Agent2Agent (A2A) protocol

The Agent2Agent (A2A) protocol is a communication protocol for artificial intelligence (AI) agents, initially introduced by Google in April 2025. This open protocol is designed for multi-agent systems, allowing interoperability between AI agents from varied providers or those built using different AI agent frameworks.

A2A is an open standard for AI agent communication, similar to the Agent Communication Protocol (ACP) introduced by IBM's BeeAI. While earlier agent orchestration frameworks like crewAI and LangChain automate multi-agent workflows within their own ecosystems, the A2A protocol acts as a messaging tier that lets these agents "talk" to each other despite their distinct agentic architectures.

Think of A2A as a common language or universal translator for agent ecosystems. It aims to break down silos, enhancing agent interoperability.

The A2A protocol was initially launched by Google and other technology partners under the Google Cloud platform in April 2025.[1] It's now housed by the Linux Foundation as the open-source Agent2Agent (A2A) project.[2]

What's the difference between MCP and A2A?

Previously introduced by Anthropic in 2024, the Model Context Protocol (MCP) serves as a standardization layer for AI applications to communicate effectively with external services, such as APIs (application programming interfaces), data sources, predefined functions and other tools. Meanwhile, the A2A protocol focuses on agent collaboration, facilitating communication between AI agents.

Both protocols are meant to complement each other. For example, a retail store might have its own inventory agent that uses MCP to interact with databases storing information about products and stock levels. If the inventory agent detects products low in stock, it notifies an internal order agent, which then uses A2A to communicate with external supplier agents and place orders.

information.

Core architectural components of the A2A protocol

The Agent2Agent protocol consists of several building blocks for agent interactions:

- A2A client (client agent)
- A2A server (remote agent)
- Agent card
- Task
- Message
- Artifact
- Part

A2A client (client agent)

The A2A client, also known as the client agent, can be an app, service or other AI agent that delegates requests to remote agents. It uses the Agent2Agent protocol to initiate communication.

A2A server (remote agent)

The A2A server, also called the remote agent, takes requests, processes tasks and responds with status updates or results. It exposes an HTTP endpoint that's compatible with the Agent2Agent protocol.

Agent card

This JSON file outlines agentic AI metadata and can be accessed using a URL. It contains basic information about an agent, including its name, description, version, service endpoint URL, supported modalities or data types and authentication requirements.

Agent cards are similar to model cards for large language models (LLMs). They also advertise an agent's capabilities and skills, serving as a business card, résumé or LinkedIn profile that allows agents to discover each other.

Task

A task represents a unit of work needed to accomplish a request. It has a unique ID and progresses through a lifecycle of defined states (submitted, working, input-required, completed, failed). Tasks are useful for multi-turn processing or long-running agent-to-agent collaboration.

Message

As a fundamental unit of communication, a message depicts a single exchange or turn in a conversation. It contains one or more parts holding the actual content.

Messages relay answers, context, instructions, prompts, questions, replies and status updates. Depending on the sender, each message has an associated role, which can either be an agent role for server-sent messages or a user role for client-sent messages.

Artifact

An artifact is a tangible product generated by the A2A server as a result of its work. It can be a document, image, spreadsheet or any other deliverable. Like messages, artifacts consist of one or more parts and can be incrementally streamed.

Part

A part is a piece of content inside a message or an artifact. Parts have various types based on the data they carry. A TextPart is a vessel for text, a FilePart represents files and a DataPart encompasses structured JSON (JavaScript Object Notation) data.

How the A2A protocol works

The A2A protocol follows a client-server model setup with a three-step workflow:

1. Discovery
2. Authentication
3. Communication

Discovery

The A2A workflow begins when an entity (a human user or another AI agent) initiates a request to the client agent. For instance, a user might ask for assistance in scheduling a trip or an AI agent places an order for a retail store’s low-stock item.

A client agent then proceeds with the process of discovery, looking up remote agents and fetching their agent cards to determine the best fit for the task.

Authentication

Once the client agent identifies a remote agent capable of fulfilling the designated task, it then goes through authentication according to the security scheme indicated in the agent card. A2A supports security schemes aligned with the OpenAPI specification, such as API keys, OAuth 2.0 and OpenID Connect Discovery.

When the client agent has been successfully authenticated, the remote agent is responsible for authorization and granting access control permissions.

Communication

Communication starts with a client agent sending a task to the chosen remote agent. Agent-to-agent communication occurs over HTTPS for secure transport, with JSON-RPC (Remote Procedure Call) 2.0 as the format for data exchange.

The remote agent then processes the task. If it requires more information, it notifies the client agent asking for additional details. Once it completes the task, the remote agent sends a message to the client agent along with any generated artifacts.

A2A also provides task management features for more complex tasks that can't be completed immediately, such as those needing human intervention or involving multiple steps. In the case of long-running tasks that take hours or days or if a client agent gets disconnected, the A2A protocol allows for asynchronous updates through push notifications sent to a secure client-supplied webhook. For large or long outputs or continuous status updates, the A2A protocol supports real-time streaming using server-sent events (SSE).

Benefits of the A2A protocol

The Agent2Agent protocol offers these advantages for agent communication within real-world AI systems:

- Privacy
- Seamless integration
- Security

Privacy

The protocol treats agentic AI as opaque agents. This opacity means autonomous agents can collaborate without having to reveal their inner workings, such as internal memory, proprietary logic or particular tool implementations. This helps preserve data privacy and intellectual property.

Seamless integration

A2A is built on established standards, including HTTP, JSON-RPC and SSE. This makes it easier for enterprises to adopt the protocol and helps ensure compatibility with their current technology stack.

Security

The Agent2Agent protocol is designed with security in mind. It supports enterprise-grade authentication and authorization mechanisms and allows for secure information exchange.

The future of A2A

A2A is still in its early stages, so organizations can expect improvements as the protocol matures. Among these improvements include the formal inclusion of authorization schemes and optional credentials in agent cards, a method for dynamically checking unanticipated or unsupported skills, support for dynamic user experience (UX) negotiation within tasks (such as adding audio or video in the middle of a conversation), and enhancing push notification methods and streaming reliability.[3]

For more information, visit the official A2A site to learn about key concepts, dive into the protocol specification, explore Python tutorials and download software development kits (SDKs). A2A also has a Github site for code samples and demos.