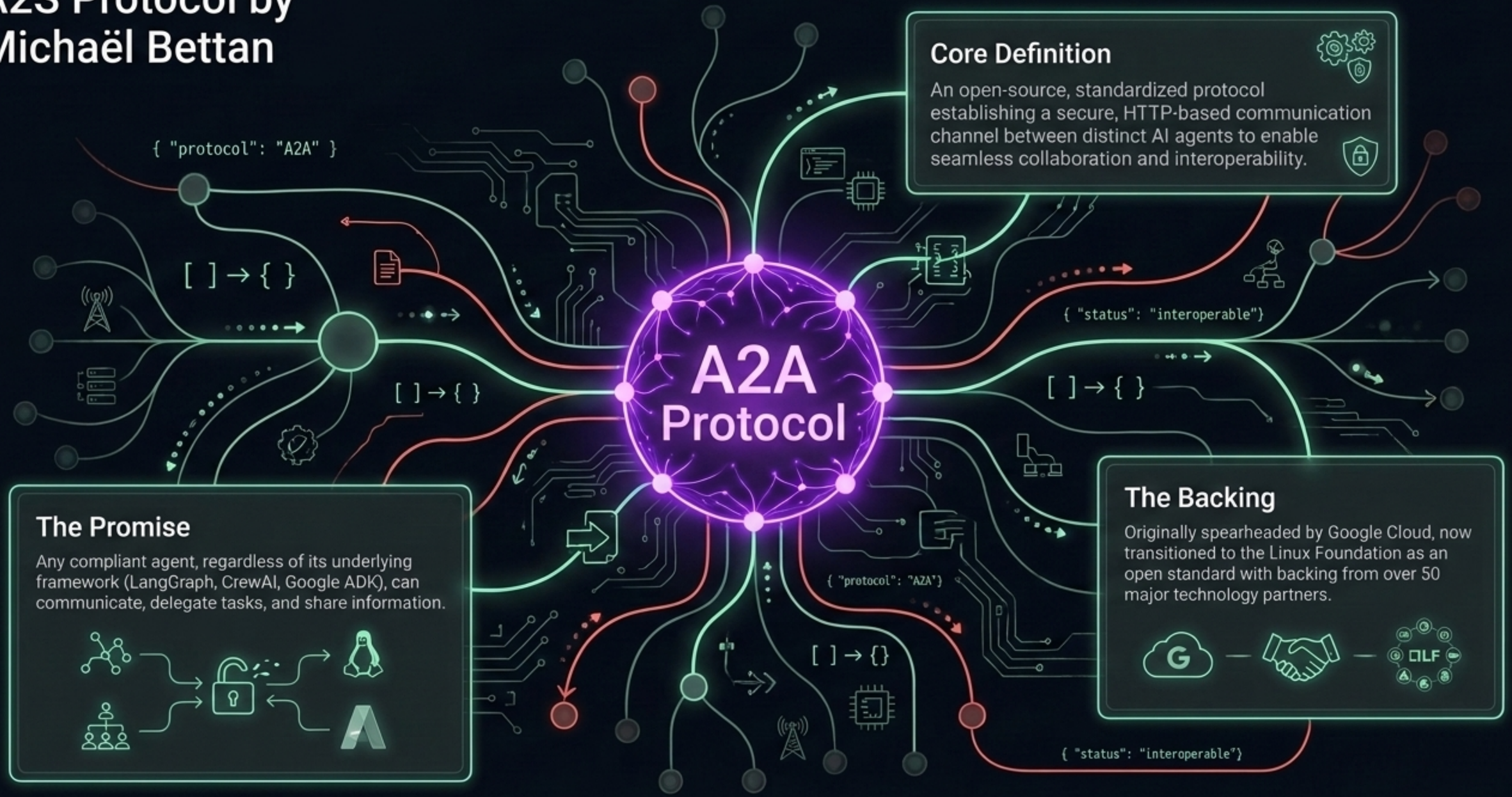
# A2S Protocol by Michaël Bettan




## Core Definition

An open-source, standardized protocol establishing a secure, HTTP-based communication channel between distinct AI agents to enable seamless collaboration and interoperability.

## The Promise

Any compliant agent, regardless of its underlying framework (LangGraph, CrewAI, Google ADK), can communicate, delegate tasks, and share information.

## The Backing

Originally spearheaded by Google Cloud, now transitioned to the Linux Foundation as an open standard with backing from over 50 major technology partners.

# Overcoming the Multi-Agent Silo Problem


## The Current State

Individual AI agents are built on different frameworks and operate in total isolation.

## The Impact

The lack of a common language hinders cohesive problem-solving. Agents cannot discover each other, share capabilities, or negotiate state.

## The A2A Fix

A standardized framework that tells an AI agent exactly how to
↪ **dynamically discover peers**,
↪ **read their capabilities**, and
**securely negotiate** UX and state.

