SIEM your security team already watches are planned for the Enterprise edition. It will be described here when it runs, not before.

7. Limits, stated as limits

A record that claims more than it can prove is worse than no record. These are the limits today, from the public documentation.

- **Machine attestation is not a person's signature.** Many documents (reports, verifications, verdicts) are attested by the machine that wrote them. The screens say so, and a person's decision is signed separately.
- **The offline check is not yet one command end to end.** An offline verifier, `saphan-verify`, ships in the release packages and checks one change's signed chain without our servers. Producing the bundle it reads is not yet a single command.
- **"Verified" means verified as of a checkpoint.** A revocation recorded after that checkpoint is covered only by a newer one.
- **One signer per act.** There is no quorum signing.

The full list: What this does not prove and Limits of the signed record.

Read next

Page	What it answers
Audit and compliance	What the record proves, where the evidence lives, and what it does not prove.
The questions a framework makes an auditor ask	Where in the record each question is answered.
Verify a change yourself	Checking one change on your own machine, with no Saphan store and no network.
Two sentences, two hours, a signed train	One feature from an idea to a signed plan, screen by screen.

Saphan Systems, Inc. · saphan.ai · docs.saphan.ai
U.S. and EU patent pending. Member of the NVIDIA Inception program.