

Making Enterprise Data Agent-Ready: A Framework for Context, Governance, and Trust



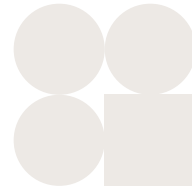


Table of Contents

Executive summary	3
The problem: agents cross platforms, governance doesn't	4
Why the platforms haven't solved it	6
MCP makes it easier to connect, not safer	7
AI-ready isn't agent-ready: the consumer changed	8
The step teams miss	10
The framework: context, governance and trust	11
Why all three have to travel together	15
The mechanism: give agents a contract, not a connection string	16
We call these data products	17
What changes when agents are the consumer	18
From data products to an ontology: from reading to acting	19
How DataOS makes data agent-ready across platforms	20
What that looks like in practice	21
Why "across platforms" is the point	23
Built to be owned, not rented	23
Where to start: one agent, one cross-platform use case	24
How this plays out by industry	25
Conclusion	26
FAQs	27
Sources	29

Executive summary



Most advice on getting enterprise data ready for AI agents assumes your data lives in one place. But it generally doesn't. And that gap is where agent programs quietly stall.

In 2026, the problem hard to ignore. IBM's 2026 Tech Leader Study found enterprises detected an average of 54 AI agent incidents in a year, and 37% of them exposed data. Gartner now predicts that by 2028, a quarter of enterprise security breaches will trace back to AI agent abuse. And on October 20, Gartner named both AI context platforms and AI agent identity among its top strategic technology trends for 2027.

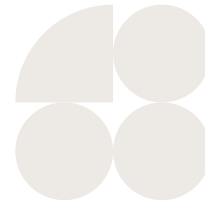
Snowflake, Databricks, Microsoft and Google have each taken steps to address this, shipping some version of governed context and agent controls. These are real improvements. **But each one governs the agent inside its own walls, and the agents enterprises are building don't stay inside one wall.** A single task might pull a policy from one system, claims from another and a customer record from a CRM.

This paper makes three arguments:

- 1. AI-ready is not agent-ready.** Data that was good enough for a chatbot or a dashboard is not good enough for software that reads, decides and acts on its own.
- 2. An agent needs three things from data: context it can understand, governance it can't get around, and trust it can verify.** Miss one and the other two don't hold.
- 3. Those three have to travel together, in a single governed interface the agent consumes, regardless of which platform the data sits on.** We call that interface a data product. You can call it whatever you like; the label matters far less than the contract. On top of those products, an ontology connects them into one model of the business, including the actions agents are allowed to take, so agents can reason and act across domains without knowing which platform holds what.

If you lead data, AI or architecture and you're being asked to put agents in production next year, this is written for you.

The problem: agents cross platforms, governance doesn't



Agents are moving into production faster than the controls around them. Surveys of data, security and technology leaders bear this out.

Finding	Source
Enterprises detected an average of 54 AI agent incidents in the past 12 months. 37% caused data exposure or a breach, 33% triggered cascading system failures, 17% created compliance issues.	IBM Institute for Business Value, 2026 Tech Leader Study (2,000 tech leaders, June 2026)
77% of tech leaders say AI adoption is outpacing their governance. Only 11% of CIOs and CTOs feel completely prepared for the scale of agents coming in the next year.	IBM IBV, same study
65% of organizations had at least one security incident caused by AI agents in the past year; 61% of those involved data exposure.	Cloud Security Alliance and Token Security , April 2026
75% of data leaders say governance hasn't kept pace with AI adoption, while roughly half have already moved into agentic AI.	Informatica CDO Insights 2026 (600 data leaders)
48% have not implemented programs to govern the data used for AI.	Precisely and Drexel LeBow, 2026 State of Data Integrity and AI Readiness (505 leaders)

Finding	Source
By 2028, 25% of enterprise security breaches will trace back to AI agent abuse, driven by unmanaged machine privileges and legacy credentials.	Gartner Top Strategic Technology Trends for 2027 , October 2026
Over 40% of agentic AI projects will be canceled by the end of 2027 due to escalating costs, unclear business value or inadequate risk controls.	Gartner , June 2025

A caveat on these numbers: most are self-reported, several surveys are vendor-sponsored, and "incident" means different things in different studies. We wouldn't hang a business case on any single figure. But when IBM, the Cloud Security Alliance, Informatica and Gartner all point the same way, the direction is hard to argue with.