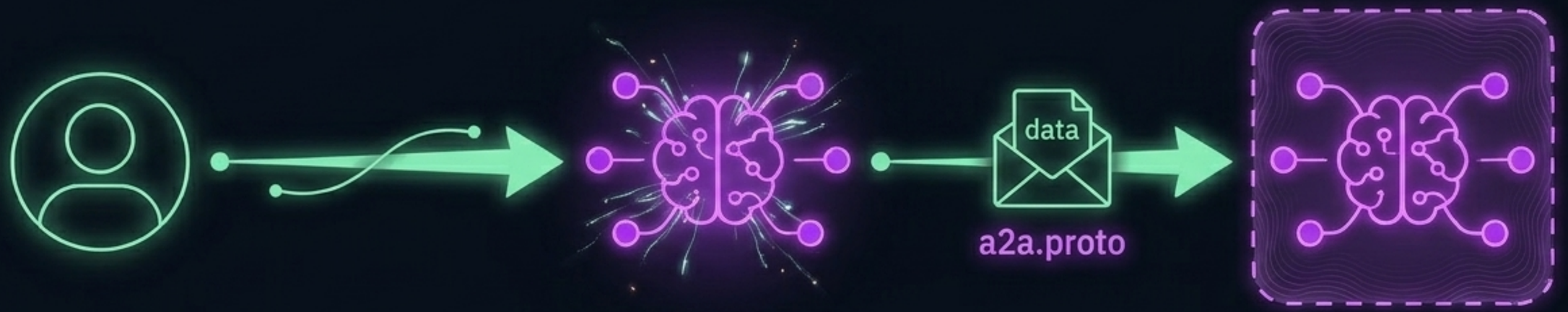
# Composable Layers for Production: MCP vs. A2A

These are not competing standards; they are composable layers for production agentic systems.

| MCP | A2A |
| --- | --- |
| **Integration:** Vertical (Agent ↔ System) | **Integration:** Horizontal Orchestration (Agent ↔ Agent) |
| **State:** Stateless, synchronous request/response | **State:** Deeply stateful, cognitive reasoning |
| **Use Case:** Querying DBs, calling APIs, reading files | **Use Case:** Multi-step planning and delegation between specialized agents |
| **Limitation:** Lacks semantic structures for peer negotiation or long-running task management | |

**Production Pattern:** An A2A meta-agent distributes tasks to specialized sub-agents, which then use MCP to execute deterministic actions.

# The Core Architecture of an A2A Request




**Step 1: User**

Initiates the high-level request for agent assistance.

**Step 2: A2A Client (Client Agent)**

Formulates tasks and acts on the user's behalf to discover capabilities and orchestrate the workflow.

