We Already Have the Graph

Autonomy Platform · Public Blog

PERSPECTIVE

Read more at azirella.com/blog

Technical April 2026

We Already Have the Graph

Most AI-native risk products are solving the wrong problem, extracting a supply-chain graph from text. Autonomy starts from the other direction. The technique stack inverts.

The standard story

A risk-intelligence platform, pick any of the well-known names, does something hard, and does it well. It crawls news, regulatory filings, court records, customs data, financial disclosures, and a list of paid feeds. From that corpus it *extracts* a graph: nodes for suppliers, products, geographies, customers; edges for “supplies to,” “located in,” “subsidiary of,” “exposed to.”

Then it overlays risk on that graph. A factory fire in Shenzhen lights up the supplier node. A typhoon track lights up a region. A sanctions update lights up a corporate hierarchy. Customers subscribe and watch the dashboard.

This is the right thing to do **if you are a third-party intelligence provider with no privileged view of any one customer’s supply chain**. You don’t have the graph. You have to build it from text. So you build it from text.

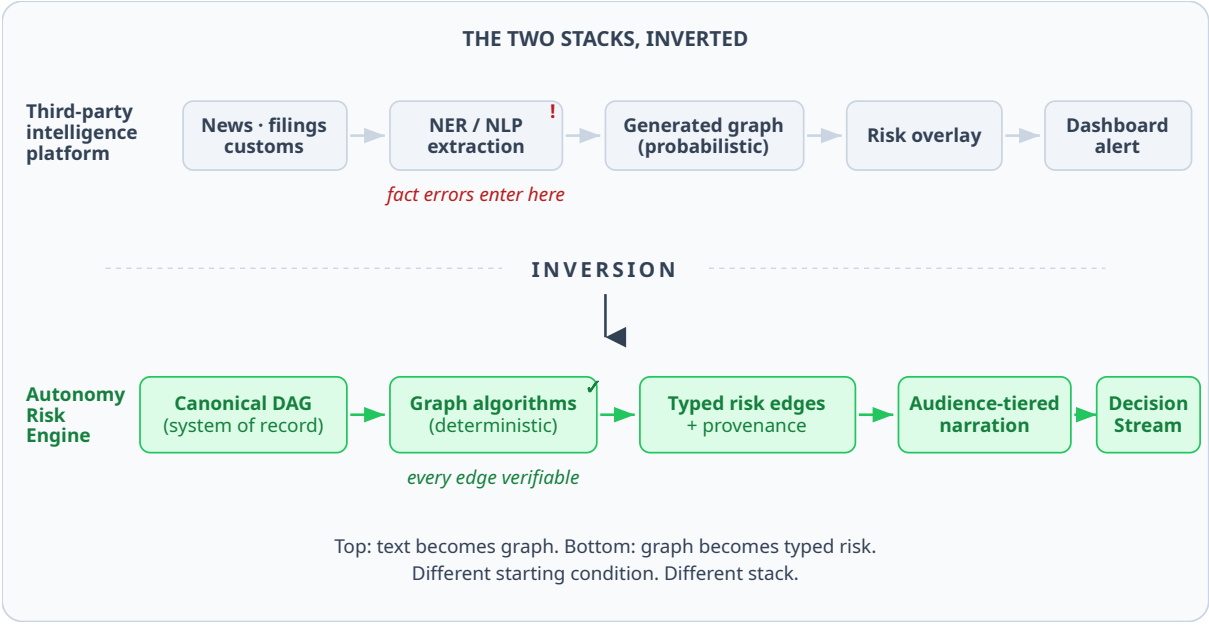
Resilinc, Everstream, Interos, Sphera, Sayari and the rest are doing this well; the work is real.

> *“ The graph is the output. The risk is the overlay. The customer is on the outside. ”*

The Autonomy story

Autonomy starts from the other direction. The canonical supply-chain DAG, sites, lanes, BOMs, suppliers, commitments, capacities, **already exists** as the system of record inside the platform. Every customer’s tenant carries it. It’s not extracted from documents; it’s the operational truth that drives planning.

So the problem inverts. We don’t need to *build* the graph. We need to *derive which relationships in it are dangerous*.



THIRD-PARTY PLATFORM

Text → graph → risk overlay

Hard NLP problem first. Risk second. Customer watches a dashboard.