Why the platforms haven't solved it



To be fair, the major platforms saw this coming. At Summit 2026, Snowflake shipped [per-agent identity, agent audit trails and a context capability inside Horizon](#), and announced the acquisition of an MCP gateway. Databricks, Microsoft Fabric and Google BigQuery have made similar moves around governed semantics and agent access. Inside a single platform, these are real improvements.

The catch is the phrase inside a single platform. Ask a claims agent at an insurer to explain why a renewal premium jumped, and it needs the policy record, the claims history, the agent hierarchy and maybe a third-party risk score. In most enterprises we work with, those don't live in one system. Even a vendor whose business depends on Snowflake and Databricks [acknowledges](#) that platform-native context is strong within the platform but doesn't resolve entities across systems.

So you end up with several governance islands, each with its own identity model, its own semantic definitions and its own audit trail. The agent is the only thing that touches all of them. That's backwards: the least trusted actor in the architecture has the widest view.

The industry knows this. The [Open Semantic Interchange effort](#), now Apache Ossie (incubating) since July 2026, exists precisely because of what it calls metric drift, manual translation, AI hallucinations from conflicting business logic, and integration debt. A standard for exchanging definitions is welcome. But exchanging definitions isn't the same as enforcing them at the moment an agent asks a question. And metric definitions alone don't tell an agent how a policy relates to a claim, a claim to a customer, or what it's allowed to do with any of them.

MCP makes it easier to connect, not safer



The Model Context Protocol has become the default way agents reach tools and data. But Gartner's April 2026 analysis was blunt: MCP ["was built for interoperability, ease of use and flexibility first"](#), and agent permissions are "not inherited from human user roles." Gartner calls the combination of sensitive data, untrusted content and external communication in one workflow a "no-go zone."

The gap is structural. MCP standardizes how an agent finds and calls a tool. It says nothing about what the data behind that tool means, whether it's current, or whether this particular agent should see it. Those decisions fall to whoever built each server, so an enterprise running a dozen MCP servers can end up with a dozen different answers to the same access question. Faster connections without shared controls spread the risk across more endpoints.

Gartner recommends assigning domain owners to MCP servers so they can define and enforce guardrails before any client touches the data. In other words, put an owned, governed interface in front of the data, which is the approach this paper builds toward.

AI-ready isn't agent-ready: the consumer changed



For the past two years, "AI-ready data" mostly meant clean, well-documented data that a model could be trained on or retrieve from. Gartner's original definition centered on data being representative of the use case and qualified through metadata. That was the right bar for copilots and RAG apps, where a human reads the answer before anything happens.

Agents change who the consumer is. Gartner's August 2026 research says it plainly in the title: [AI-ready data needs to expand to agent-ready data](#). A separate Gartner note from July describes AI agents as [rapidly becoming the primary consumers of data](#).

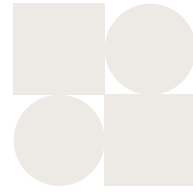
What's different when the consumer is software acting on its own?

	AI-ready (Copilots, RAG, BI assistants)	Agent-ready
Who checks the answer	A human, before acting	Often no one, until after
What the AI does with data	Reads and summarizes	Reads, decides, and writes back or triggers actions
How meaning is resolved	The human fills in gaps ("oh, they mean net revenue")	Must be explicit; the agent guesses otherwise
How relationships are known	The human infers them ("this claim belongs to that policy")	Modeled explicitly, so the agent can follow them across systems
Access model	Inherited from the user who asked	Needs its own identity, scoped to the task and revocable

	AI-ready (Copilots, RAG, BI assistants)	Agent-ready
Scope of data	Usually one app or one platform	Routinely spans systems in a single task
Failure mode	A wrong answer someone catches	A wrong action that propagates

The last row is the one that keeps CIOs up at night. A human analyst who sees a revenue number that looks off will stop and ask. An agent [trusts bad data confidently](#), as Thoughtworks put it in their [August 2026 piece](#) on the same topic, and moves onto the next step.

