Semantic Kernel Integrated AI Into Enterprise Code. The Agents It Creates Have No Schema.

by Nick Clark | Published March 27, 2026 | PDF

Microsoft's Semantic Kernel made LLM integration natural for enterprise developers by providing plugins, planners, and memory connectors that work with C# and Python codebases. The SDK treats AI capabilities as functions that compose with existing enterprise code. But Semantic Kernel agents are plugin compositions, not schema-defined objects. There is no canonical definition of agent identity, governance, or memory structure. The gap is between SDK integration and structural agent definition.

Semantic Kernel's approach to enterprise AI integration is well-considered. Treating LLM capabilities as native functions, providing strongly-typed plugin interfaces, and supporting multiple AI services through a unified API are genuine contributions. The gap described here is not about the SDK. It is about the agents the SDK produces.

Plugins are capabilities, not agent fields

Semantic Kernel agents are assembled from plugins (functions the agent can call), memory (conversation and semantic memory connectors), and planners (strategies for composing function calls). The kernel orchestrates these components to respond to user requests.

But plugins are capability descriptors, not agent schema fields. A Semantic Kernel agent has no typed governance field. It has no policy reference that constrains its behavior. It has no integrity metric that tracks behavioral consistency. The agent's structure is whatever the developer composes. There is no canonical schema that says: this is what an agent must be.

Memory connectors are not semantic memory

Semantic Kernel provides memory connectors for conversation history, semantic search, and vector stores. These are storage abstractions, not semantic memory with governance.

In a canonical agent schema, memory is a typed field with defined structure, lineage tracking, and governance constraints. Every memory entry has provenance. Memory mutations are validated. Memory persists across execution cycles with full lineage. Semantic Kernel's memory connectors provide storage. They do not provide governed semantic memory.

What a canonical agent schema provides

A canonical agent schema defines typed fields that constitute an agent: identity, memory, governance, capabilities, execution state, and lineage. Semantic Kernel's plugins would map to the capabilities field. Its memory connectors would implement the memory field. But the governance, identity, and lineage fields would be new: structural components that make the agent validatable and governable.

With a canonical schema, Semantic Kernel agents deployed in enterprise environments would be structurally interoperable. An agent built by one team could be validated by another team's governance framework because both share the same schema. Compliance becomes structural rather than procedural.

The remaining gap

Semantic Kernel made AI integration natural for enterprise developers. The remaining gap is in agent definition: a canonical schema that makes agents structurally defined, governable, and interoperable objects rather than ad hoc compositions of plugins and memory connectors.

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

Last updated: 2026-03-03

- Inventive Steps
- Licensing
- Patents
- Articles
- Legal
- Opportunities
- Sitemap

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

Invented by Nick Clark | Founding Investors: Devin Wilkie