AUTONOMY RISK ENGINE

Canonical graph → risk-edge derivation → narration

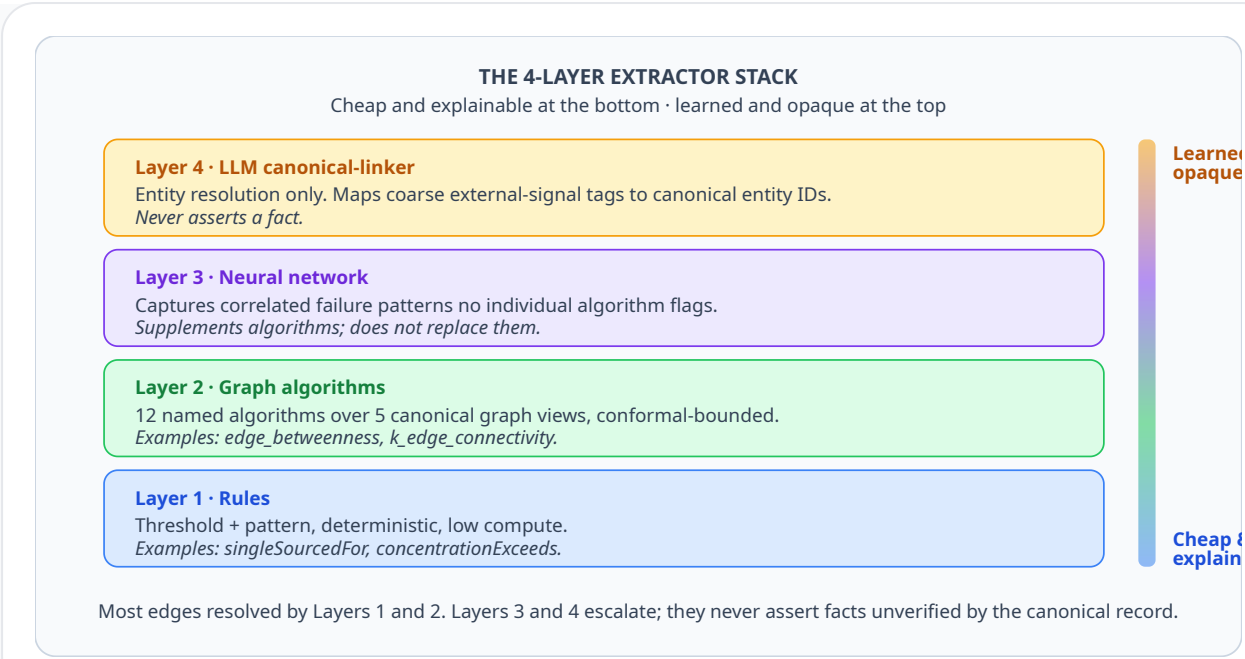
Graph is the operational truth. Risk is a typed edge. The agent acts on it.

What changes when the graph is given

The technique stack inverts

Building a graph from text means leaning on LLMs and named-entity recognition for the extraction itself. Errors there are *fact errors*: a wrong supplier name, a missed subsidiary relationship, a hallucinated edge. The architecture has to compensate downstream.

Starting from canonical state means classical graph algorithms, shortest path, betweenness, min-cut, k-edge-connectivity, topological cascade order, time-respecting paths, are the right *first* extractor. They're deterministic, explainable, training-free. They produce typed risk predicates over a graph whose entities you already know exist.



> *“ When the graph is given, "edge betweenness 0.42 on lane LAN-447" is a precise statement about a specific edge in a specific snapshot, not a probabilistic guess from a language model. ”*

NNs supplement algorithms at the tactical tier, capturing correlated failure patterns no individual algorithm flags. They don’t replace algorithms. LLMs play exactly two narrow, structured roles: **linker** (mapping coarse external-signal tags to canonical entity IDs) and **narrator** (turning structured records into prose for human consumption). They never assert facts.

The trust story is verifiable

When an LLM extracts “Supplier X depends on raw material Y from region Z” from a news article, you have a fact-shaped claim with no in-system grounding. The fact lives or dies on the LLM’s output. There’s no record to point to.