The step teams miss



Most teams treat agent readiness as a model problem or a prompt problem. They tune retrieval, add guardrail prompts and write evaluation suites. All useful. But Gartner's newest estimate puts a number on what they're leaving behind: organizations that prioritize semantics in AI-ready data could [raise agentic AI accuracy by upto 80% and cut costs by upto 60%](#) by 2027.

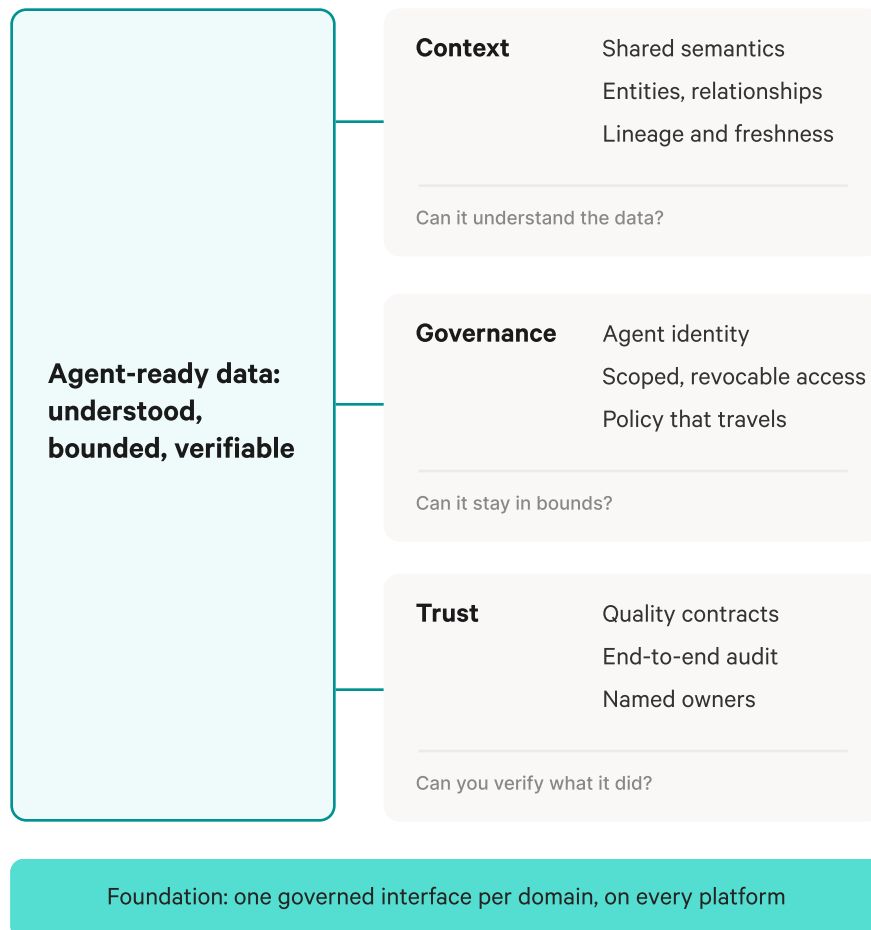
I'd treat the exact figure carefully; "upto" is doing a lot of work there. The direction, though, matches what we see in the field. The cheapest accuracy and cost gains for agents usually come from the data layer, not the model layer.

The framework: context, governance and trust



Strip away the vendor language and an agent needs three things from enterprise data. Each one answers a question the agent can't answer for itself.

An agent needs context, governance and trust, delivered together



Each pillar answers one question the agent can't answer for itself; the foundation is what keeps them consistent.

1. Context: Can the agent understand what it's looking at?

Context is business meaning made explicit and machine-readable. Not a wiki page about what "active customer" means, but a definition the agent queries through, so it can't compute it any other way.

