Enterprise AI Agent Interoperability Through Canonical Schema

by Nick Clark | Published March 27, 2026 | PDF

Every enterprise AI deployment runs agents from multiple frameworks: LangChain agents, AutoGen multi-agent systems, custom-built inference pipelines, vendor-specific chatbots. Each framework defines agents differently, stores memory differently, and handles governance differently. The canonical agent schema provides structural interoperability by defining what an agent is through six typed fields, enabling agents from any framework to participate in shared governance environments without protocol translation layers.

The interoperability crisis in enterprise AI

An enterprise running LangChain agents for customer service, AutoGen agents for code generation, and a custom RAG pipeline for internal knowledge faces a fundamental integration problem. These systems define agents differently. They store agent state in incompatible formats. They have no shared

model for what it means for an agent to be authorized, capable, or trustworthy.

The current approach is middleware: API adapters, message translation layers, and orchestration platforms that bridge between frameworks. This creates a combinatorial integration burden. Every new framework requires adapters to every existing framework. Every agent interaction that crosses framework boundaries requires translation, and translation introduces semantic loss. An agent's governance constraints in one framework may have no representation in another.

The deeper problem is that these frameworks do not share a structural definition of what an agent is. Without agreement on the essential properties of an agent, interoperability can only be achieved at the communication layer. Agents can exchange messages, but they cannot share governance, verify each other's integrity, or participate in common trust relationships.

Why communication protocols do not solve structural interoperability

Standards like MCP (Model Context Protocol) and A2A (Agent-to-Agent) define how agents communicate. They specify message formats, transport mechanisms, and interaction patterns. But they do not define what the agents exchanging those messages actually are in structural terms.

Two agents can communicate via the same protocol while having completely incompatible governance models, incompatible memory structures, and no ability to verify each other's execution eligibility. Communication interoperability is necessary but not sufficient. Structural interoperability requires agreement on the essential fields that constitute an agent.

How canonical agent schema addresses this

The canonical agent schema defines six typed fields that every agent must carry: governance policy, memory state, lineage history, execution eligibility, trust scope, and capability declarations. These fields are not communication protocols. They are structural properties of the agent object itself.

Any agent from any framework can be wrapped in or mapped to the canonical schema. A LangChain agent's tool permissions become capability declarations. An AutoGen agent's conversation history becomes memory state. A custom pipeline's access control rules become governance policy. The mapping does not require changing the underlying framework. It requires expressing the agent's essential properties in canonical typed fields.

Once agents share the canonical schema, structural interoperability follows. A LangChain agent can verify an AutoGen agent's governance credentials before accepting a delegation. A custom pipeline agent can evaluate a vendor chatbot's trust slope before sharing sensitive context. Interoperability happens at the identity and governance level, not just the message level.

Partial validity rules allow agents with incomplete schemas to participate at reduced capability. An agent that carries governance and memory fields but lacks lineage history can operate within trust zones that do not require lineage verification. The schema accommodates real-world heterogeneity while maintaining structural guarantees.

What implementation looks like

An enterprise adopting canonical agent schema deploys a schema validation layer at the boundary of each agent framework. Agents from different frameworks register their canonical fields when entering a shared governance environment. The validation layer verifies that the agent's fields meet the minimum requirements for the intended trust zone.

For a financial services firm running compliance agents from one vendor and trading agents from another, the canonical schema enables the compliance agent to structurally verify the trading agent's governance constraints before authorizing a trade. The verification happens through typed field comparison, not through vendor-specific API calls.

For platform companies offering agent marketplaces, the canonical schema provides a structural certification standard. An agent listed in the marketplace carries verifiable governance, capability, and trust fields. Buyers can evaluate an agent's structural properties before deployment, not just its API specification.

Agent Schema All 21 steps →

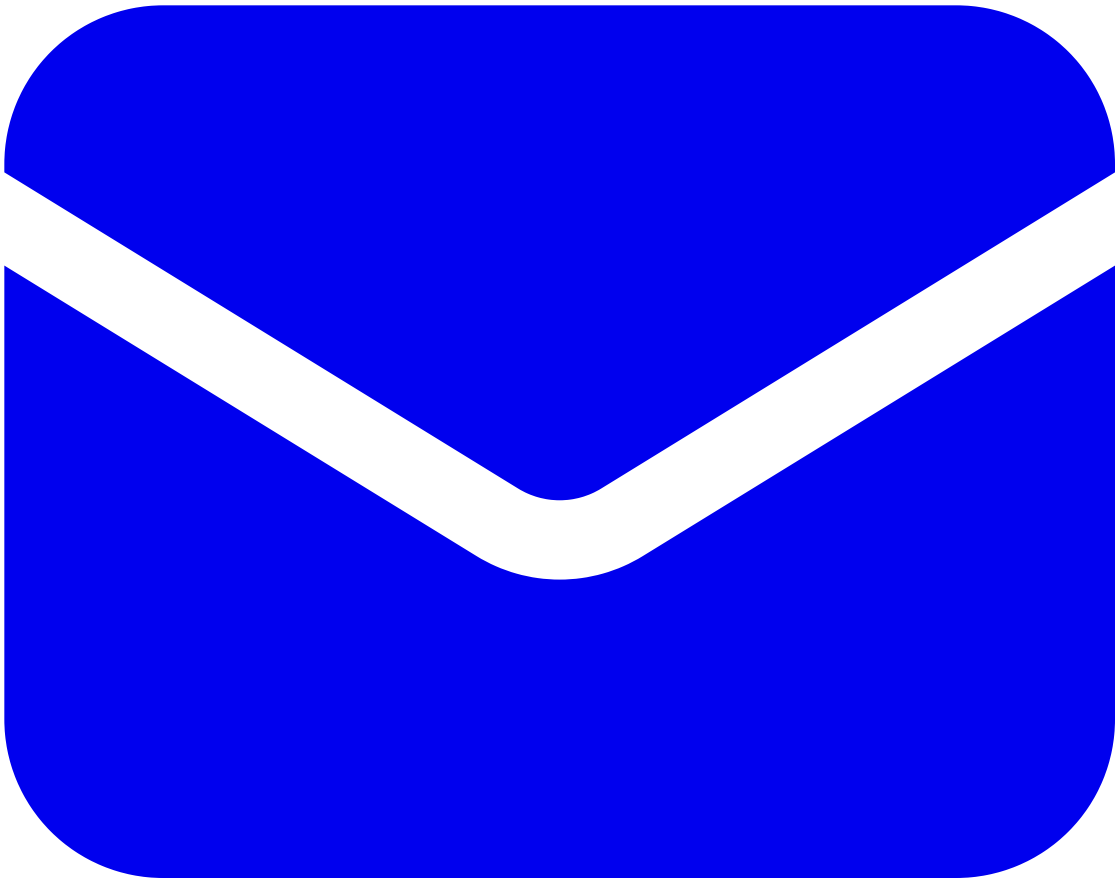
Define what an autonomous agent is — structurally.

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