When the Risk Engine emits a structured edge, `criticalLaneFor(LAN-447, [customer-set])` with provenance, value score, evidence quote, and a versioned algorithm reference, and a narrator turns it into “Lane LAN-447 is your most fragile lane this week”, you have something verifiable. Every entity, number, and date in the prose must appear in the structured record. Post-hoc validation enforces this. Mismatch → narration regenerated.

LLM as fact source

"The model said it. We're showing it. Hope it's right."

LLM as narrator

"Here's the structured record. The prose is bounded paraphrase, validated against it."

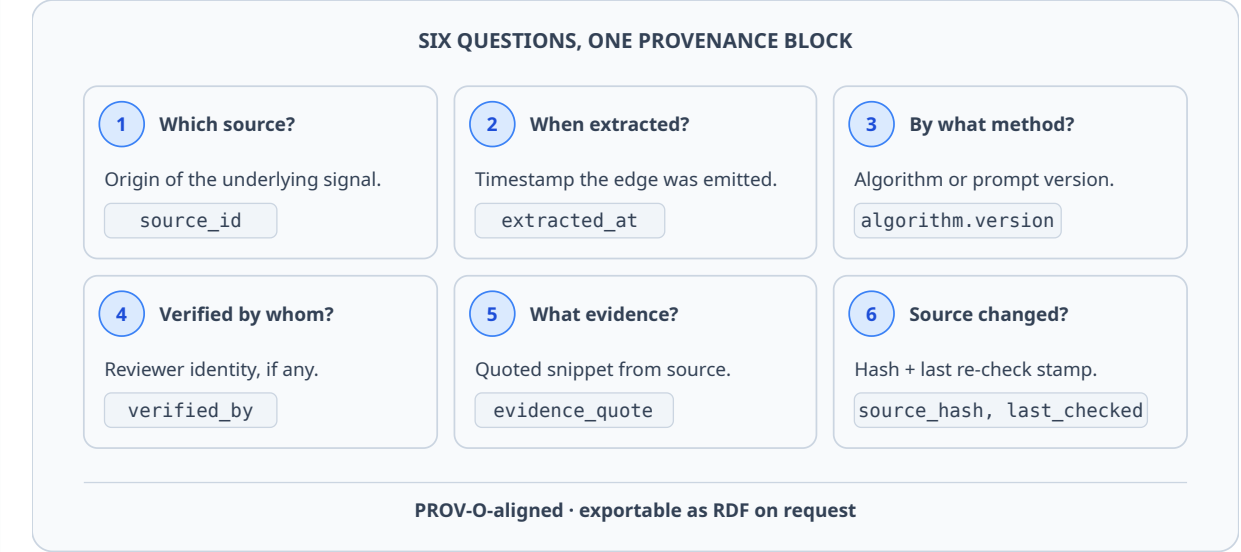
The audit story is real

Every Risk Engine edge carries enough provenance to answer six questions:

1. Which source generated this?
2. When was it extracted?
3. By what method (which algorithm version, which prompt version, which rule)?
4. Verified by whom?
5. What evidence supports it?
6. Has the source changed since?

Implementation is JSONB on the risk-edge row, not a separate triple store, but the field shape is PROV-O-aligned so we can export RDF when consumers ask. This is what lets a regulated customer treat the engine as

evidence in a SOC II audit, and it's what a third-party platform structurally cannot offer at the same fidelity, because it doesn't own the underlying graph.



The failure mode this avoids

> *" The dominant failure mode of LLM-backed knowledge graphs is letting the model assert a fact, then post-hoc justifying it. The Risk Engine treats that as a regression, not a feature. "*

The architectural posture is simple: **the structured record is always the source of truth**. Algorithms produce records. Rules produce records. The NN produces records. The LLM linker produces *candidate* records that go through validation, scoring, and human review before they reach the Decision Stream. The narrator produces prose *from* records.

At no point is the LLM's output the unbacked source of a fact a planner sees. That separation is what makes the system inspectable, debuggable, and, when the inevitable model-version change comes, measurably better or worse against a fixed benchmark.

What this means for buyers

If you already have a third-party risk-intelligence subscription, this is not an either/or. The Risk Engine consumes structured risk feeds, your existing intelligence vendor becomes a high-authority source flowing through the Context Engine into the value-scoring formula. Their work is amplified, not replaced.