In practice, agent-ready context includes:



Shared semantics. Metrics, dimensions and entities defined once, versioned, and reused across every consumer. If finance and the agent compute "net written premium" differently, one of them is wrong in front of a customer.



Entities and relationships across systems. The customer in the CRM, the account in billing and the member in the loyalty system are the same person, and the agent knows how that person relates to their policies, claims and orders. That's an ontology: a model of the business the agent can navigate, not a list of tables.



Lineage and freshness. Where a value came from, when it was last updated, and whether it's still within its SLA.



Usage guidance. What this data is for, what it isn't for, and the known caveats. This is the part humans carry in their heads and agents don't have.

Ask yourself these:

- If you pointed an agent at your three most-used data sources today, would it compute your top five KPIs the same way finance does?
- When the same entity appears in two platforms, who decides which record is authoritative, and could an agent follow its relationships from one platform to the other?
- Could an agent tell, without asking anyone, that a table is stale?

2. Governance: Can the agent only do what it should?

Governance for agents is enforcement, not documentation. A catalog that describes a policy is useful to people. An agent needs the policy applied at the moment it queries.

Agent-ready governance includes:



Agent identity. Each agent has its own identity, distinct from the developer who built it and the user who invoked it. Gartner put [AI agent identity](#) on its 2027 trends list for exactly this reason.



Scoped, short-lived access. Least privilege, granted for the task, with a clear way to revoke it. One survey cited by Kiteworks found [60% of organizations can't terminate a misbehaving agent](#). I'd treat that as directional, but it should make you uncomfortable.



Policy that travels with the data. Masking, row-level filters and purpose limits enforced the same way whether the agent hits Snowflake, Databricks or Postgres.



Action boundaries. Clear rules about which agents can only read, which can write, and which writes need a human in the loop. These work best when they're defined on business objects (approve a claim, update a member's address) rather than on raw tables.

Ask yourself these:

- How many different places would you need to update to revoke one agent's access to customer PII?
- If an agent joins data from two platforms, whose masking rules apply to the result?
- Do your agents run on a service account that has more access than any single human user?

3. Trust: Can you verify what the agent did and why?

Trust is what lets you give an agent more autonomy over time. It comes from evidence, not intent.

Agent-ready trust includes:



Quality contracts. Schema, quality and freshness checks expressed as code, with bad data quarantined before an agent can reach it.



End-to-end audit. Every agent query and action traceable to an identity, a policy decision and the data version it used.



Observability. Usage, cost and accuracy of agent access, tracked the same way you track any production service.



Clear ownership. A named owner for every dataset an agent depends on. Unowned data drifts, and agents spread the drift faster than people do.

Gartner's January 2026 prediction frames the stakes well: by 2028, [50% of organizations will adopt zero-trust data governance](#), because "organizations can no longer implicitly trust data."

Ask yourself these:

- If a regulator asked why an agent denied a claim last Tuesday, could you reconstruct the exact data it saw?
- Do quality failures stop agents, or just raise alerts nobody reads?
- Is there a named owner for every source your first production agent touches?

Why all three have to travel together



Here's the part most frameworks skip. Context, governance and trust are usually owned by different teams and stored in different tools: semantics in a BI layer, policies in each platform's access controls, quality in an observability tool, lineage in a catalog.

That's workable when the consumer is a person who can stitch it together. An agent can't. If the definition of "premium" lives in one tool, the masking rule in another and the freshness check in a third, the agent gets whichever pieces happen to be wired to the endpoint it calls. Usually that's the data, and not much else.

So the real design question isn't "Do we have context, governance and trust?" Most enterprises have some of each. It's "Does the agent get all three, together, every time it touches data, on every platform?"

The mechanism: give agents a contract, not a connection string



If context, governance and trust have to arrive together, they need a container. The unit an agent consumes can't be a raw table or a database connection. It has to be a governed interface that bundles everything the agent needs to use the data correctly.

