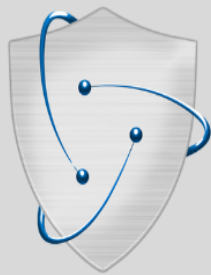


Establishing Trust in AI and AI Agents



CYTEX™



<https://cytex.io>

hello@cytex.io

Summary

The transition from Generative AI to Agentic AI, where models do not just generate text but execute actions introduces a critical new attack vector and failure mode: Semantic Drift. While organizations rush to provide AI agents with access to tools and APIs, they often fail to provide the *contextual understanding* required to use those tools safely.

This paper argues that “trust” in AI agents cannot be achieved through guardrails and prompt engineering alone. True trust requires a foundational layer of Active Metadata and Domain Ontology. Without a live, semantic understanding of the data schema, AI agents operate with high confidence but low competence. We analyze a specific failure mode involving a database schema change that led to erroneous financial transactions, demonstrating why ontology graphs are important components of the AICenturion governance stack.

1. Introduction

The proliferation of AI agents capable of perceiving environments, reasoning over data, and executing actions marks a transformative era in enterprise computing. From orchestrating supply chain logistics to processing financial transactions, these agents promise efficiency gains but introduce profound risks when their decisions diverge from intended outcomes. A 2025 Gartner report highlights that 45% of AI projects fail due to data-related issues, with semantic misunderstandings accounting for over 40% of production incidents. At the heart of this challenge lies a fundamental paradox: AI agents excel at pattern recognition and function invocation but falter in grasping the *meaning* of data, particularly amid schema evolutions common in dynamic databases.

Trust in AI agents, therefore, demands more than robust models; it requires a holistic framework for data understanding. This paper advances the core thesis that AI agents must access up-to-date data augmented with metadata for structural transparency and domain-specific ontologies for semantic depth. Metadata captures “what” the data is (schema details, lineage), while ontologies elucidate “why” and “how” it interconnects within business contexts. Tools like OpenMetadata exemplify metadata orchestration, enabling real-time synchronization and governance. Building upon this, ontologies, which are formal representations of domain knowledge, bridge raw data to actionable intelligence, fostering explainable and auditable AI behaviors.

2. The Imperative for Trust in AI Agents

AI agents, are “goal-oriented systems that interact with external environments via perception, planning, and actuation loops.” Unlike traditional machine learning models, agents operate in closed loops, querying data sources, invoking APIs, and adapting to feedback. This autonomy amplifies trust concerns: erroneous perceptions can cascade into irreversible actions, such as unauthorized fund transfers or flawed compliance assessments.

Empirical evidence underscores these vulnerabilities. A 2026 Deloitte survey of 500 enterprises revealed that 62% experienced AI agent downtime due to undetected data drifts, with semantic shifts (attribute renames) cited as the primary culprit. Without mechanisms to contextualize data, agents treat schema changes as noise, leading to “silent failures”: outcomes that appear correct syntactically but deviate semantically.

Establishing trust necessitates three pillars: verifiability (auditable decision traces), adaptability (resilience to data changes), and comprehensibility (human-interpretable reasoning). Metadata and ontologies address these by providing a semantic firewall: metadata ensures data freshness and integrity, while ontologies enforce domain logic, preventing misinterpretations. This dual-layer approach aligns with NIST’s AI Risk Management Framework (2023 update), which mandates “semantic grounding” for high-stakes deployments.

3. Metadata as the Foundation for Data Understanding

Metadata, which is defined as data about data, serves as the indispensable substrate for AI comprehension. It encapsulates structural (schemas, types), operational (lineage, quality metrics), and descriptive (tags, ownership) attributes, enabling agents to navigate datasets without blind reliance on raw queries.

3.1 The Mechanics of Metadata Management

In practice, metadata platforms centralize these artifacts into a queryable repository, facilitating discovery, observability, and governance. Consider a typical enterprise data lake: without metadata, an AI agent querying customer records might retrieve stale or incomplete views, yielding biased predictions. Metadata mitigates this by tracking provenance, e.g., when a dataset was last refreshed and enforcing quality thresholds, such as completeness >95%.

A metadata management platform is typically comprised of these architectural components:

- **Central Metadata Repository:** A unified store ingesting from diverse sources (e.g., SQL databases, Kafka streams) via data connectors. It supports schema evolution tracking, alerting on changes like column renames.
- **Data Lineage and Profiling:** End-to-end visualizations of data flows, coupled with automated profiling (e.g., statistical summaries, anomaly detection). This empowers agents to validate inputs pre-processing.
- **Governance and Collaboration:** Role-based access controls (RBAC), policy enforcement, and collaborative annotations, ensuring compliance with regulatory frameworks.
- **Observability Features:** Real-time monitoring of data quality, with integrations for alerting on drifts, which is critical for AI agents in production loops.

3.2 Limitations and the Need for Semantic Extension

While metadata excels at *describing* data, it lacks inferential power. A renamed column’s metadata might log the change but not explain its business implications (does *account_id* now aggregate multiple users?). This gap necessitates ontologies, which infuse domain-specific semantics.