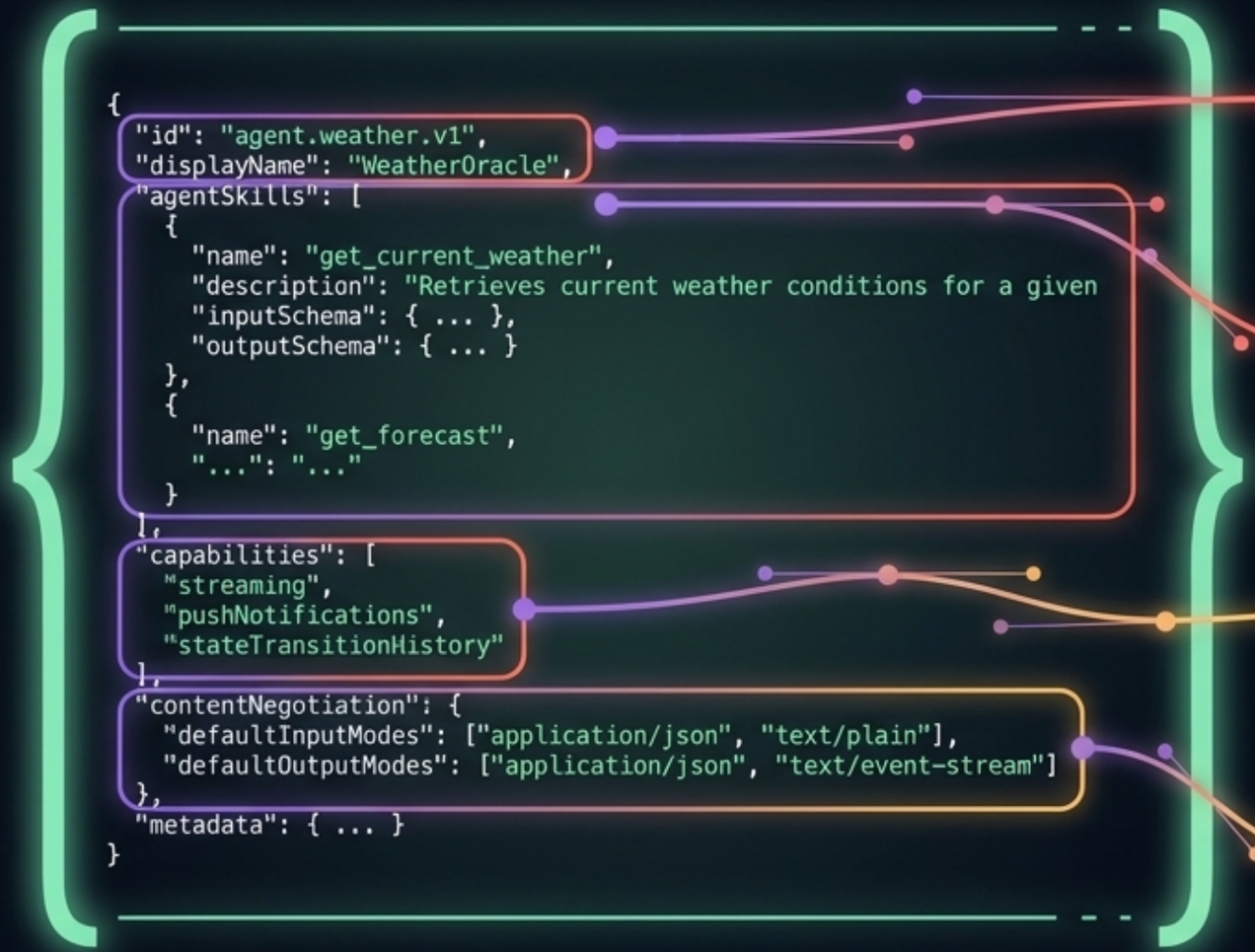
**Step 3: A2A Server (Remote Agent)**

An AI agent providing an HTTP endpoint. It operates as an opaque system.

**The Interoperability Key:**
The client agent does not need to understand the remote agent's internal framework or model provider. All frameworks serialize internal state into the Canonical Data Model (a2a.proto) before transmission, neutralizing framework incompatibility.

# Anatomy of an Agent Card

The digital identity of an agent (JSON format)—the A2A equivalent of MCP's tools/list.




### Identity

The digital fingerprint and routing information.

### AgentSkills

Structured capability declarations (e.g., get_current_weather) with specific input/output schemas.

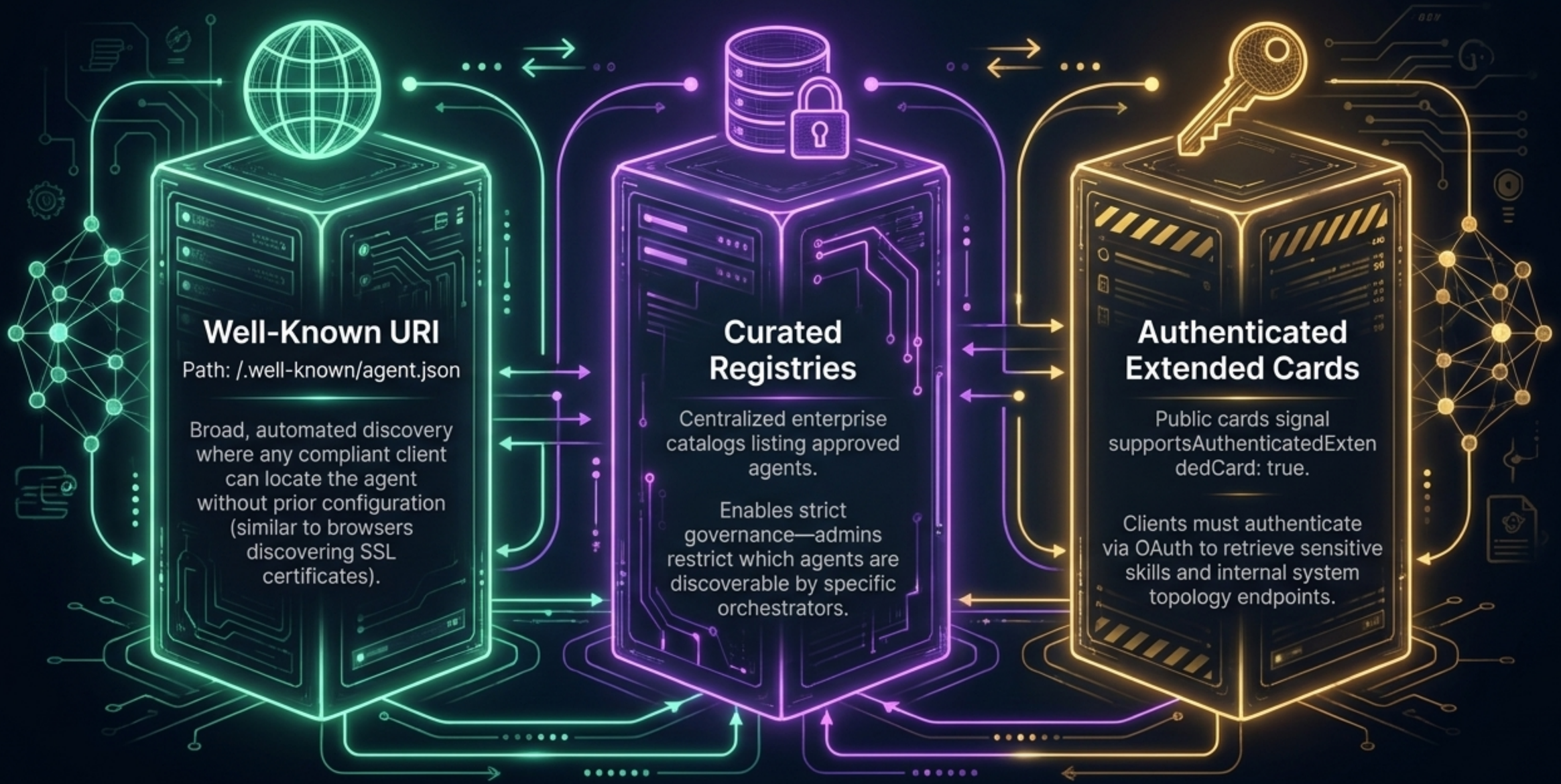
### Capabilities

System-level features like streaming, pushNotifications, and stateTransitionHistory.