That interface has five parts:

- 1 **The data itself**, served from wherever it lives, without forcing a copy into yet another platform.
- 2 **Its meaning**: semantic definitions, entities and relationships the agent queries through.
- 3 **Its policy**: who and which agents can access it, with what masking and filters, for what purpose.
- 4 **Its guarantees**: quality checks, freshness SLAs and the version the agent is reading.
- 5 **Its owner**: a named person or team accountable when it breaks.

Put those five together and expose them through the protocols agents already use (SQL, APIs, MCP), and the agent no longer has to guess. It asks for "customer lifetime value" and gets the governed number, the policy applied for its identity, and the lineage behind it.

We call these data products



Yes, that's the term. I know it carries baggage. For a lot of teams, "data product" still means a data mesh reorganization that took two years and stalled. Thoughtworks' [State of Data Mesh in 2026](#) concedes the term is "overused and inconsistently defined."

But set the label aside and look at what the interface above actually is: a versioned contract with an owner, semantics, policy and quality guarantees. That's what a well-built data product was always supposed to be. Agents just made the case for it impossible to ignore.

The evidence is starting to show up:

- Organizations with company-wide data products are 3.4x more likely to move AI projects into production, and 77% of them have agentic systems in production versus 23% of those without, according to a [BARC study commissioned by Actian](#) (300+ data leaders, April 2026). Adoption rose from 48% to 69% in about a year.
- IBM found organizations with governance designed into their architecture, rather than bolted on through manual review, [deploy 16x more agents](#) than their peers.
- Thoughtworks' guidance for agentic AI recommends exactly this shape: [owned data products with contracts](#), with meaning defined in one place and agents using business capabilities rather than raw schemas.
- And Gartner's own fix for MCP risk, quoted earlier, is to give MCP servers domain owners who enforce guardrails before any agent touches the data. That's a data product with a different name.

Two honest caveats. The BARC study is vendor-sponsored, and both it and IBM report correlation, not causation; the organizations that adopt data products may simply be more mature to begin with. Still, the pattern is consistent across sources that don't share a commercial interest.

What changes when agents are the consumer



Data products built for dashboards and analysts were mostly read-only tables with a catalog entry. Agent-ready data products need a few things those didn't:

- **Machine-readable semantics**, not just column descriptions.
- **Agent-aware policy**, so access is evaluated per agent identity and purpose, not just per human role.
- **Protocol-native access**, including MCP, so the governed interface is the easiest path for an agent developer, not the hardest.
- **Usage and cost telemetry per agent**, because agents query far more often than people do.

That last point matters more than it looks. If the governed path is slower or more awkward than a direct connection, developers will route around it. The interface has to be the path of least resistance.

From data products to an ontology: from reading to acting



Data products answer one question well: can I trust this data? But they're shaped around domains, and agent tasks aren't. "Why did this member's premium jump?" touches policy, claims, billing and a third-party risk score. "Approve this claim" needs to know what the claim is attached to, who's allowed to approve it, and what happens next.

That's the job of an ontology. It models the business itself: the entities that matter (customer, policy, claim, order), how they relate to each other, and the actions that can be taken on each. Each entity maps onto the governed data products underneath, wherever they live.

Three things change when agents work through an ontology instead of straight against data products:

- **Agents navigate the business, not the schema.** An agent asks for a customer's open claims on active policies. It doesn't need to know which platform holds which table or how to join them.
- **Actions become governed objects.** "Approve claim" or "update address" is defined once, with its own rules about who can do it and when a human must step in. That turns the hardest part of agent readiness, letting agents act and not just read, into something you can control and audit.
- **Every agent builds on the last.** The second agent reuses the entities, relationships and actions the first one needed. The model of the business grows one agent at a time.

Order matters here. An ontology built on raw tables inherits every quality and access problem underneath it, and one built from scratch tends to turn into a multi-year modeling exercise. Built on governed data products, it inherits their semantics, policy and quality guarantees from day one.

This is also where the market is heading. Gartner's newest trend research describes [AI context platforms](#) as combining scattered enterprise knowledge into governed, machine-readable context for agentic AI. An ontology on top of data products is a practical way to get there.

