



[Home](#) [Licensing](#) [Patents](#) [Articles](#)

Schema Governance Through Versioned Policies: Cross-Version Structural Interoperability

by [Nick Clark](#) | Published March 27, 2026 | [PDF](#)

Decentralized enforcement of semantic integrity through structural validation using versioned policy references, supporting cross-version interoperability. Within the semantic agent architecture, this capability operates as a structural primitive at the schema level. It is not an optional enhancement or a configurable plugin but a mandatory architectural property that every participant encounters. The result is a system where schema governance through versioned policies is enforced by construction rather than by convention, policy, or external oversight.

What It Is

Decentralized enforcement of semantic integrity through structural validation using versioned policy references, supporting cross-version interoperability. This is a structural mechanism within the semantic agent architecture that operates at the schema level. It is not advisory, not configurable at the discretion of individual participants, and not dependent on external enforcement infrastructure.

Every interaction within the system encounters this mechanism as a mandatory constraint. The behavior it produces is deterministic: given the same inputs and the same system state, the outcome is identical regardless of which node evaluates it, when the evaluation occurs, or what substrate hosts the computation.

Why It Matters

Conventional agent frameworks address this problem through external type systems, runtime configuration, or convention-based interfaces. These approaches function adequately under controlled conditions but introduce structural fragility when agents from different frameworks attempt to interoperate or structural assumptions change between versions. The underlying assumption that all agents share the same runtime conventions and external type definitions becomes a liability precisely when reliability matters most.

Schema governance through versioned policies removes this fragility by embedding the relevant capability directly into the schema layer. There is no external dependency that can fail independently, no middleware that can be misconfigured, and no trust assumption that can be violated by a single compromised participant. The guarantee is structural.

How It Works

The mechanism operates through deterministic evaluation embedded in the semantic agent architecture. When a relevant operation is initiated, the system evaluates the applicable structural constraints against the current state. This evaluation consults the fields, policies, and lineage records that travel with the objects themselves rather than relying on external state that may be stale, unavailable, or compromised.

The outcome of each evaluation is recorded in an append-only lineage structure. This record is cryptographically committed, ensuring that the complete history of decisions, transitions, and state changes remains auditable and tamper-evident. No evaluation outcome can be retroactively altered without breaking the cryptographic chain.

Because the evaluation logic and the data it operates on travel together, the mechanism functions identically across network partitions, substrate migrations, and administrative boundaries. There is no central evaluation point that must be available for the system to operate correctly.

What It Enables

With schema governance through versioned policies as an architectural primitive, systems built on this foundation can operate autonomously while maintaining the structural guarantees that centralized architectures achieve through oversight. The capability is not a tradeoff between autonomy and governance but a resolution of the apparent conflict between them.

This enables deployment across centralized cloud infrastructure, federated multi-party environments, fully decentralized networks, and edge installations with intermittent connectivity. The structural guarantees hold regardless of deployment topology because they are properties of the objects and protocols themselves, not properties of the infrastructure that hosts them.

[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

Subject to one or more pending U.S. and international patent applications, see [Patents](#) for the current list and status. No license, express or implied, is granted. Any use requires a separate written agreement—see [Licensing](#). Patent applications referenced on this site are pending. Claim scope, if any, is subject to examination and may issue in altered form or not at all. See [Legal](#) for terms and conditions.

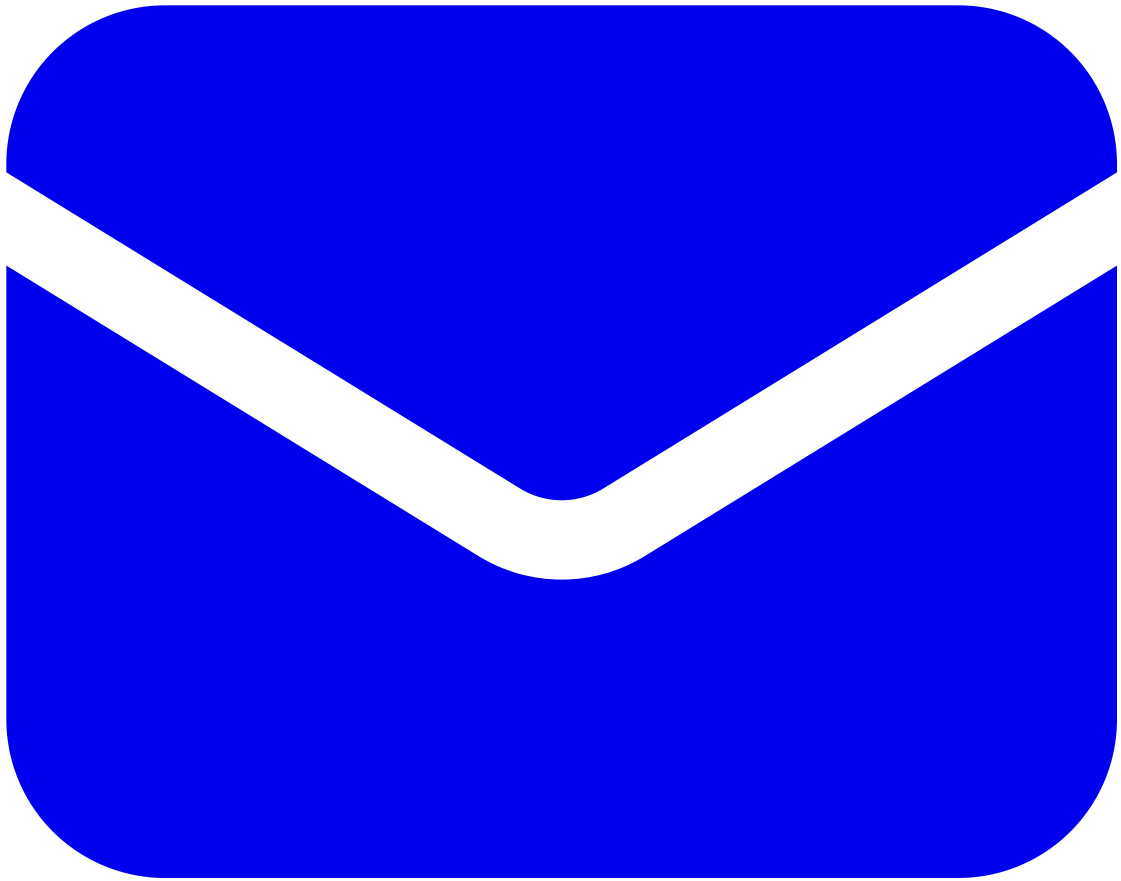
Adaptive Query™ is a trademark of Nicholas Clark. U.S. federal registration is pending. federal registration. AQ™, AQ Inside™, Adaptive Index™, Adaptive Network™, Semantic Agent™, @AQ™, AQID™, and Adaptive Coin™ are used as trademarks in connection with the Adaptive Query platform and brand. Other names may be trademarks of their respective owners.

Platform operated by Adaptive Query LLC, which provides patent and trademark licensing services. Copyright © 2025-2026 Nicholas Clark. All rights reserved.

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