What Is Agent-to-Agent Communication?

How AI agents discover, message, and delegate to one another — from agent cards and A2A to on-chain trust.

Alain AI Lab Research · Published July 15, 2026 · 10 min read

A single AI agent is a worker; a group of them is an organisation, and organisations only function if their members can talk to each other. **Agent-to-agent communication** is the machinery that lets one autonomous agent find another, describe what it needs, hand off a task, and receive a result — without a human relaying messages between them. It is the difference between a lone assistant that does everything itself and a network of specialists that divide labour, exactly the shift that makes a multi-agent system more than the sum of its parts. This report explains what that communication actually is, how it works on the wire, and why the question of who an agent can trust is turning out to be the hard part.

AT A GLANCE

CORE IDEA **Structured message passing between agents**	FLAGSHIP PROTOCOL **A2A (Google → Linux Foundation)**	DISCOVERY **JSON "Agent Cards"**
TRANSPORT **JSON-RPC over HTTP, SSE streaming**	COMPLEMENTS **MCP (agent-to-tool layer)**	OPEN PROBLEM **Trust, identity, authorisation**

01

What the Term Actually Means

Strip away the branding and agent-to-agent communication is **structured message passing between programs**. One agent composes a request with a defined shape — not a paragraph of prose, but a machine-readable object with fields, types, and a schema — and sends it to another agent, which parses it, does the work, and returns a result in an equally

defined shape. The contrast that matters is with two humans reading a chat window: humans tolerate ambiguity and infer intent, whereas machines need a contract. That contract is the protocol.

This is why the substrate underneath is **structured output**: JSON schemas, function-calling formats, and typed payloads that can be validated rather than merely read. Free text is where agent messaging goes to fail — a slightly reworded field silently breaks a downstream parser. Schema-conformant messages are what make the exchange reliable enough to build on, and every serious agent-communication effort, old or new, is essentially an agreement about that shape.

02

From KQML to A2A: a Short History

The idea is far older than large language models. In the early 1990s, out of a DARPA knowledge-sharing effort, came **KQML** — the Knowledge Query and Manipulation Language — which wrapped messages in "performatives" borrowed from speech-act theory: an agent could *ask*, *tell*, or *reply*. It was followed by **FIPA-ACL**, standardised by the Foundation for Intelligent Physical Agents (formed in 1996), which defined a message format with a sender, receiver, content, ontology, and a performative such as *inform*, *request*, or *propose*. These were the agent communication languages of classical multi-agent research, and they established the durable insight that agents should exchange *intentions*, not just data.

What is new is the modern incarnation. In April 2025, Google announced the open **Agent2Agent (A2A)** protocol as a way for agents built by different vendors and on different frameworks to interoperate, and in mid-2025 donated it to the Linux Foundation to place it under neutral, vendor-agnostic governance. A2A is the most prominent of a cluster of new protocols, but it inherits a lineage three decades deep: the vocabulary of performatives has become the vocabulary of tasks and messages, dressed in the web standards that now carry them.

03

How Agents Find and Talk to Each Other

Before one agent can delegate to another, it has to **discover** what that other agent can do. A2A solves this with an **Agent Card** — a JSON document, hosted at a standardised well-

known URL, that advertises the agent's identity, its skills, the endpoints to reach it, and the authentication it expects. Capability discovery is the on-ramp: an orchestrator reads the card, learns that a peer can, say, translate documents or price a trade, and only then routes work to it. Registries and directory services generalise the same idea for larger networks.

The conversation itself rides on ordinary web plumbing. A2A is built primarily on **JSON-RPC 2.0 over HTTP(S)**, with **Server-Sent Events** for streaming partial results, and it organises work around three objects: a *task* with a defined lifecycle, *messages* made of typed parts, and *artifacts* that carry the output. Exchanges can be **synchronous** — a blocking request and immediate response — or **asynchronous**, where a long-running task streams updates or notifies the caller through a registered webhook while it works. A task moves through explicit states, from submitted to working to completed or failed, so both sides always know where the job stands.

> *The protocol is not the intelligence. Agent-to-agent communication does not make agents smarter; it makes them composable — and a network of composable agents can attempt work no single one of them was built to finish.*

04

Patterns of Coordination

Once agents can talk, the question becomes how to arrange them. Several coordination patterns recur across the field. The most common is the **orchestrator–worker** (or supervisor) shape: a lead agent decomposes a goal and delegates sub-tasks to specialists, then assembles their answers. **Hierarchical** arrangements nest that structure over several layers. A **blackboard** pattern has agents read from and write to a shared memory rather than message each other directly. **Sequential pipelines** hand a task from one agent to the next like a relay, and **peer-to-peer** or swarm topologies let agents negotiate among themselves without a central conductor.

These patterns are largely framework-level concerns, and today the orchestration is handled by tools such as agent frameworks like LangGraph, CrewAI, and AutoGen, alongside newer entrants like OpenAI's Agents SDK and Google's ADK. A crucial distinction hides here, though: most of these frameworks coordinate agents *they themselves manage*, sharing state directly in memory. That is **intra-framework** communication. A2A targets the harder **inter-framework** case — agents from different vendors, running on different

infrastructure, that share no common runtime and therefore need an open protocol between them.