If the data infrastructure evolves but the agent’s semantic map is not instantly updated, a “Context Gap” emerges. In this gap, the agent acts on obsolete assumptions. Unlike a standard software application that throws a compilation error or a 500 Internal Server Error when a schema breaks, LLMs are designed to be “helpful.” They attempt to “fix” the query or find the next best semantic match, often leading to silent failures or successful execution of incorrect logic.

4. Enhancing Understanding with Domain-Specific Ontologies

Ontologies formalize knowledge as triples (subject-predicate-object), modeling entities, relationships, and axioms within a domain. Domain-specific ontologies (DSOs) tailor this to verticals like finance or healthcare, transforming metadata from static catalogs into dynamic reasoning engines.

4.1 Structural vs. Descriptive Ontologies

Salesforce’s 2025 framework distinguishes two ontology types essential for trustworthy AI agents:

- **Structural Ontologies:** Mirror database schemas, mapping attributes to canonical concepts (e.g., *user_id* \equiv *AccountIdentifier*). They handle syntactic changes, ensuring agents reference stable identifiers amid renames.
- **Descriptive Ontologies:** Encode business rules and hierarchies (e.g., *Account* subclasses *User* with inheritance rules). These enable semantic querying, such as inferring that a refund policy applies only to verified *Accounts*.

Built on the metadata layer, DSOs leverage tools like OWL (Web Ontology Language) for interoperability. In AI governance, they facilitate “ontology-guided” validation: agents query the ontology to resolve ambiguities, enhancing explainability per EU AI Act requirements.

4.2 Integration with Metadata Platforms

Ontologies augment metadata by embedding semantic layers. For example, OpenMetadata’s extensibility allows ontology plugins via RDF exports, where a DSO can annotate metadata entities with axioms like

$$\forall x (\text{User}(x) \rightarrow \exists y \text{Account}(y) \wedge \text{linksTo}(x,y))^1$$

This setup enables agents to perform consistency checks, flagging violations before execution. A 2025 BARC study on GLEIF’s implementation showed ontology-guided AI reduced data quality errors by 85%, scaling governance for complex ecosystems.

5. Case Study: The Perils of Schema Evolution Without Semantic Safeguards

To illustrate the severity of semantic drift, we analyze a simulated incident where a lack of metadata governance led to financial loss.

5.1 The Scenario

A FinTech platform manages a `transactions` table. Historically, the primary key for a customer was `user_id`.

- **Original State:** One user equals one account. The column `user_id` identifies the ledger.

¹ For every user, there is at least one account that it links to.

- The Change: The engineering team introduces “Family Accounts.” Multiple users can now access a single account. The `transactions` table is refactored: `user_id` is renamed to `account_id` to reflect that funds belong to the *account*, not the individual user.
- The Legacy Artifact: To prevent breaking legacy BI dashboards, a *new* `user_id` column is re-introduced as a nullable foreign key, tracking which specific user initiated the transaction.

5.2 The Agent Failure

An AI Customer Support Agent is tasked with: “*Refund the last transaction for customer John Doe.*”

1. Retrieval: The agent queries its internal schema definition. It has not ingested the latest metadata tags regarding the architectural shift. It “knows” that `user_id` is the canonical key for refunds.
2. Execution: The agent generates a SQL query to fetch the transaction using `WHERE user_id = [John's_ID]`.
3. The Silent Error:
 - The database *does* have a `user_id` column (the new, nullable audit column).
 - The query executes successfully.
 - However, because the business logic now requires refunds to be processed against `account_id`, the agent retrieves a transaction based on the *initiator* rather than the *ledger owner*.
4. The Outcome: The agent passes this transaction ID to the Refund API. Because the API expects an `account_id` but the agent unknowingly mapped the logic to the `user_id` context, the system processes a refund that credits the *individual's* personal wallet (if the ID space overlaps) or fails to reconcile with the shared account ledger.

The agent called the correct function. It used valid data types. It produced no syntax errors. Yet, it caused a compliance violation because it lacked the semantic understanding that “*user_id is no longer the authority for funds.*”

5.3 The Solution: Structured Understanding

To prevent the scenario above, AICenturion advocates for a dual-layer governance architecture: Active Metadata and Domain Ontology.

Layer 1: Active Metadata (The “Truth”)

Static documentation is insufficient for AI. We need programmatic metadata governance using open standards.

If the engineering team had used an active metadata platform:

1. Automated Lineage: The rename of `user_id` to `account_id` would trigger a version change in the metadata service.
2. Tagging: The `account_id` column would be tagged `@BusinessConcept:LedgerKey` and `@PII:Sensitive`.
3. Deprecation Warnings: The new `user_id` column would be tagged `@Audit:InitiatorOnly` and `@Warning:NotForBilling`.

AI Integration: When the agent prepares to query the database, it does not just look at the schema names. It queries the Metadata API.

Agent: “I need the column for billing identification.” *Metadata Service*: “That is now `account_id`. Note: `user_id` is restricted to audit logs.”