### Content Negotiation

defaultInputModes and defaultOutputModes defining supported MIME types.

# Dynamic Agent Discovery Methods



# The Standard Task Workflow

Communication is structured around asynchronous tasks (fundamental units of work) using JSON-RPC 2.0 over HTTP(S).

### 1. Capability Discovery

Client locates the remote agent's Agent Card.

### 2. Task Formulation

Client evaluates the remote agent's skills and drafts the request.

### 3. Execution Request

Payload sent via message/send (sync) or message/stream (persistent SSE connection).

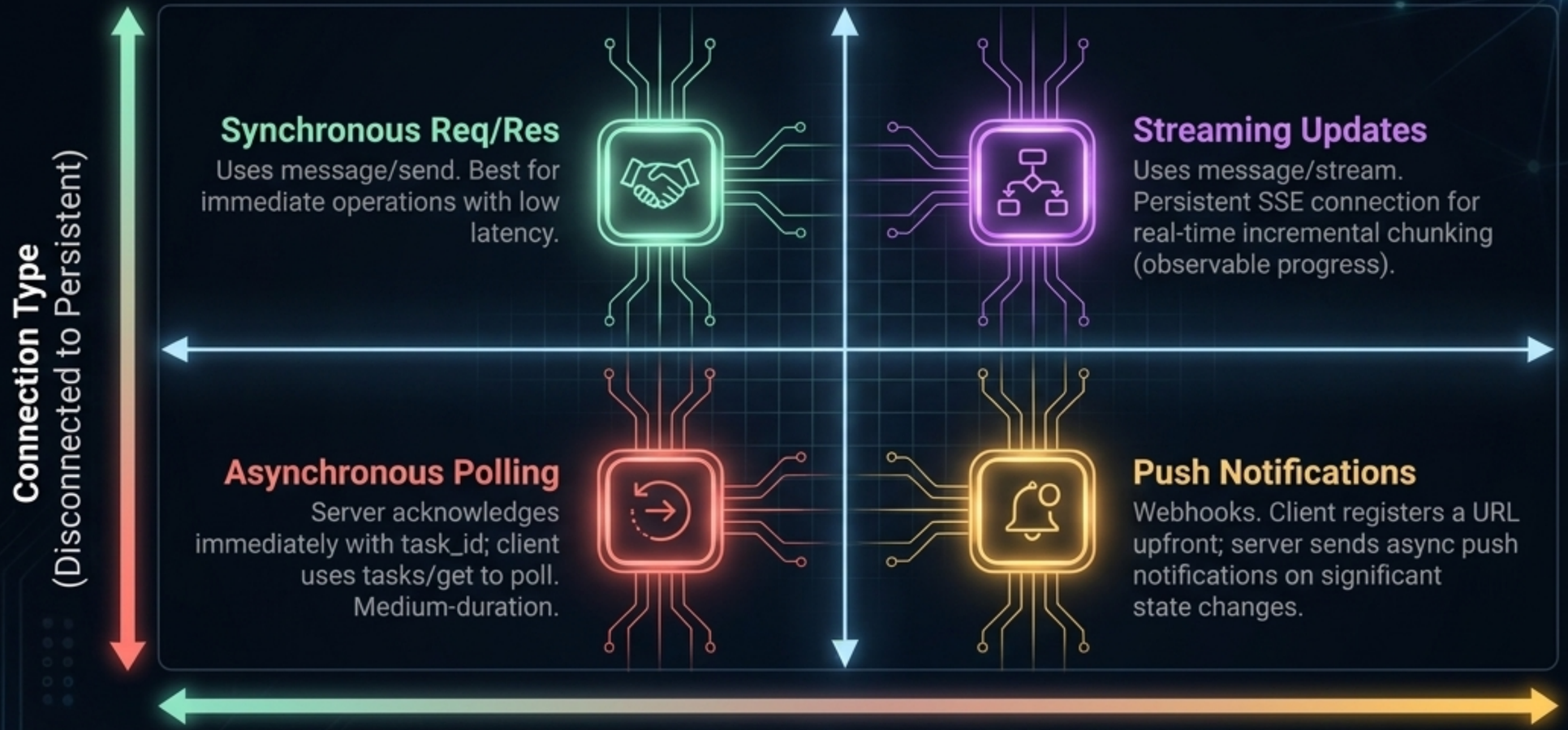
### 4. Collaboration & State

Agents exchange Messages containing key-value attributes and preserve conversational threading.