05

A2A and MCP: Two Layers, Not Two Rivals

A2A is frequently confused with Anthropic's **Model Context Protocol (MCP)**, and the two are routinely, wrongly, framed as competitors. They operate on different axes. MCP is *vertical*: it connects a single agent to the tools, data, and context it needs — an agent-to-tool protocol. A2A is *horizontal*: it connects agents to *each other* as collaborating peers. Google described A2A explicitly as complementing MCP rather than replacing it, and in practice the two are used together — MCP giving an individual agent its reach into systems, A2A letting that agent hand part of a job to another agent entirely.

Keeping the layers distinct matters because the failure modes differ. A weakness in the tool layer leaks data or misuses an API; a weakness in the agent layer lets a bad instruction or a bad actor propagate *across* a network of agents. The two protocols are also not alone: IBM's Agent Communication Protocol (ACP) and the community's Agent Network Protocol (ANP) emerged in the same period as early-stage efforts, though the field is already showing signs of consolidation rather than endless proliferation.

06

The Trust Problem

The moment agents can act on one another's messages, **trust and authentication** become the central difficulty. An agent about to delegate a sensitive task, or hand over data, must first establish that the peer is who it claims to be and is authorised to receive what it is about to get. A2A does not reinvent this — it leans on existing standards such as OAuth 2.0, declared in the Agent Card — but that only covers the familiar case of known parties. In an *open* ecosystem, where agents meet without any prior relationship, verifiable identity and delegated authority remain genuinely unsolved, which is why decentralised-identity approaches (W3C DIDs and verifiable credentials) are being actively proposed for the role.

The security risks are concrete, and OWASP's guidance for agentic systems catalogues them. **Prompt-injection propagation** lets a hostile instruction planted in one agent flow downstream to peers that trust its output. **Impersonation** lets a malicious agent masquerade as a trusted one to exploit that trust. **Cascading failures** let a single

compromise ripple through connected tools, memory, and agents. And **over-permissioning** means an agent often carries more authority than any one task needs, widening the blast radius of any breach. None of these are exotic; they are the ordinary consequences of wiring independent programs together and letting them act.

07

The On-Chain Angle

For crypto specifically, agent-to-agent communication raises a further question: how do agents that do not know each other transact and coordinate with *settlement* and *verifiable identity*? Proponents argue that blockchains are a natural fit — a neutral layer where agents can pay one another, prove who they are, and coordinate without a trusted intermediary. This is an active area of discussion, but it remains early and largely unproven at scale, and much of the loudest advocacy comes from parties with a direct interest in the outcome; it should be read as a proposed architecture, not a demonstrated one.

Two concrete efforts illustrate the direction. ERC-8004, a *draft* Ethereum proposal titled "Trustless Agents," sketches on-chain registries for agent identity, reputation, and validation so that agents can discover and vet each other across organisational boundaries — useful infrastructure, but a draft, not finalised. On the payments side, **x402**, an open standard from Coinbase that revives the long-dormant HTTP 402 "Payment Required" status code, aims to let an agent pay for a resource programmatically, often with stablecoins, and then retry its request. These are nascent building blocks; adoption is early, and the sensible posture is curiosity, not conviction.

08

A Standard Still Being Written

The honest summary is that agent-to-agent communication is real, useful, and unfinished. Multiple protocols — A2A, MCP, ACP, ANP — arrived within a single year, which raises a legitimate **fragmentation risk**: a world of incompatible agent dialects would defeat the whole purpose. Yet the fuller picture is not pure chaos. Several of these protocols address different layers rather than the same one, and early consolidation — efforts folding together under neutral bodies like the Linux Foundation — suggests the field is converging as much as it is splintering. The standards are still forming, and today's clean integration may need revisiting.

What will not change is the underlying shape. Agents that can discover each other, exchange structured intentions, and delegate under verifiable trust are the substrate of every ambitious multi-agent system now being built. The protocols will keep shifting names and versions; the requirement — a common language two agents can agree on before they act — is permanent. For anyone building in the space, the pragmatic move is to treat the communication layer as strategic infrastructure, design for the trust problem from the start rather than bolting it on, and assume the standard you pick this year is a waypoint, not a destination.

“Can two walk together, except they be agreed?”

AMOS 3:3

METHODOLOGY & SOURCES

This report was produced with a parallel multi-agent verification process, each claim checked and labelled verified, partial, or uncertain before inclusion. Figures are kept qualitative where precise public numbers are contested or unconfirmed against a primary source; no adoption counts, transaction volumes, partner tallies, or dollar figures were invented, and several unverified third-party statistics encountered during research were deliberately excluded.

Primary references include Google’s Agent2Agent (A2A) protocol announcement and the Linux Foundation project that now governs it; the A2A specification (Agent Cards, JSON-RPC 2.0 over HTTP, Server-Sent Events, and the task/message/artifact model); Anthropic’s Model Context Protocol documentation for the agent-to-tool distinction; the classical agent communication languages KQML and FIPA-ACL; OWASP’s guidance on agentic-AI security risks including prompt-injection propagation, impersonation, cascading failure, and excessive agency; the draft Ethereum proposal ERC-8004 (“Trustless Agents”), confirmed against the official EIP as a Draft; and Coinbase’s x402 payment standard. Forward-looking crypto-coordination claims are labelled emerging and unproven at scale. Educational content only; not financial advice.

Alain AI Lab — independent crypto & AI research.
intelligencecrypto.org · Research is educational and analytical, not financial advice.