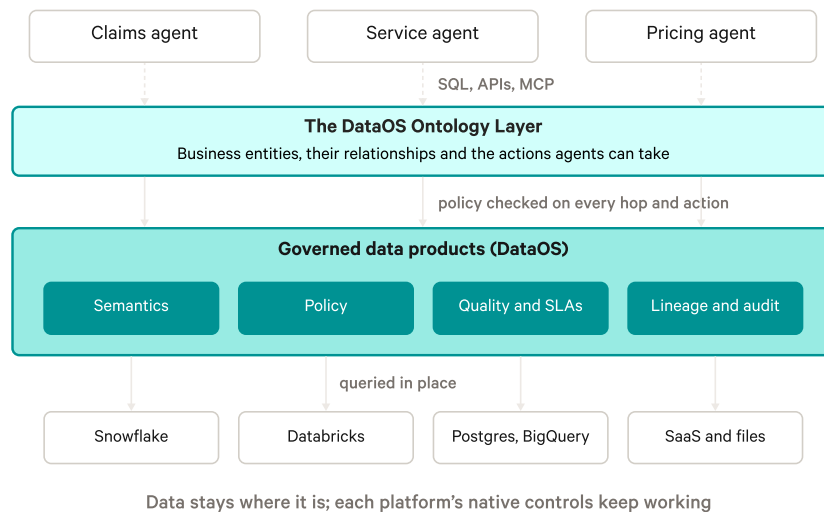
How DataOS makes data agent-ready across platforms



None of the framework above requires a specific platform, to be clear. You can assemble it from a catalog, a semantic layer, a policy engine, a quality tool and an MCP gateway, and some teams do. The hard part is keeping five tools in agreement about the same data, for every agent, on every platform, as things change.

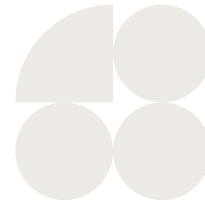
DataOS was built to make that bundle the default rather than an integration project. It sits above the platforms you already run and turns data in them into governed data products that people and agents consume through the same interface. The DataOS Ontology Layer then connects those products into a model of the business that agents can navigate and act on.

Agents act through one ontology, on governed data, across every platform



Agents never hold a direct connection to a platform; every request passes through the same semantics, policy, quality checks and audit.

What that looks like in practice



Framework need	Data product (Technical)	DataOS Ontology Layer (business)
Data from many platforms, without another copy	Connects to sources such as Snowflake, Databricks, BigQuery, Postgres and object storage, and queries them in place	Maps business concepts such as Customer, Policy and Claim to the data products that hold them, so a business question never depends on which platform answers it
Context the agent can query	Defines the tables, joins, dimensions and individual metrics owned by its domain, such as claim count or paid amount	Defines enterprise concepts and the relationships between them: one Customer every data product maps to, how a policy relates to a claim, and business metrics that span domains, such as customer lifetime value
Governance enforced at query time	Enforces technical access controls on each request: attribute-based policies, masking and row filters, for human and machine identities alike	Sets business policy: which agents may use which concepts and relationships, for what purpose, and which actions need human approval as well as what classifies as PII or Sensitive information.

Framework need	Data product (Technical)	DataOS Ontology Layer (business)
Trust you can verify	Runs technical quality checks such as schema, completeness and freshness SLAs, captures lineage to source, and has a named technical owner	Runs business rules and cross-platform consistency checks, such as a customer's status matching across billing and CRM, records which concepts an agent used, and has a named business owner
Relationships and actions across domains	Exposes the data, and the write-back interfaces, for its own domain	Models business entities, their relationships and the actions agents can take, such as approving a claim. Policy is enforced on every concept and relationship an agent queries and every action it takes
An interface agents already speak	Served over SQL, REST and GraphQL for analytics, applications and single-domain tasks	The default interface for agents, through APIs and MCP, so agents work in business terms. The same policies apply whichever layer a consumer uses

Why "across platforms" is the point



If your entire agent estate will live inside one platform, its native tools may be enough, and you should use them. We'd rather say that up front.

