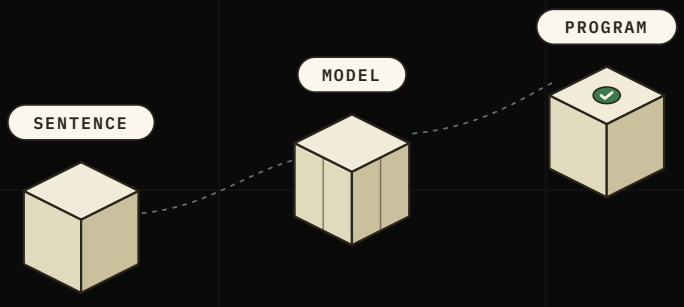
LOCI

TECHNICAL BRIEF · ARCHITECTURE

Inverting fraud detection for scale.

Three inversions that keep every decision readable at scale: compile the rules so evaluation is cheap, compute behavior live so freshness is free, and declare the features so nothing arrives unexplained.



language, then model, then machine
compiled once, replays exactly

TECHNICAL BRIEF

Inverting the fraud detection problem for scale

The industry scales fraud detection by adding things: more models, more compute, more copies of your data, more rules stacked on rules until nobody can say what the system actually does. Loci scales by inverting three assumptions instead. Each inversion removes work from the critical path rather than adding machinery to it, and each one keeps a property most platforms give up first: every decision stays readable by the person who has to defend it.

THE CHAIN WE BUILD ON

An analyst's sentence becomes an explainable model, and the model becomes a compiled program: language, then model, then machine. Everything below is that chain applied three times.

INVERSION ONE · MADIE, THE LOCI DETECTION ENGINE

The rule is a model, and the model is compiled

Most detection platforms treat a rule as either code (fast, unreadable by the risk team that owns the outcome) or as a formula built in a GUI (readable one at a time, interpreted from scratch on every decision). Both choices tax you at scale. Code accumulates into a system only engineers can change. Interpreted formulas re-read the recipe every time they cook, and the platform must keep a copy of your data close by just to make that re-reading fast enough.

MADIE inverts the definition. A Loci rule is an authored artifact written in FLM, a JSON language close enough to plain description that an analyst reads it directly: named evaluations, weighted like a model, with explicit conditions over the entity's history. At registration, not at decision time, the engine compiles that artifact into a verified, executable detection program. The compiler validates every field, every table reference, every time window before the rule can exist. A mistake dies at publish, on a screen, with a named error. It does not surface three weeks later on a live customer transaction.

What this buys at scale

Latency that holds
Evaluation runs an already-verified program over the entity's history. Rule evaluation stayed around 4 to 6 ms in benchmark testing, and nothing grows with rule complexity.

Decisions that replay
Each compiled plan is fingerprinted. A decision from last quarter re-runs under the same program and returns the same answer, which is what an auditor or a post-incident review needs.

Weights an analyst owns
Scoring is transparent arithmetic over the weights the author chose. Nothing is learned silently, nothing drifts. The rule is the model, and the model is on the record.

This is a deliberate constraint, not a limitation we apologize for. The engine refuses whole categories of cleverness (unbounded lookbacks, improvised operators, hidden score adjustments) because refusing them is what makes the compile-and-replay guarantee possible.

BENCHMARK NOTE

On a modest 6-vCPU environment, evaluating fraud and risk rules over a 60-day history window and entity histories of up to 500,000 transactions, the full gateway path sustained roughly 500 transactions per second at 21 to 49 ms p95, while rule evaluation itself stayed around 4 to 6 ms. Actual latency depends on rule shape, data distribution, storage, and topology; the result that matters is where the time is spent: rule evaluation remains a small slice of the full path.

INVERSION TWO · CUSTOM TABLES, AND THE REPLICA YOU NEVER BUILD

Bring the questions to the data the engine already has

The standard integration pattern in this market is replication: copy your operational data into the vendor's data model, keep the copy fresh forever, and let rules run over the replica. The copy is the scale problem. It is a second system of record with its own staleness, its own reconciliation burden, and its own bill.

Loci inverts the direction of travel. The engine's transaction history fills itself from the decision path: evaluating a transaction records it, so the aggregations every rule depends on (counts, sums, averages, deviations, percentiles over an entity's behavior) read the same history the hot path writes. There is no separate behavioral replica to hydrate, reconcile, or keep warm. A behavioral baseline in Loci does not drift from a batch feature pipeline; it is computed live from the engine-held history at evaluation time.

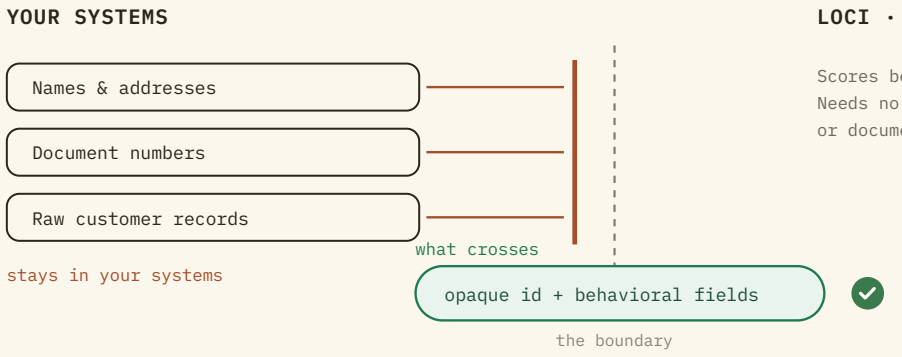
Custom tables carry only what genuinely cannot be derived from transactions, in three shapes:

Per-entity state
A lifecycle stage, a limit, a risk tier. One row per entity, read as a scalar inside a control.