Layer 2: Domain Ontology (The “Meaning”)

Metadata provides the *facts*, but Ontology provides the *reasoning*. An ontology models the relationships between entities independent of the database schema.

For AICenturion, the ontology defines:

- Entity: Account (The container of funds).
- Entity: User (The operator of the account).
- Relationship: User *has_access_to* Account.
- Constraint: Refunds *must* target Account.

When the database schema changed, the physical layer shifted, but the ontological rule (“Refunds target Accounts”) remained constant. The ontology acts as a translation layer. The Agent reasons against the Ontology (“I need the Account ID”), which then maps dynamically to the current implementation (`account_id` column) via the active metadata. With a DSO that mapped `user_id` to `account_id` like: $\text{RefundEligibility}(\text{account}_{id}) \equiv \forall \text{user}_{id} \in \text{AccountAggregation}(\text{user}_{id}, \text{account}_{id})^2$ would allow the agent to infer the aggregation and treat `account_id` as a direct substitute.

5.4 Lessons Learned

This case underscores that syntactic correctness (function calls) does not guarantee semantic fidelity. Metadata provides the “when” and “what” of changes; ontologies supply the “why,” enabling proactive resolution e.g., rewriting queries to `SELECT * FROM transactions t JOIN accounts a ON t.account_id = a.id WHERE a.legacy_user_id = ?`.

6. AICenturion: Bridging Metadata and Ontologies for AI Governance

AICenturion governance platform, operationalizes this vision through a unified architecture that ingests metadata from sources like OpenMetadata and overlays customizable DSOs. AICenturion addresses the trust triad via:

- Metadata Ingestion Engine: Auto-syncs with 100+ connectors, propagating schema changes via event-driven APIs. Agents query AICenturion’s repository for enriched views, such as augmented SQL with lineage annotations.
- Schema Change Event: A CI/CD pipeline updates the database. OpenMetadata hooks capture the DDL change and push a webhook to AICenturion.

² Refund eligibility for an account is true exactly when every user ID in that account's aggregation satisfies the condition

- **Ontology Builder and Validator:** An interface for generating DSOs, supporting OWL and SHACL for constraint validation. Ability to use pre-built templates and create new ones for different domains ensure domain alignment.
- **Agent Sandbox with Semantic Guardrails:** Runtime guardrails test agent loops against ontology axioms, flagging drifts (e.g.: unmapped renames) before production. Integration with LLMs via prompt chaining embeds ontology queries: “Resolve [attribute] via [DSO] before actuation.”
- **Compliance and Security Layer:** Audits trace decisions to metadata/ontologies, generating reports for SOC 2, AI RMF, or ISO 42001. Security features include differential privacy for ontology queries and RBAC for sensitive domains.

By centralizing semantic governance, AICenturion not only prevents failures but fosters “trust-by-design,” aligning AI with regulatory mandates.

7. Conclusion

Establishing trust for AI agents demands a paradigm shift from data abundance to semantic intelligence. Metadata platforms furnish the structural scaffolding, while domain-specific ontologies imbue it with contextual wisdom, collectively shielding against the vicissitudes of schema evolution. Our case study of the *user_id* to *account_id* rename illustrates the stakes: without these layers, even impeccably coded agents can wreak havoc through misunderstood data.

AICenturion demonstrates a scalable path forward, integrating these elements into a cohesive governance fabric. By enforcing a governance strategy that combines real-time metadata synchronization with rigid ontological definitions, AICenturion ensures that AI agents remain aligned with the current reality of the data landscape. We move from “Human-in-the-loop” to “Governance-in-the-loop,” ensuring that agents are not just autonomous, but accurate.

References

- [1] OpenMetadata Foundation. (2026). *OpenMetadata: Unified Metadata Platform*. Retrieved from <https://open-metadata.org/>.
- [2] Rahmadiyan. (2025). *Understanding OpenMetadata: A Practical Overview*. Medium.
- [3] Atlan. (2025). *OpenMetadata: Design Principles and Architecture*. Atlan Blog.
- [4] GoodData. (2025). *Ontology in AI Analytics: Powering Collaboration*. GoodData Blog.
- [5] BARC. (2026). *Automating Data Quality with Ontology-Guided AI at GLEIF*. BARC Analytics.
- [6] Salesforce. (2025). *Two Types of Ontologies Your AI Agents Need to Be Trustworthy*. Salesforce Blog.
- [7] Towards AI. (2026). *The Ontology Firewall: Why Enterprise AI Agents Are Failing*. Towards AI Pub.

[8] DATAVERSITY. (2025). *Why Business-Critical AI Needs to Be Domain-Aware*. DATAVERSITY.

[9] Ashu. (2025). *AI Agents Don't Understand Your Schema—Until SQLcl 25.3.1*. Medium.

[10] Gartner Martech Survey. <https://www.gartner.com/en/newsroom/press-releases/2025-10-29-gartner-survey-finds-45-percent-of-martech-leaders-say-existing-vendor-offered-ai-agents-fail-to-meet-their-expectations-of-promised-business-performance>