What changes is the *action* side. Instead of a dashboard your planners watch, you get:

1. **Structured risk edges over your canonical state**

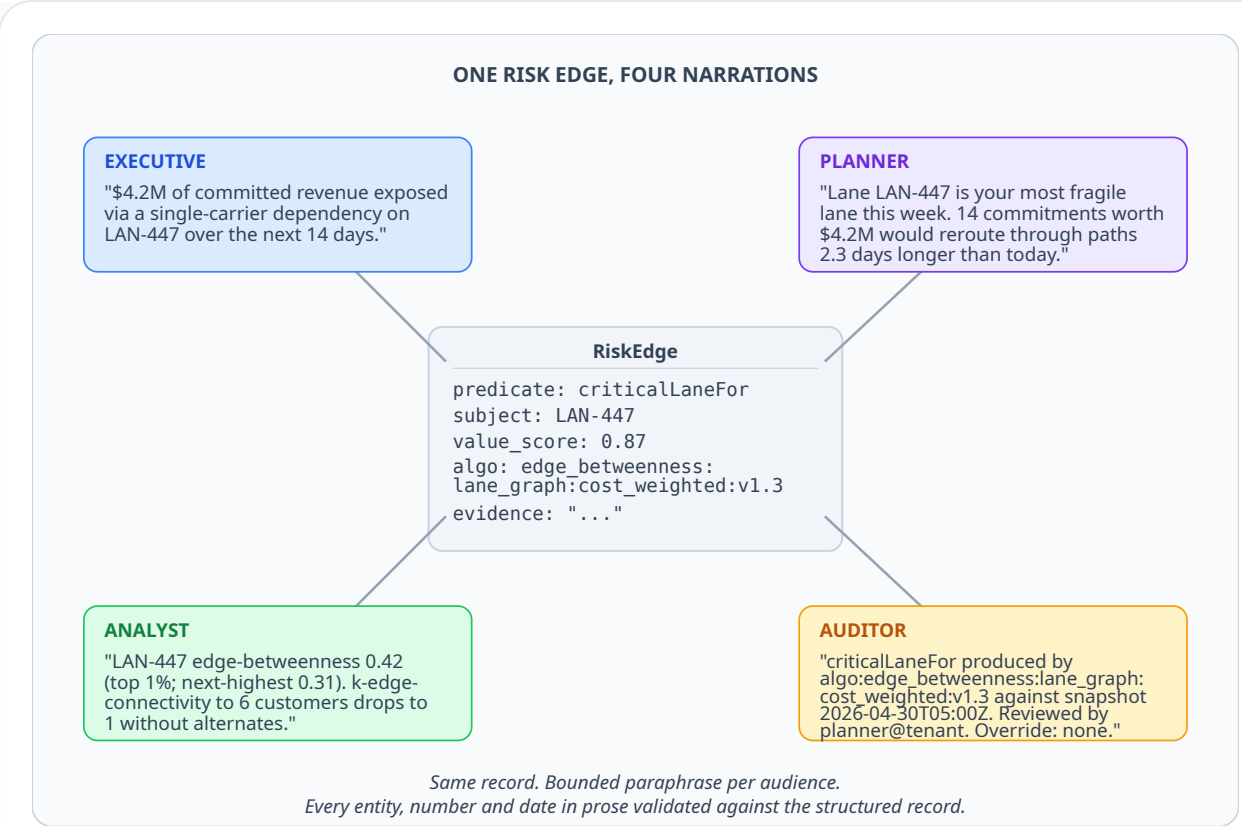
   Not "there's a risk in your supply chain." Specifically: "*this* lane, *these* commitments, *this* dollar exposure."

2. **Audience-tiered narration**

   Same edge, four narrations. Executive: dollar-anchored. Planner: action-anchored. Analyst: structural. Auditor: full provenance.

3. **Decision Stream routing, not dashboard alerts**

   High-value edges with full provenance can act under guardrails. Mid-value edges go to inspect. Everything sorted by *value*, not confidence.



Closing thought

The AI-native risk story everyone is telling right now is centred on language models extracting graphs from text. That story is true for vendors without the graph.

It’s the wrong story for a planning platform that already has it.

The Risk Engine is in beta today, sequenced behind the Context Engine event-awareness upgrade. We’ll write more as it lands.

See Autonomy in action

Walk through how Autonomy models, executes, monitors, and governs supply chain decisions with autonomous AI agents.

See It Live