Event streams
Append-only rows for sequencing: was there a password reset in the 24 hours before this transfer?

Reference lists
Membership checks for trusted counterparties or watched corridors. Edits change coverage instantly.

Every table and column a rule references is verified when the rule is registered. Typos fail at deploy time with a named error, the same compile-don't-hope discipline applied to data.



Personal data stays in your systems. Only an opaque identifier and behavioral fields cross into Loci, and the model that helps author detection never reads customer records; only pattern conclusions travel.

A BOUNDARY COMPLIANCE TEAMS NOTICE FIRST

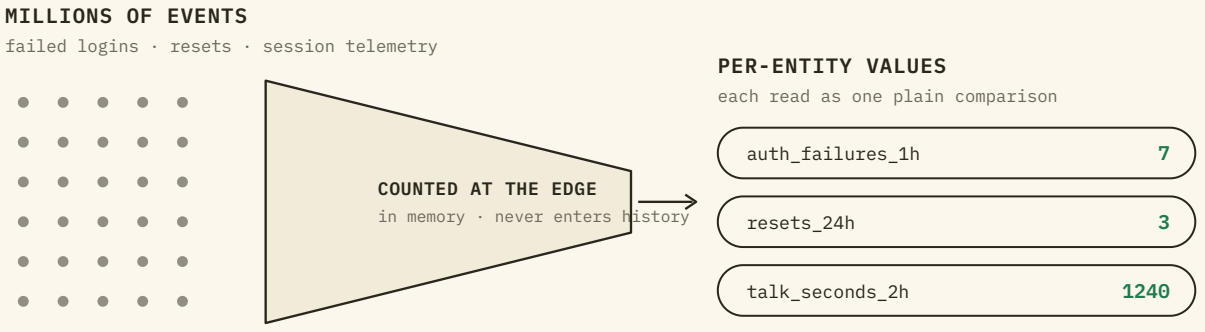
Detection runs on behavioral fields keyed to an opaque entity identifier; nothing in the engine requires a name, an address, or a document number, so personal data can stay in your systems while Loci sees behavior. The same boundary holds for AI: rules are authored from plain-language intent and a field schema, so the model that helps write detection never reads customer records. Even when Loci mines historical data for candidate patterns, only its conclusions travel, the predicates and their statistics, not the records beneath them. And because rules execute as compiled programs, no model sits in the decision path at all. Where regulators are not ready for customer data inside an LLM, that separation is structural, not a policy you have to trust anyone to follow.

INVERSION THREE · EVENT COUNTERS, FOR SHORT-LIVED ACTIVITY

Feature engineering with every feature on the record

Some of the strongest fraud signals never look like transactions: a burst of failed logins, repeated password resets, session telemetry from web and mobile channels. The conventional answers are both poor. Force these events into the transaction store and they poison the behavioral baselines, because an account that pings daily never looks dormant again. Build an ML feature pipeline and you inherit exactly the opacity the rest of the platform exists to remove.

Loci's event counters invert feature engineering into declared configuration. An organization declares its counters once, ingests events one at a time or in batches, and the gateway provisions the per-entity counter table that rules read as one more plain comparison. Each counter has a name, the event type it consumes, an aggregate (count, sum, or last-seen), and a rolling window. The platform counts arriving events at the edge, in memory, without touching the detection engine. Crossing a declared alert threshold publishes the value immediately; everything else lands within a stated freshness bound.



High-volume events are counted at the ingest edge into a handful of per-entity values, each read as one plain comparison. The events never enter transaction history, and the decision path never feels them.

The properties are the point

Authored features
Every counter is named, typed, versioned, filterable, and readable by the analyst who will use it, not a transformation buried in a pipeline repo.

Declared staleness
Each value carries its update time, and the bounds are part of the contract, never discovered after the fact.

Hot path untouched
Millions of events are absorbed at the ingest edge and compressed into thousands of per-entity value updates the decision path never feels.

Adoption without surgery
The ingest accepts batches with idempotent replay, so a first deployment can feed events from an existing scheduled job, real-time posting later.

COMPOSITION

How the inversions compose

A single rule can now draw on four kinds of evidence in one place: live aggregations over transaction history, counter values for short-lived activity, existence checks against event streams, and membership checks against reference lists. All four arrive through the same compiled evaluation, and all four are legible in the result: which evaluations passed, at what weight, producing what score.

Above the engine, decision policy stays deliberately separate from detection fact. Thresholds, escalation, and allowlist handling are organization-level configuration, editable without touching a single rule, and every outcome explains itself: what fired, what policy applied, what the effective decision was.

Detection says what is true.
Policy says what to do about it.

The claim, plainly

Scale in fraud detection is usually purchased with opacity: more inference, more replication, more machinery between the analyst and the outcome. Loci's bet is the opposite, and it is structural rather than rhetorical. Compile the rules so evaluation is cheap and repeatable. Compute behavior live so freshness is free. Declare the features so nothing arrives unexplained. The result is a platform that gets faster to operate as it grows, because everything in it, from a rule to a counter to a decision, was authored by someone, is accountable to someone, and stays on the record.

Loci exists so that every risk decision a machine makes can be read, defended, and re-run by the human who asked for it. The three inversions are how that survives contact with scale.

SEE THE INVERSIONS ON YOUR DATA

A short technical walkthrough of how a rule compiles, what your own history already answers, and where counters fit. Read this brief online at runloci.com/brief/inverting-fraud-detection-for-scale.

LOCI FRAUD AI · TECHNICAL BRIEF · JULY 2026 runloci.com · sales@runloci.com