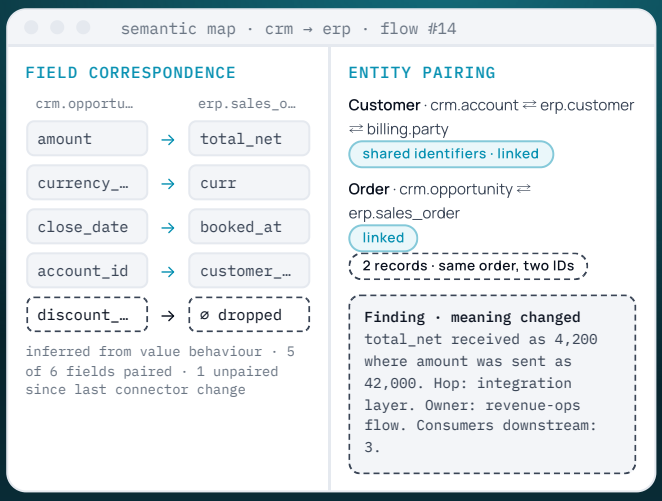
See what your data means, not just that it moved

Every integration ships with a map: which CRM field becomes which ERP field, which billing identifier is the same customer as the one in the warehouse. The map is declared once, at build time. It goes stale on the next connector update, and no tool checks it against the data afterwards.

A schema says a field is a three-character string. It does not say it is a currency code, or that it must agree with the amount beside it. **Schema checks, pipeline tests and APM all pass while the meaning of a record changes between one system and the next.** That is where decimals move, currencies swap, and the same order lands twice under two IDs.

mmune builds the map itself. It samples the landing tables and queues on both sides of your integration layer on a short cycle, fingerprints each field by how it behaves, and infers what it is and which field in the neighbouring system corresponds to it. No declarations, no rules to author. **The result is a live semantic map of the estate:** which fields mean the same thing, which records across CRM, ERP and billing are the same business object, and who owns each flow. Drift detection, reconciliation and duplicate detection all read from that one map.



ILLUSTRATIVE MOCK-UP · NOT PRODUCT OUTPUT · FIGURES ARE CONSTRUCTED

With the map learned from the data rather than declared at build time, mmune turns field-level meaning into five working capabilities:

Learned baselines, not declared rules

A contract baseline for every monitored system, established with no per-integration setup, and rebuilt as the data changes.

Rename told apart from removal

A field that moved under a new name is noise. A field that disappeared breaks a ledger. mmune knows which is which.

Cross-system entity pairing

Identifier columns are fingerprinted and compared across integrations to link records nobody declared as the same thing.

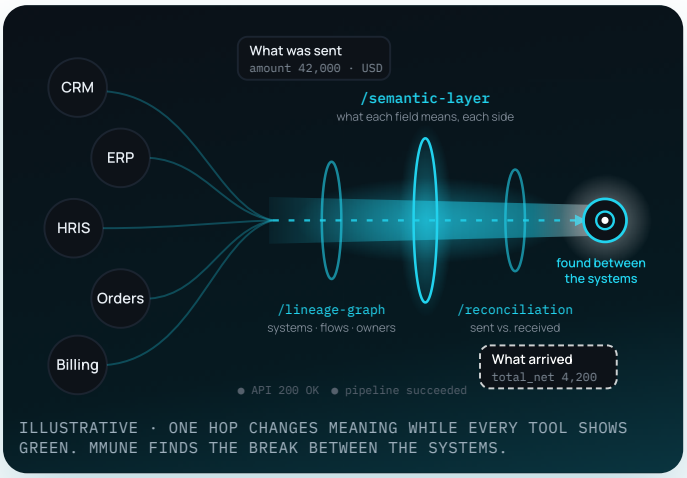
Duplicates by meaning, not by key

The same order or claim recorded twice under different IDs, caught by what the records mean, which is why it gets past existing dedup.

Blast radius through lineage

When a field changes, the map resolves which downstream consumers it feeds, so you know who is affected before the tickets arrive.

© 2026 mmune



One map, built from where the data lands

mmune does not sit in the path and does not read the wire. It reads the tables and queues that data lands in, on a short cycle, on both sides of the integration layer. From the landing points it works out which system feeds which, through what, and who owns each flow, and it keeps that picture current when an integration changes.

Meaning is inferred from behaviour, not from documentation. Value ranges, formats, cardinality and the identifiers a field shares with other systems are enough to say what a field is and what it corresponds to on the other side. The map is rebuilt from the data, so it does not drift away from reality the way a spreadsheet does.

No code change, nothing in the flow, no added latency. mmune keeps metadata and aggregates, not the payloads.

Why a declared map goes stale

WHAT IT COSTS TODAY

- **Months of schema mapping** on every migration, cutover and acquisition, redone for each pair of systems.
- **Mappings written for yesterday's API version.** The next vendor update changes what flows through them; the iPaaS still shows them as valid.
- **A map nobody re-validates.** Lineage tools show which table feeds which; none checks that a field still means the same thing on the other side.

WHAT MMUNE DOES INSTEAD

- **Learns the map from the data** instead of reading a declaration, and rebuilds it as the data changes.
- **Compares what was sent with what arrived** field by field, using the inferred correspondence, and flags a disagreement within minutes.
- **Routes the finding to the owner** with the hop where it broke and the downstream consumers affected, in Slack or Jira.

A trust verdict before an AI agent acts

The same map answers a question your AI initiatives are about to ask: **can this table be trusted right now?** mmune exposes a data-trust check over MCP, so an agent can ask whether a table and the flow feeding it are currently healthy before it reads from it.

This does not stop a model from confabulating. It stops an agent from acting on data mmune already knows is broken: a dropped field, a swapped currency, an order present twice. **Fewer tokens spent reasoning over corrupted inputs, and fewer automated actions taken on a number that was wrong before the agent ever saw it.**

mmune is an AI-native data integrity layer for data in motion between enterprise systems. It sits read-only next to the integration stack you already run, learns what each field means and which records are the same business object across systems, and tells the owning team within minutes when what arrived no longer means what was sent. Keep your iPaaS, APM and data-quality tests; mmune covers the part between them. To learn more, visit mmune.io

© 2026 mmune