One signed receipt. One offline verifier. That is the whole trick.

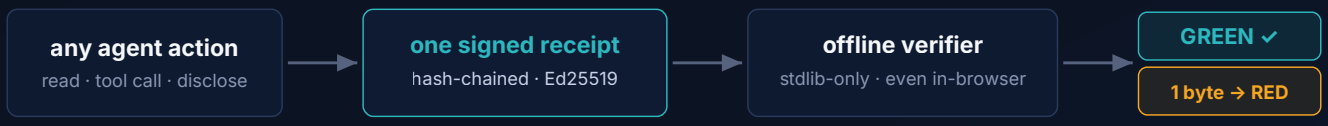
Governance that proves itself.

You are putting AI agents into real work — writing code, moving money, touching data. The day one does something wrong, you discover you cannot actually **prove** what it did: the logs are yours, so you can edit them, and a record you control is one no regulator, customer, or court has reason to trust. **That gap — between what agents can do and what you can prove they did — is the entire problem.**

THE INNOVATION THAT MAKES IT WORK

Every action an agent takes is written as a tamper-evident receipt in one standard format — and a tiny verifier can re-check it with no network, no engine, and no trust in us.

Most "AI governance" is a dashboard you have to believe. Ours is a proof you can re-run. Because every product speaks the *same* receipt format, two things fall out for free. **One:** any auditor, customer, or court can verify everything themselves, offline, long after the incident — the verifier is a few hundred kilobytes and ships in the bundle. **Two:** the products compose. Twelve tools become *one platform*, because they all write and read the same proof.



Change a single byte anywhere in a recorded run and verification turns RED, naming the spot — and a record *forged with a valid key* (an insider fabricating blame) fails just the same, because attribution is pinned to an anchored signer, not asserted in the record. A genuine cryptographic failure, not a script that complains.

SIGNED

Hash-chained & sealed

Each action is content-addressed, linked to the last, and signed. Nothing can be inserted, removed, or back-dated without breaking the chain.

OFFLINE

Verify without the vendor

The check needs no server, no network, and no ByteVerity. You are not trusting a log — you are re-deriving the result.

HONEST

Proven vs. asserted

The system proves only what it can prove. Where evidence isn't sound, it **refuses rather than guesses** — and labels the two differently.

> **"We don't ask you to trust our log. We hand you a proof you can check yourself — and we refuse to assert anything we can't prove.**

THE PLATFORM — SEVEN MOVES, ONE PROOF

Governing an AI agent comes down to seven things you must be able to do — and prove. ByteVerity ships a point product for each. They interlock because they all speak the same signed receipt — so this is one platform, not twelve tools.

VERB	POINT PRODUCT	WHAT IT DOES, PLAINLY
IDENTIFY	**Identity Projection**	Binds every action to a short-lived, revocable identity — minted in-platform or carried from your own IdP (Okta ID-JAG, SPIFFE/SPIRE).
CONTROL	**Data Gateway**	Sits between agent and data: redacts secrets, quarantines prompt-injection, releases only what policy allows.
	Cleanroom / Vault	Encrypts sensitive source & secrets into a vault; the agent sees only a harmless stub, never the real value.
	Browser Runtime	Governs what staff paste into AI tools in the browser, at the moment of send. Holds no keys of its own.
GOVERN	**Control + Governors**	The policy gate that *denies a risky action before it happens* and seals the decision — with release & underwriting editions.
LOCATE	**Bisect**	A flight recorder + 'git bisect' for agents: lands on the exact bad step, or the exact poisoned knowledge write — who & when.
PROVE	**Lineage**	Traces a break across code, run, knowledge, policy & approval into one signed record anyone re-verifies offline.
REVOKE	**Capability Gateway**	Revoke a capability — or an identity in your IdP — and every action under it fails closed, re-derived offline from a signed status list (no issuer call).
POSTURE	**Atlas**	A searchable, verifiable index over every signed record: "what did our agents know or disclose, and when?"

WHAT TIES THEM TOGETHER

Identity · Data Gateway · Cleanroom · Browser · Control · Capability · Bisect · Lineage · Atlas

ByteVerity Kernel — the shared proof substrate `one receipt format · one verifier, even in-browser`

Every product above emits Kernel-shape receipts; one tiny verifier checks them all. Adopt one product, then the next — the proof carries over, so you never re-buy trust.

THE FRONT DOOR & YOUR STACK

You meet the platform through one record — and it plugs into the identity system you already run.

