Multi-Vendor AI Agent Interoperability

by Nick Clark | Published March 27, 2026 | PDF

The AI agent ecosystem is fragmenting across vendor-specific frameworks that define agents differently. An agent built in one framework cannot delegate to, coordinate with, or even understand an agent built in another. A canonical six-field agent schema provides the structural standard that enables agents from any vendor to interact through shared field semantics: governance, memory, lineage, execution eligibility, identity, and policy are intrinsic typed fields rather than vendor-specific implementation details.

The fragmentation problem in AI agent frameworks

Every major AI platform has released its own agent framework with its own agent definition. Each framework defines what an agent is differently: what state it carries, how it communicates, what governance applies, and how it relates to other agents. The result is an ecosystem where agents are locked to their framework as thoroughly as applications were locked to operating systems in the 1990s.

An enterprise deploying agents from multiple vendors for different functions, a customer service agent from one vendor, an analytics agent from another, a compliance agent from a third, faces an integration problem that grows with each vendor. Each agent speaks its own protocol, carries its own state format, and implements its own governance model. Making them work together requires custom integration for every pair of frameworks.

The absence of a structural standard for what an agent is means that agent interoperability is treated as an API integration problem rather than a schema compatibility problem. APIs can change. Schemas define the structural contract.

Why API standards alone cannot solve interoperability

API standards define how agents communicate. They do not define what agents are. An agent communication protocol can transmit a message from agent A to agent B. But if agent A carries governance as a policy field and agent B implements governance as an external service, the protocol cannot reconcile the structural difference. The message arrives, but the governance semantics do not translate.

Similarly, agent discovery protocols can advertise what an agent can do, but they cannot verify that an agent is structurally capable of governed interaction. An agent that advertises a capability but has no intrinsic governance field cannot be trusted to behave within governance constraints. The discovery tells you what the agent claims. The schema tells you what the agent structurally is.

How the canonical agent schema addresses this

The canonical agent schema defines six typed fields that every agent must carry: governance, memory, lineage, execution eligibility, identity, and policy. These fields are not optional metadata. They are structural requirements. An object that lacks any of these fields is not a valid agent, regardless of what it calls itself.

Interoperability becomes structural rather than protocol-based. When an agent from vendor A interacts with an agent from vendor B, each agent can inspect the other's canonical fields to determine governance compatibility, verify identity through lineage, and evaluate execution eligibility. The vendor-specific implementation behind each field is irrelevant. The schema defines the interface contract.

Delegation between agents follows the same schema contract. A delegating agent can verify that the delegate has governance fields compatible with the task's requirements, memory sufficient to carry the task state, and lineage that demonstrates the delegate's operational history. These evaluations are structural, not negotiated through ad hoc API calls.

What implementation looks like

An enterprise deploying multi-vendor agents adopts the canonical schema as the structural contract for agent interaction. Each vendor's agents expose the six canonical fields through a standard interface. Vendor-specific capabilities are implemented behind those fields but interact with other agents exclusively through the schema contract.

For enterprise architects, the canonical schema reduces agent integration from a per-pair problem to a per-vendor compliance problem. Each vendor certifies that its agents implement the canonical fields. Cross-vendor interaction is then structural rather than requiring custom integration.

For compliance teams, the canonical schema provides a uniform governance inspection surface across all deployed agents. Regardless of which vendor built the agent, the governance field carries the policy that governs the agent's behavior. Audit and compliance verification operate against the same schema for every agent.

For agent developers, the canonical schema provides clear structural requirements for building interoperable agents. Instead of implementing compatibility with each potential interaction partner, the developer implements the six canonical fields and gains interoperability with every other agent that implements the same schema.

Agent Schema All 21 steps →

Define what an autonomous agent is — structurally.

Patent
US 19/452,651 · filed
Primary Technical Disclosure
○ Cognition-Compatible Semantic Agent Objects and Structural Validation
Secondary Technical
○ Partial Agent Structural Validity: Fewer Fields, Still Deterministic○ Minimum Two-Field Validation Threshold: The Floor of Semantic Structure○ Field Interaction Rules: Deterministic Constraints Between Canonical Fields○ Field-Based Role Typing: Agent Roles Derived From Structural Composition○ Semantic Templates: Predefined Field Arrangements as Agent Class Contracts○ Structural Scaffolding Logic: Resolving Missing Fields Through Inference or Defaulting○ Field-Aware Default Resolution: Deterministic Behavior When Fields Are Absent○ Traceable Semantic Lineage Graph: Mutation History Embedded in Agent Objects○ Serialization With Stateless Compatibility: Reconstruction Without External Session State○ Schema Governance Through Versioned Policies: Cross-Version Structural Interoperability
Applications (General)
○ Enterprise AI Agent Interoperability Through Canonical Schema○ Robotic System Standardization via Structural Field Composition● Multi-Vendor AI Agent Interoperability○ Digital Twin Standardization Through Canonical Fields○ Healthcare AI Agent Portability○ Defense Coalition Interoperability○ Insurance Claims Processing Through Standard Agents○ Legacy System Integration via Schema Bridging
Applications (Specific)
○ LangChain Built the Agent Framework. It Did Not Define What an Agent Is.○ AutoGen Enabled Multi-Agent Conversations. The Agents Have No Structural Definition.○ CrewAI Organized Agents Into Teams. The Agents Still Have No Schema.○ Semantic Kernel Integrated AI Into Enterprise Code. The Agents It Creates Have No Schema.○ OpenAI Assistants API Provides Agent Tooling. It Does Not Define Agent Structure.○ Google Vertex AI Agents Provide Managed Agent Infrastructure. The Agents Have No Canonical Schema.○ Amazon Bedrock Agents Orchestrate Foundation Models. The Agents

Have No Structural Definition.○ Haystack Built Composable NLP Pipelines. The Pipeline Components Have No Agent Schema.○ LlamaIndex Built the Data Framework for LLM Applications. The Data Objects Have No Agent Schema.○ Dify Made LLM Application Development Visual. The Applications Have No Agent Schema.
Agent Schema overview →
AQ
deterministic
autonomy

Legal

Last updated: 2026-03-03

•

-
- nick@qu3ry.net
- 72 28 14 36 01

Invented by Nick Clark | Founding Investors: Devin Wilkie