Most of the enterprises we talk to aren't in that position. They run a warehouse and a lakehouse, a handful of operational databases, and SaaS systems they don't control. Their agents need to reason across all of it. In that world, DataOS plays a specific role: one place to define meaning, one model of the business, one place to set policy, one audit trail, regardless of where the data lands.

That also means DataOS is complementary to the platforms, not a replacement. The data stays in Snowflake or Databricks. Their compute, their native governance and their agent features keep working. DataOS adds the layer that makes the agent's view consistent when a task spans more than one of them.

Built to be owned, not rented

Gartner [predicted in September](#) that 70% of enterprises will abandon agentic AI built through vendor forward-deployed engineering by 2028, largely because they can't evolve it on their own. That's a useful warning for any data foundation too. Data products in DataOS are declared as configuration that your teams version, review and change, so the context and policy behind your agents belong to you, not to a services engagement.

Where to start: one agent, one cross- platform use case

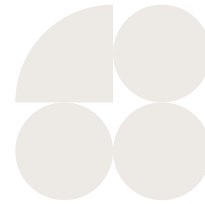


Trying to make all enterprise data agent-ready at once rarely works. The teams that get traction pick one agent with real business value whose task spans at least two systems, and build backwards from it.

- 1. Pick the agent, then the questions.** Write down the 10 to 20 questions or actions it must handle. These define the data products you need, not the other way round.
- 2. Map the data each question touches.** Note which platform each source lives on and who owns it. If no one owns it, fix that first.
- 3. Define the semantics and the business objects once.** Agree on the handful of metrics and entities the agent depends on, with the business owner in the room. Encode them in the semantic layer, and model the core entities, relationships and allowed actions in the ontology, not in the prompt.
- 4. Set agent-specific policy.** Give the agent its own identity. Decide what it can read, what it can't, and which actions it can take on which business objects. Apply masking and filters at the data product, not in application code.
- 5. Add contracts and quality gates.** Freshness SLAs, schema checks and quality rules that block bad data before the agent sees it.
- 6. Expose it the way agents connect.** MCP and APIs, with the governed path as the easiest path.
- 7. Run in shadow mode, then earn autonomy.** Log every query and decision. Compare agent outputs to human ones. Widen permissions only as the evidence supports it.

Once the first agent is running, the second one is cheaper. Most of the data products, semantics, policies and ontology objects are reusable. Each new agent extends the model of the business instead of starting over, which is where the real return shows up.

How this plays out by industry



Industry	Where the pressure comes from	What usually matters most
Insurance	Claims, underwriting and service agents touching regulated personal data across policy, claims and billing systems	Explainable decisions and a reconstructable audit trail per agent action
Banking and credit unions	Member-service and fraud agents spanning a core banking platform, digital channels and analytics	Agent identity, least-privilege access and model risk oversight
Distribution, CPG and retail	Sales, pricing and supply agents reconciling accounts, SKUs and territories across ERP, CRM and a warehouse	Consistent definitions, so agents in different teams don't contradict each other
Healthcare and life sciences	Clinical and operational agents working with protected data spread across many systems	Purpose-limited access and provable masking

The common thread: the more regulated the data and the more systems a task crosses, the less you can rely on any single platform's controls.

Conclusion



Most agent programs that run into trouble do so at the data layer. The data they're given lacks defined meaning, the access they're granted isn't scoped to the task, and their actions leave no record that can be checked. Each platform addresses some of this within its own boundaries, but agent tasks routinely cross those boundaries.

What holds up across platforms is a governed interface that carries context, governance and trust together, wherever the data lives, with an ontology on top so agents can act across domains under the same rules they read by. Start with one agent and a handful of data products. Each agent after that reuses what the first one built, which is where the cost and risk come down.

FAQs



Isn't this just a semantic layer?

Overlaps, but it's broader. A semantic layer handles context. Agent-ready data also needs policy enforced per agent identity, quality guarantees and an audit trail, delivered through the same interface.

We're standardized on one platform. Do we need this?

Maybe not yet. If every agent task stays inside one platform, start with its native controls. The question to ask is whether that will still be true when your third or fourth agent goes live.

