/Idora/

Receipts for what your agents ship.

Prevent incidents, ground edits, and prove deploys.

AI coding agents now produce the majority of code in modern engineering organizations.

Trust is the procurement blocker: Cisco RSAC 2026 reports 85% of enterprise AI pilots, 5% reach production.

> *Has anything broken since the last deploy, and which agent change caused it?*

> *What governs this file before an agent edits it?*

> *Did the v2.1 release ship its violations or fix them?*

Today these answers come from heavy instrumentation, custom dashboards, and hours of cross-tool reconciliation. Reconstructed every time. Held nowhere. The reconstruction approach breaks at agent volume.

Idora is a context graph for software delivery. Every event compounds into a receipt that agents and reviewers query.

Pushes, builds, tests, deploys, and verifications all write receipts. Receipts are content-addressed and accessible through MCP and Skills.

> *Ask what governs a file.*

> *Ask what drifted between versions.*

> *Ask whether a release conforms.*

Get the answer in one call with the proof.

Reads what your stack already emits. Holds what observability tools can’t: the integrity record.

claude code · auth.service.ts

> "What governs this file and what should I be aware of before editing?"

idora-context → file_context

3 governing requirements

AUTH-001 · conforms · verified 12×

high confidence · mapping_clarity: explicit

Last verification: Apr 28

AUTH-005 · indeterminate · evidence insufficient

runtime config dependency · graph boundary

Last attempted: Apr 26

AUTH-007 · does_not_conform · violation since Apr 10

high confidence · mapping_clarity: explicit

Re-verification required for any edit

↳ receipt · sha256:b3c1...44

The agent now knows what governs the file before writing a line. Three requirements: one conforming, one indeterminate where the runtime configuration sits outside the graph today, one in violation since April 10. The next edit accounts for all three. Edit-time grounding is what prevents the incident the next slide responds to.

pagerduty · INC-4127 · v2.1

> "Which agent shipped this and what drifted between v2.0 and v2.1?"

idora-agent · incident_traversal → drift

2 files degraded · 1 violation emerged

src/auth/auth.service.ts

  AUTH-007 · does_not_conform · violation since Apr 10

  high confidence · mapping_clarity: explicit

src/auth/middleware.ts

  coverage decreased · 2 verifications became indeterminate

↳ idora-proof → proof(claim: "AUTH-007 at v2.1")

  Last conforming: Apr 3 · violation: Apr 10

  Build #412 · auth.service.ts hash a3bf...c8

  Deployed v2.1 on Apr 19 despite violation

↳ code_provenance · sha256:e2d4...91

  claude-code · sess_01HXKM2R8N9P3Q5V

When the page fired, the on-call didn't reconstruct from logs. They queried the drift between v2.0 and v2.1 and got the proof chain plus the originating agent session in the same call. The receipts had been there since each version pushed. The substrate answers what no single-vendor tool can: which agent shipped this, on what session, against which requirement.

PROOF · RELEASE EVIDENCE · 06 / 08

release candidate · v2.1

> "What evidence should the reviewer see before approving this release?"

idora-agent · release_review → release_trust(ref: "v2.1")

determination: does_not_conform

confidence: high · mapping_clarity: explicit

blockers (1)

AUTH-007 · session integrity across token refresh

not satisfied at src/auth/auth.service.ts

↳ idora-proof → proof(claim: "AUTH-007 at v2.1")

evidence

28 of 31 requirements conform

2 indeterminate · runtime config dependencies

evidence layer · complements code review

847 receipts · ingestion_lag 4s

Evidence doesn't replace code review. It carries what a human reviewer can't carry alone: the determination, the blockers, the proof chain, and the boundary cases the substrate flagged as indeterminate. The reviewer reads the diff; the substrate reads the integrity record. Both meet at release.

Three patterns shown. The surface has seven tools, eight Skills, one Idora Agent.
Full reference: idora.dev/docs/product-surface

FOUNDER

Rodney Nespeca

Designed Idora’s substrate architecture, product surface, distribution layer, and the workflow the team builds it on. Previously founded and scaled a manufacturer serving national accounts. Background in corporate finance and M&A.

SUBSTRATE

Filip Szymanski

Built Idora’s graph data model with integrity semantics on Neo4j, the deterministic scoring engine, and the receipt pipeline. Previously engineer at Box.

SURFACE

Mateusz Rzepka

Built Idora’s MCP envelope, Skills, backend infrastructure, and ingestion paths. Previously backend engineer at mBank, a large European regulated bank.

No outside engineering.

WHERE WE ARE

Idora runs in production at Idora itself. Every push to the substrate writes a receipt. Design partner pilots open in June 2026 for engineering teams running coding agents in production at scale.

THE CONVERGENCE

The architectural pattern is named. Foundation Capital, Mitchell Hashimoto, the Linux Foundation, OpenAI, Anthropic. Context graphs and harness engineering are now the institutional architecture.

Trust is the named procurement blocker. Cisco at RSAC 2026: 85% pilot, 5% reach production. a16z: coding is the dominant enterprise AI use case at Fortune 500 scale. CodeRabbit: AI PRs produce 1.7× more bugs.

The cross-boundary gap is open. Anthropic Managed Agents and Salesforce Testing Center ship within-platform verification. Both validate the thesis. Both leave cross-boundary integrity open. Three regulatory frameworks effective Q3 2026 make the gap a legal requirement.

Read the full evidence catalog → idora.dev/docs/external-signals

THE ASK

We are opening design partner pilots. If your team runs coding agents in production at scale, we want to talk.

`engineering@idora.dev`