ON CONNECTORS, CANONICAL FORM AND THE COST OF A NEW SOURCE

The integration is where the model gets decided

Nobody buys a platform for its connectors, and the connector layer quietly determines what every question above it can ask. Fields dropped at ingestion are dropped for the life of the record.

IN BRIEF

Integration is treated as plumbing and priced as a feature count. It is neither: it is the point at which a vendor's data model is imposed on your estate, and the decisions made there are irreversible in practice. This paper sets out what a canonical event model has to preserve, why “number of integrations” is close to meaningless as a comparison, what a connector must state about its own fidelity, and the governance problem created by letting third-party code run inside a security platform.

1 · Mapping is lossy, and the loss is silent

A source emits records with sixty fields. The platform's schema has room for twenty-two. A connector author, working from a sample capture and no knowledge of which query will matter in eighteen months, chooses the twenty-two. Nothing anywhere records the thirty-eight.

This is the ordinary case, not a failure of care. The author cannot know that the field describing which management console initiated a change will be the field that distinguishes an administrator's routine work from an adversary using the same tool. And when that question is finally asked, the answer is not “we do not have that field” — the query simply returns a result that looks complete and is wrong.

Two consequences follow, and only the second is usually discussed. The first is that **capability is bounded at ingestion**: no amount of analytical sophistication recovers a discarded attribute. The second is lock-in — a platform that normalised aggressively holds your history in a shape only it can interpret, which turns a technical preference into a commercial position.

FIGURE 1 · TWO WAYS TO ACCEPT A RECORD

NORMALISE ONTO A SCHEMA

60 fields → 22 columns. The other 38 are discarded at the boundary. Storage is efficient, queries are uniform, and the estate's own vocabulary is gone. Adding a field later means re-ingesting history you no longer have.

CLASSIFY AND KEEP

A common envelope — class, subject, time, severity — plus the source's own attributes retained beside it. Correlation works on the envelope; the specifics survive for the question nobody has asked yet.

The right column is not free — it costs storage and it makes query authoring harder. The claim is narrower than "keep everything": keep what the source said, and be honest in the interface about which attributes are common and which are source-specific.

2 · What the envelope has to carry

If domain attributes are preserved rather than flattened, the common envelope can be small. Four things have to be in it, and each earns its place by being required for correlation rather than for display.

Class. Domain, category, type, activity — enough to ask "show me every authentication event" across sources that agree on nothing else. Classification is the one place normalisation is unambiguously right.

Subject. The entity the record is about, resolved to a stable identity rather than whatever string the source used. This is the hard part of integration and the part that makes graph correlation possible at all.

Two times. When the thing happened and when the platform learned of it. Collapsing these into one field destroys the ability to reason about detection latency, and makes a delayed batch source indistinguishable from a live one.

Severity, or an explicit absence. Most records are observations nobody judged, and "not judged" must be a value rather than a zero — otherwise the un-alarmed majority becomes invisible to exactly the queries that need it.

Identity resolution is the integration's real product. A connector that delivers perfect records against unresolved subjects has delivered a searchable log, not a graph. Ask how a source's user string becomes the same subject as your directory's — and what happens when it cannot.

3 · Why the integration count is a bad number

“Four hundred integrations” is the standard claim and it survives because it is unfalsifiable. Three things it does not distinguish:

3.1 · Reading a log is not covering a product

A connector that parses one log format from a product with eleven telemetry channels counts as one integration and covers a fraction of the product. The honest unit is not the vendor name; it is the record types actually consumed, and a coverage sheet should be able to list them.

3.2 · Ingesting is not resolving

A source whose subjects never resolve to your directory contributes rows that cannot participate in a cross-domain path. It is in the count. It is not in the graph. This is the single most common gap between what an integration claims and what it delivers.

3.3 · Read is not write

An integration that can observe a system but not act on it cannot participate in response, which means the automation story stops at the boundary of whichever products were only ever read. Counting both directions as one number hides the half that matters when something is on fire.

FIGURE 2 · WHAT A CONNECTOR SHOULD DECLARE ABOUT ITSELF

`record types consumed`	named, not counted — and which of the source's channels are *not* read
`subject resolution`	how a source identifier becomes a graph subject, and the failure behaviour
`latency profile`	streaming, polled at an interval, or batch — the difference decides which detections are possible
`write actions`	which response verbs it supports, each with a reversibility class
`fidelity note`	what this connector is known to lose, stated by its author

The last row is the unusual one and the most useful. A connector author knows what they dropped; writing it down turns an invisible ceiling into a documented limitation someone can plan around.

4 · Third-party code inside a security platform

An extensible integration plane implies a marketplace, and a marketplace means code written by someone else executing inside the system that holds your entire estate's telemetry. The convenience is real and so is the exposure, and the mitigations are not exotic: signed modules, a declared capability set enforced by a sandbox rather than by policy, no ambient credential access, and a resource budget so a badly written module degrades itself rather than the pipeline.

The harder question is governance rather than isolation. Who reviews a module, on what schedule, and what happens when a popular one is abandoned? A marketplace whose listings never expire accumulates unmaintained code with privileged placement — and "widely installed" becomes an argument for keeping it that has nothing to do with whether it is still safe.

> **The unifying idea.** An integration is a claim about fidelity — what it reads, what it resolves, what it can do, and what it loses. A connector catalogue that states those four things is a specification. One that states a number is marketing.

5 · Questions worth asking, whoever you are buying from

01 For one source I care about: which record types do you read, and which of its channels do you not?

02 What happens to a field your schema has no home for — dropped, or retained beside the envelope?

03 Do you keep observed and recorded time separately, and can I query the gap between them?

04 How does a source's user string become the same subject as my directory's, and what if it cannot?

05 Which of your integrations can act, not merely observe? Show me that list separately.

06 Does each connector publish a fidelity note saying what it loses? If not, why not?

07 What can a marketplace module reach, and is that enforced by a sandbox or by a policy document?

08 If we leave, in what form does our history leave with us — and can anything else read it?

6 · Where this sits in the series

Paper 02 argued that an alert should be a subgraph. Paper 03 required change records and entitlements as first-class sources. Paper 04 depended on retained events being rich enough to backtest against. Paper 06 asked what the organisation can prove years later. Every one of those claims is settled, quietly, at the ingestion boundary.

Which is the reason this paper exists at all. Integration is the least discussed layer and the one with the longest reach, because it decides what the rest of the platform is permitted to know.

> *A field discarded at ingestion is not missing from your queries. It is missing from your options.*

EventLake

ABOUT THIS PAPER

Paper 09 in a series on the architecture of active defense, published by **EventLake Research**. Papers 01–08 cover enforcement at the edge, explainable alerts, legitimacy, compiling findings into rules, bounded autonomy, evidence, continuous trust, and coverage honesty. The series is deliberately vendor-neutral, and the questions in §5 are meant to be asked of us as readily as of anyone else.

EventLake classifies rather than flattens: a common envelope for correlation, the source's own attributes retained beside it, observed and recorded time kept apart, and a fidelity note per connector. eventlakes.com