THE FRONT DOOR — LINEAGE

One record. Seven planes. Anyone can re-check it.

Lineage is the single record a buyer, auditor, or regulator actually opens. It re-runs nothing — it *cites* the signed receipts every other product already produced and composes them into one causal chain across seven planes: **code, run, knowledge, policy, approval, data, and drift.**

From that one record it renders whatever each audience needs — a root-cause trace, a behavioral diff between two runs, an SLSA / in-toto provenance attestation, and the regulator pack itself: `EU AI Act Art.12` record-keeping plus an AI-BOM (`CycloneDX 1.6 / SPDX 3.0`).

trace | diff | attest · SLSA / in-toto | query | evidence · EU AI Act + AI-BOM | CI gate

All from one signed record — nothing re-run, nothing re-trusted.

THE PROOF TRAVELS — VERIFY IT IN A BROWSER

The verifier is a few hundred kilobytes and also compiles to WebAssembly. A skeptic pastes a record into a browser portal and re-checks it client-side — **zero install, zero network, and we never see it** (secrets are masked on display). The record cites your upstream receipts verbatim, so treat the exported bundle as sensitive.

IT RIDES YOUR IAM — IT DOESN'T REPLACE IT

Non-human identity, finally held to account.

Your identity provider already issues the grants — **Okta ID-JAG**, **SPIFFE / SPIRE.** ByteVerity doesn't stand up a parallel identity system; it binds every agent action to the identity your IdP already issued, and proves what that identity did.

Revoke the grant in Okta and **every past action under it turns RED** — re-derived offline from a signed status list, with no call back to the issuer. The agent's whole history fails closed, provably, with Okta and every engine absent. That closes the blind spot enterprises name most with agents: the service accounts and agents your IdP can issue but cannot yet hold to account.

PROOF, FIT & THE ASK

It’s built, and it runs today.

> *Each claim below is a shipped binary over real signed artifacts — we ran them. Every “tamper → RED” is a genuine cryptographic failure.*

Govern what it reads `cleanroom`	A live API secret is locked into the vault; the coding agent on disk sees only a stub. Every access is signed and verifies offline.
Locate the bad step `bisect / bisect-data`	An agent refunds an already-delivered order. Bisect lands on the exact step — and, for a poisoned knowledge base, names the exact write, who made it, and when.
Prove the cause `lineage`	The break is traced across seven planes — run, code, knowledge, policy, approval, data, drift — into one signed record that re-verifies with our engine absent (even in the browser, zero-install), the data plane on a real data-gateway capsule. A one-byte tamper turns it RED, naming the broken link.
Withstand a forgery `lineage · adversary demo`	A compromised agent forges a run verdict, a knowledge write and an approval — with valid keys. All three verify RED, each naming the break. **Forged blame never renders ‘proven.’** Offline — no IdP, no engine.
The whole lifecycle `end-to-end capstone`	One agent, one chain: `identified → governed → bad step located → proven offline → revoked` — then shown on a jargon-free fleet posture pane. No network.

THE HONEST MOAT

Every audit vendor claims a tamper-proof trail. Ours **draws the line between what it proves and what it merely asserts, and renders them differently** — proven causes solid, unproven ones dashed and counted apart. And the recording **can’t misreport which guardrail fired**: hide a redaction, downgrade a match, or re-seal what was released and re-derivation turns it RED. In a market full of tools that launder a guess as a fact, that refusal is the product.

WHERE IT FITS YOUR BUDGET

The **Audit** and **Provenance** pillars of Zero Trust for AI agents, done cryptographically — the two we go deepest on. Lineage renders the `EU AI Act Art.12` record-keeping pack and an AI-BOM (CycloneDX / SPDX) *straight from the signed record*, and lines up with `NIST 800-207` & `800-218A`. It attaches to security and compliance budgets you already have.

Felt first by AI-platform, security, compliance/risk and incident-response teams. (A consumer edge, **Sherpa**, applies the same signing tech to keeping a coding session on-task — same proof, different audience.)

THE ASK — START HERE

- **4-week paid pilot** on one or two of your real agent workflows. No model change, no rewrite — we sit at the boundary.
- **You** sign off the success criteria; we prove against them on your data, your keys.
- Fixed fee, **credited** toward annual (from **$60k**) on conversion.
- Adopt one product, then the next — the proof carries over.

> **If an agent acts and no one can prove what it did, you haven’t deployed a capability — you’ve accepted a liability.**