Insurance Claims Processing Through Standard Agents

by Nick Clark | Published March 27, 2026 | PDF

An insurance claim touches a dozen organizations: the insurer, the reinsurer, the adjuster, the repair shop, the medical provider, the claimant, and often a legal representative. Each operates its own systems with no structural mechanism for their agents to coordinate. A canonical agent schema enables claims agents from all parties to carry governance, compliance lineage, and decision authority as intrinsic fields, enabling automated claims processing that crosses organizational boundaries with structural governance at every step.

The coordination problem in claims processing

Insurance claims are inherently multi-party processes. A simple auto collision claim involves the claimant filing a report, an adjuster evaluating the damage, a repair shop providing an estimate, and an insurer approving payment. Each party operates independently, communicating through phone calls,

emails, and web portals. The claim state is fragmented across these systems with no single source of truth and no structural mechanism for automated coordination.

Processing delays cost the insurance industry billions annually. Most delays are not decision delays. They are coordination delays: waiting for information from one party before another party can act. These delays exist because each party's systems operate independently with no structural mechanism for agents to coordinate across organizational boundaries.

Fraud detection is similarly hampered by organizational fragmentation. Detecting coordinated fraud requires correlating information across parties, but each party's fraud detection operates on its own data silo. Cross-party fraud patterns are invisible to any single participant.

Why API integration between insurers and vendors is insufficient

API integrations between insurers and their vendor networks solve point-to-point data exchange but do not solve multi-party coordination. Each integration is bilateral and custom. Adding a new party to the claims process requires new integrations. Changing the claims workflow requires updating every integration. The integration layer becomes as complex and brittle as the manual processes it was supposed to replace.

More fundamentally, API integrations exchange data without exchanging governance. An adjuster's estimate transmitted through an API arrives as data. Whether the estimate was produced according to the insurer's guidelines, whether the adjuster was authorized to make certain determinations, and whether the estimate's lineage is verifiable are questions the API cannot answer structurally.

How the canonical agent schema addresses this

A canonical agent schema enables each party in the claims process to deploy agents that carry governance, compliance, and decision authority as structural fields. The claim itself becomes an agent: an object that carries the claim data in its memory, the applicable policy terms in its governance field, the complete processing history in its lineage, and the current decision authority in its execution eligibility field.

When the claim agent moves from the adjuster to the repair shop, the repair shop's system inspects the claim agent's governance field to determine what repairs are authorized, what cost limits apply, and what documentation requirements exist. The inspection is structural. No phone call to the insurer is needed to clarify authorization terms.

When the claim agent returns to the insurer for payment approval, the lineage field carries the complete processing history: what the adjuster determined, what the repair shop estimated, what the claimant agreed to. Each entry in the lineage is structurally verified through the canonical fields of the agent that produced it. The insurer approves payment based on structural evidence rather than accumulated documents of uncertain provenance.

What implementation looks like

An insurer deploying canonical schema claims agents creates a claim agent at the moment a claim is filed. The agent carries the policy terms as governance, the claim details as memory, and the processing authority as execution eligibility. The agent then coordinates with adjuster agents, repair agents, and provider agents through the canonical schema.

For adjusters, the canonical schema provides structural authority for their determinations. The adjuster's agent carries its credentials and scope of authority in its governance field. The claim agent can verify that the adjuster is authorized to make the determination being proposed.

For fraud detection, the canonical schema enables structural cross-party correlation. Claim agents from different cases can be evaluated together because their lineage fields carry standardized processing histories. Patterns that are invisible in separate data silos become visible when lineage records follow a common schema.

For regulators, the canonical schema provides a uniform audit surface across all claims. Every processing step is recorded in the claim agent's lineage with structural provenance. Regulatory audits inspect lineage fields rather than reconstructing processing history from fragmented logs across multiple organizations.

Agent Schema All 21 steps →

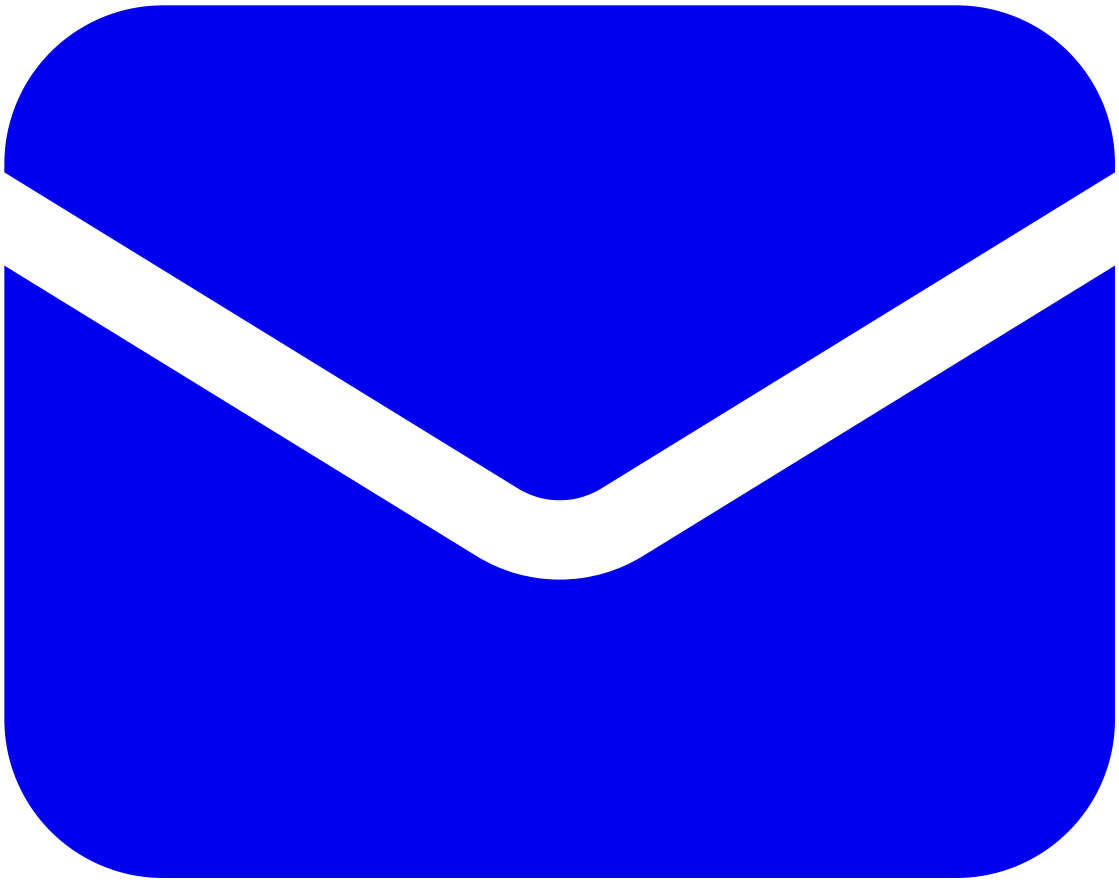
Define what an autonomous agent is — structurally.

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