### 5. Artifact Generation

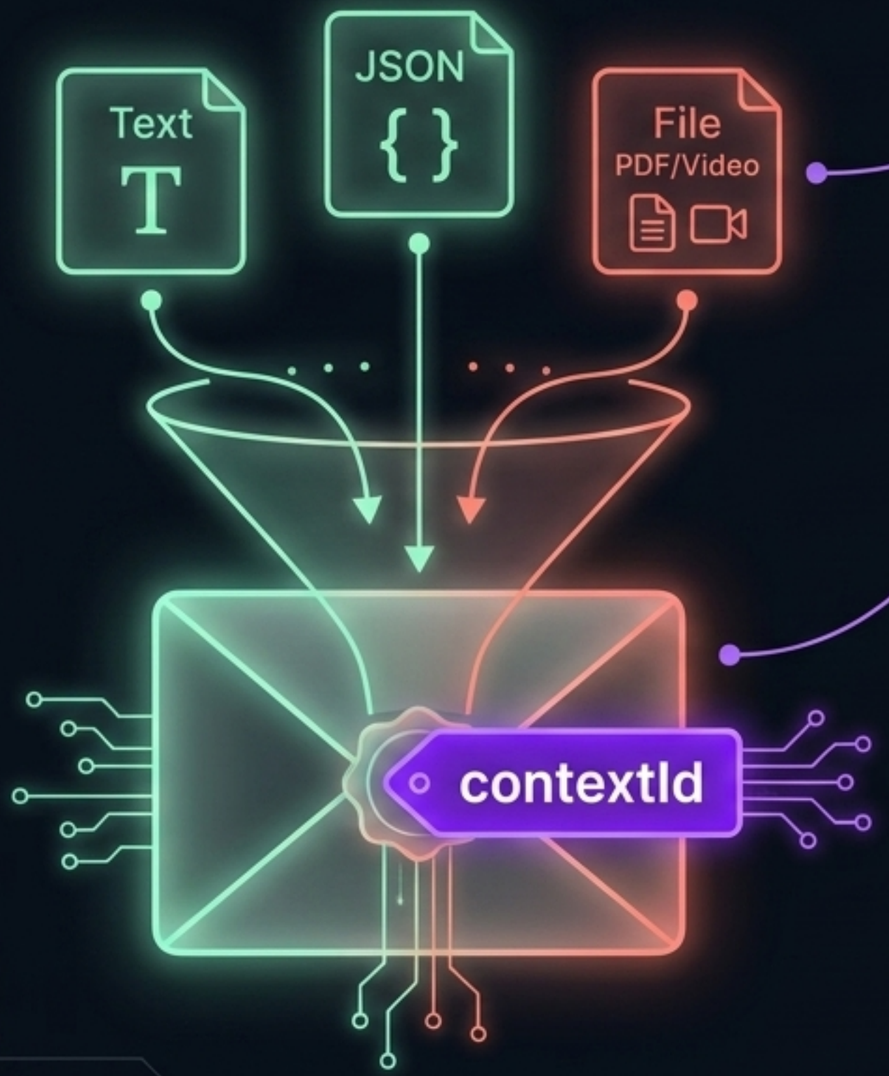
Tangible outputs of the task are streamed incrementally until completion.

# Interaction Mechanisms (Transports)




**Warning Rule:** Match the transport to the task duration. Using synchronous `message/send` for a 4-hour research task will timeout and lose state.

# Modality Agnostic UX & Stateful Continuity




## Unstructured Modalities

A2A embraces natural, unstructured modalities. Message parts can contain TextPart, DataPart (JSON), or FilePart (explicit MIME types).