Does this replace our catalog or our data platform?

No. Your platforms keep storing and processing the data, and your catalog keeps serving people. The agent-ready layer is what makes them consistent from an agent's point of view.

Do we have to adopt data mesh to do this?

No. You need owned, governed data products for the data your agents use. You don't need to reorganize the company around domains to get there. Start with one agent's worth of data products.

How is this different from an MCP gateway?

A gateway controls which tools an agent can call. It doesn't know what "active customer" means or whether the data behind the tool is fresh. You want both; the gateway is the front door, and the data product is what's behind it.

Is the DataOS Ontology Layer just a knowledge graph?

Overlaps, but it does more. A knowledge graph describes how things relate. The ontology layer also defines the actions agents can take on each business object, and it sits on governed data products, so every concept and relationship an agent queries and every action it takes is checked against policy and logged.

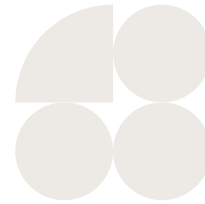
Do we need an ontology before we build data products?

No, the other way round. Start with the data products your first agent needs, then model the entities, relationships and actions it uses. The ontology grows agent by agent instead of becoming a modeling project of its own.

How long does a first agent-ready use case take?

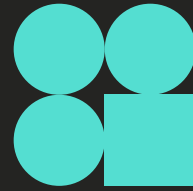
It depends on how many sources the agent touches and whether owners already exist. Scoping to one agent and a handful of data products is what keeps it to weeks rather than quarters.

Sources



- [Gartner, Top Strategic Technology Trends for 2027, Oct 20, 2026](#)
- [Gartner, AI-Ready Data Needs to Expand to Agent-Ready Data, Aug 27, 2026](#)
- [Gartner, Agent-Ready Data Will Reshape Data Management System Market Share by 2027, Jul 24, 2026](#)
- [Gartner, 70% of enterprises will abandon agentic AI built by vendor forward-deployed engineering, Sep 29, 2026](#)
- [Gartner, GenAI application security incidents and MCP risk, Apr 9, 2026](#)
- [Gartner, Zero-trust data governance by 2028, Jan 21, 2026](#)
- [Gartner, Over 40% of agentic AI projects will be canceled by end of 2027, Jun 25, 2025](#)
- [Gartner, Lack of AI-Ready Data Puts AI Projects at Risk, Feb 26, 2025](#)
- [IBM Institute for Business Value, Redefining the tech leader's mandate \(2026 Tech Leader Study\), Jun 2026](#)
- [Kiteworks summary of CSA / Token Security AI agent incident survey, Apr 2026](#)
- [Informatica, CDO Insights 2026, Jan 2026](#)
- [Precisely and Drexel LeBow, 2026 State of Data Integrity and AI Readiness, Jan 2026](#)
- [BARC / Actian, Data Products and Data Contracts in 2026, Apr 2026](#)
- [Thoughtworks, Making Your Data Ready for Agentic AI, Aug 2026](#)
- [Thoughtworks, The state of data mesh in 2026](#)
- [Apache Ossie \(formerly Open Semantic Interchange\)](#)
- [Atlan, Snowflake Summit 2026 announcements](#)
- [Atlan, Agent context layer tools directory](#)
- [Gartner, Top Strategic Technology Trends for 2027: AI Context Platforms, Oct 16, 2026](#)

Contact Us



About The Modern Data Company

The Modern Data Company works with enterprises to build a governed, contextual, AI-ready foundation for data-driven decisions.

Its DataOS platform connects enterprise data, business context and intelligence across complex operating environments, so organizations can build reusable data products, establish semantic context and support analytics, AI and intelligent applications.

With DataOS, enterprises get measurable value from their existing data stack, delivering up to **80% faster delivery cycles, 1.5X productivity gains for data engineering, and 50% lower data operational costs**. With a focused use case, organizations can validate this value and move from data foundation to measurable business outcomes in **as little as 4-6 weeks**.

To schedule a demo or learn more about DataOS:

info@tmdc.io

www.themoderndatacompany.com