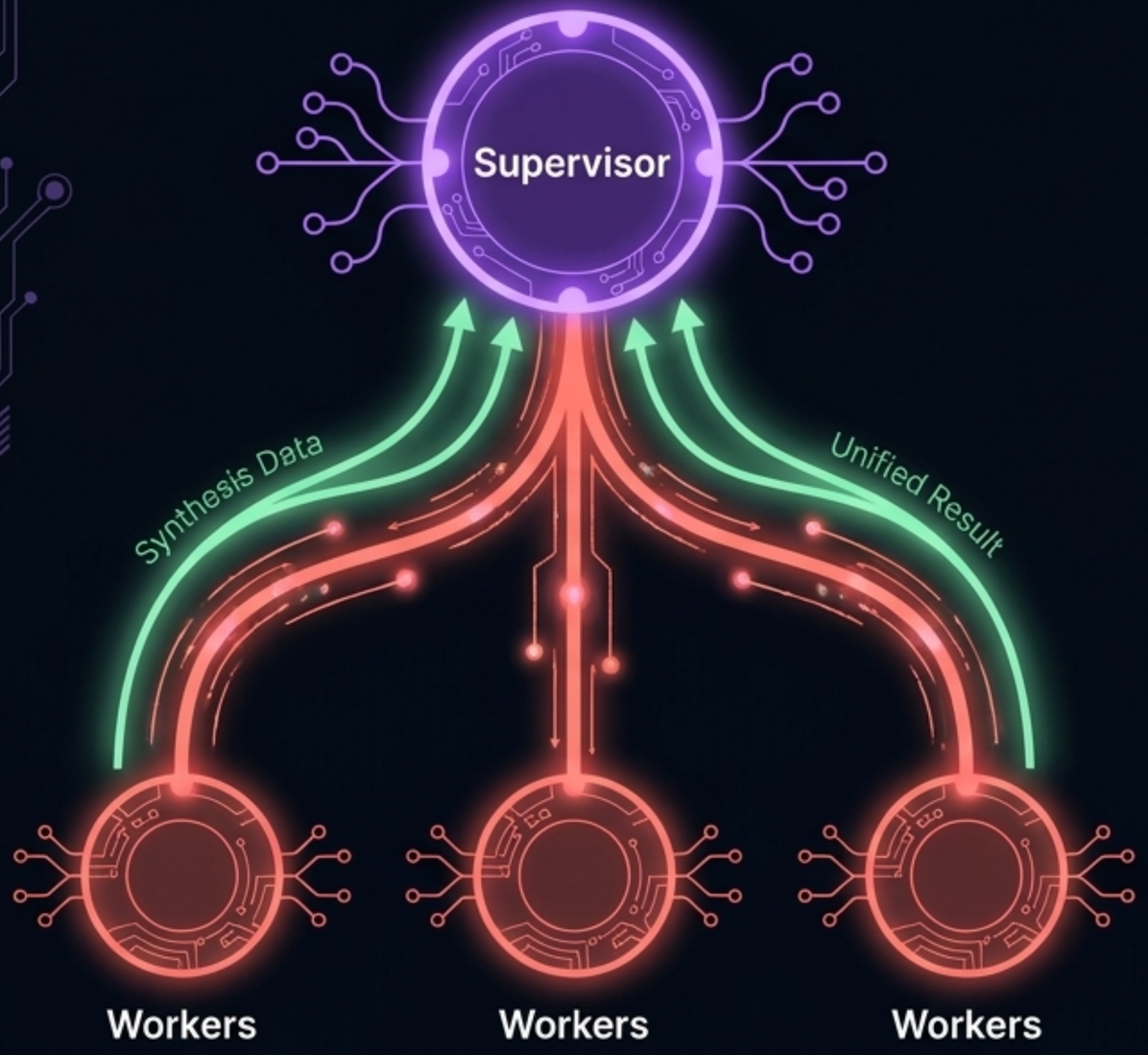
## UX Negotiation

Client agents declare UI capabilities upfront. Remote agents dynamically tailor their response format (e.g., triggering iframes, video streaming, or web forms instead of just text).

## The contextId

A persistent correlation identifier that preserves conversational threading and lineage across multiple asynchronous exchanges.

# Advanced Topology I: Hierarchical Structures




## The Architecture

A central supervisor agent dynamically delegates tasks to specialized worker agents based on the specific capabilities advertised in their Agent Cards.

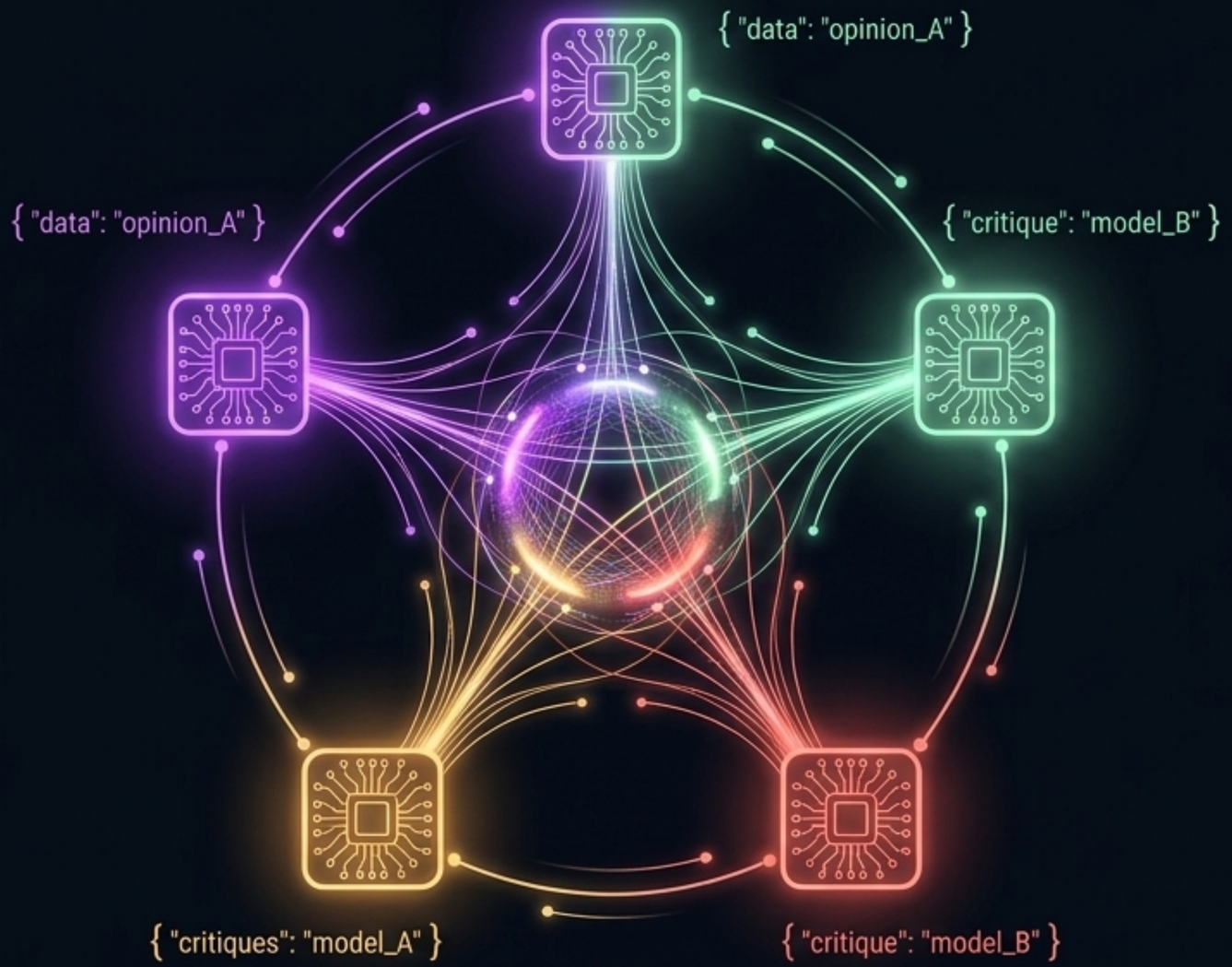
## The Execution

The supervisor synthesizes the disparate outputs from the workers into a single, unified result.

## The Outcome

This is the classic A2A production topology. It directly mirrors how human organizations delegate work and manage complex, multi-disciplinary projects.

# Advanced Topology II: Debate and Consensus




## The Architecture

Multiple distinct agents with varied perspectives, data sources, or underlying model biases are networked together.

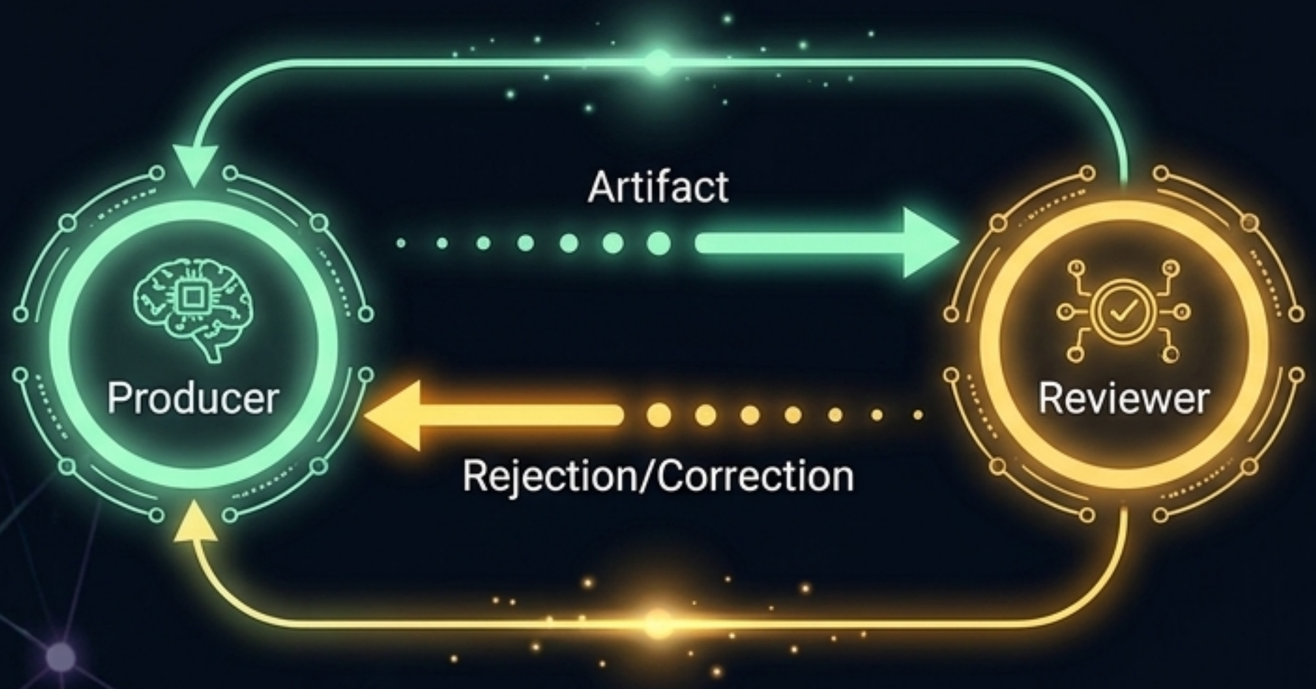
## The Execution

They engage in structured, multi-turn discussions to evaluate options, critique each other's reasoning, and converge on an informed decision.

## The Outcome

Eliminates single-model blind spots. This topology is critical for high-stakes, low-tolerance decisions such as medical triage or financial risk assessment.

# Advanced Topology III: Critic-Reviewer (Self-Correction)




## The Architecture

A producer agent generates a draft artifact. A separate, independent reviewer agent receives this artifact via A2A.

## The Execution

The reviewer critically assesses the artifact for safety, compliance, factual accuracy, and quality before it is allowed to finalize.

## The Outcome

An essential architectural safeguard against hallucinations or non-compliant actions propagating downstream in a broader pipeline.

# Under the Hood: Ecosystem & Implementation Stack


## Design Principle

Embrace agentic capabilities. Allow unstructured modality collaboration without forcing shared memory.

## Ecosystem Support

Natively supported by major platforms including Google Cloud, LangChain, CrewAI, Atlassian, Salesforce, SAP, and MongoDB.

# Security & A2AOps: Enterprise Safeguards

## Mutual TLS (mTLS)

Certificate-based mutual authentication between client and remote agents is mandatory for enterprise-grade encryption.

## HITL (Human-in-the-Loop)

Explicit human approval mechanisms triggered via an input-required task status before proceeding with irreversible actions.

## Strict Header Rules

Secure credentials (OAuth tokens, API keys) must always be passed via HTTP headers. Never embed in URLs (logged by proxies) or message bodies (persisted in Task history).

## Audit Logs & Tracing

Meticulous recording of multi-agent flow, latency tracking, and Task SLA monitoring.

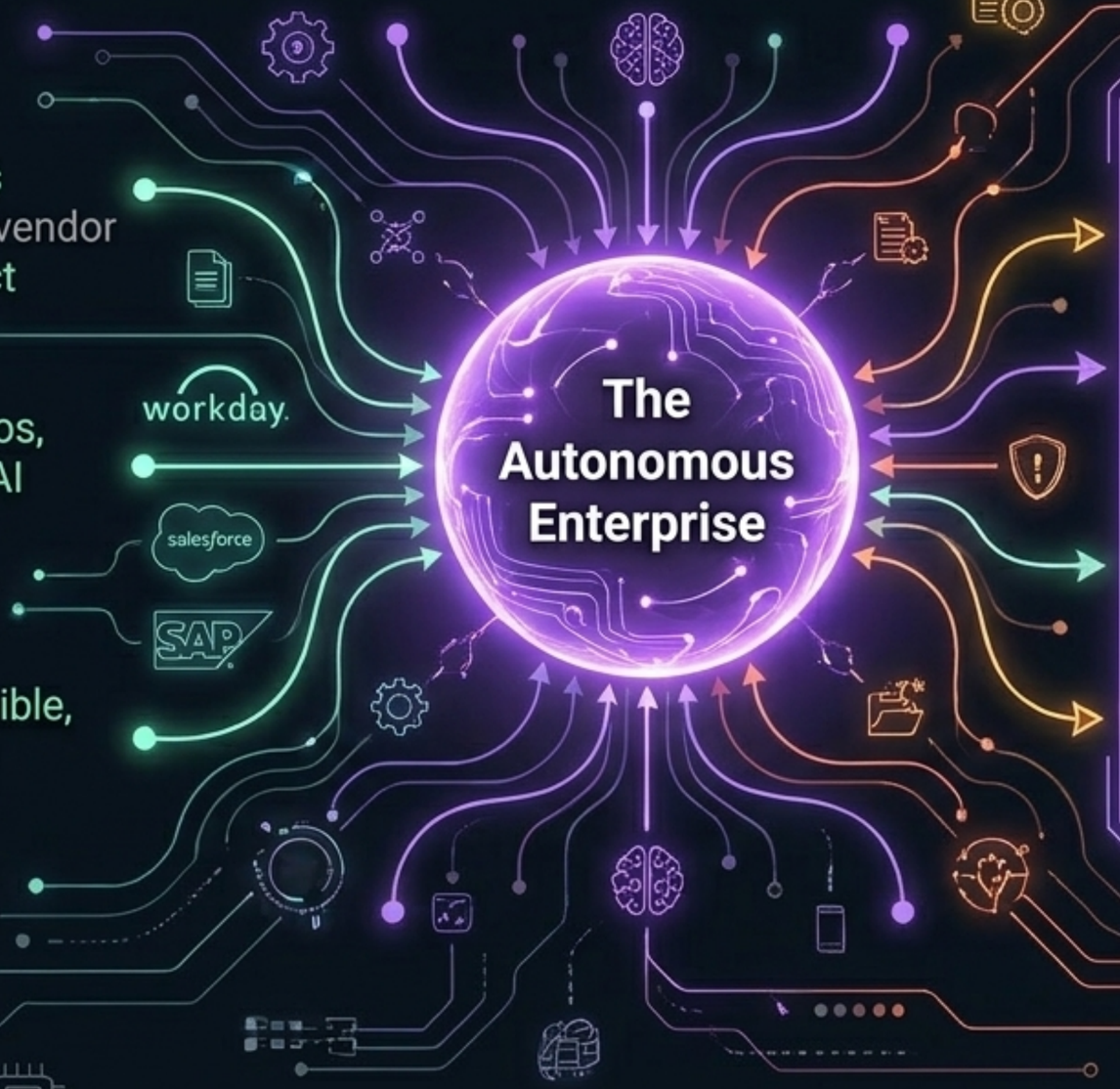
# Realizing the Autonomous Enterprise

## Value Synthesis

For Developers: Eliminates integration glue code and vendor lock-in. Build once, connect anywhere.

For Enterprises: Breaks silos, orchestrating specialized AI teams across Workday, Salesforce, and SAP.

For End-Users: Yields invisible, unified experiences where complex goals are solved autonomously.




## Quick Reference Glossary

**Task**: The fundamental, stateful unit of A2A work. rule-based.

**contextId**: Persistent identifier preserving conversational lineage.

**Artifact**: The structured output passed to the next agent.

**Agentspace**: Unified interface for interacting with A2A-coordinated teams.