Discover Azure AI Agents with A2A

1. Introduction

https://learn.microsoft.com/en-us/training/modules/discover-agents-with-a2a/1-introduction

Introduction

Completed

- 2 minutes

AI agents are powerful on their own, but many real-world tasks require collaboration across multiple agents. Coordinating these interactions manually can be complex, especially when agents are remote or distributed.

The Agent-to-Agent (A2A) protocol addresses this challenge by providing a standardized framework for agent discovery, communication, and coordinated task execution. By implementing A2A, you can easily manage connections to remote agents, delegate requests to the appropriate agent, and enables seamless communication between agents in a standardized, secure way.

For example, imagine a technical writer who wants to create compelling blog content. One agent generates compelling article headlines, while another creates detailed outlines. Using A2A, a routing agent coordinates the workflow: it sends the user's request to the title agent, passes the generated title to the outline agent, and returns the final outline to the user—all automatically.

In this module, you learn how to implement the A2A protocol with Azure AI Agents. You also practice configuring a routing agent, registering remote agents, and building a coordinated workflow that allows multiple agents to collaborate effectively.

2. Define an A2A agent

https://learn.microsoft.com/en-us/training/modules/discover-agents-with-a2a/2-define-a2a-agent

Define an A2A agent

Completed

- 5 minutes

The **Agent-to-Agent (A2A) protocol** is a standardized way for AI agents to communicate and collaborate with each other. It defines how agents can share context, invoke each other's capabilities, and exchange information securely. By adhering to the A2A protocol, agents from different vendors or platforms can work together seamlessly, enabling more complex and integrated AI solutions.

Before an A2A agent can participate in multi-agent workflows, it needs to explain what it can do. **Agent Skills** and how other agents or clients can discover those capabilities are exposed through an **Agent Card**.

Advantages of the Agent-to-Agent (A2A) protocol

The **Agent-to-Agent (A2A) protocol** offers several advantages for AI agent interactions:

- **Enhanced Collaboration:**
  A2A enables agents from different vendors and platforms to **share context and work together**, allowing seamless automation across systems that are traditionally disconnected.

- **Flexible Model Selection:**
  Each A2A agent can choose which large language model (LLM) to use for handling requests, enabling **optimized or fine-tuned models per agent**, unlike some MCP scenarios that rely on a single LLM connection.

- **Integrated Authentication:**
  Authentication is built into the A2A protocol, providing a **robust security framework** for secure agent-to-agent communication.

Agent Skills

An Agent Skill describes a specific capability or function that the agent can perform. Think of it as a building block that communicates to clients or other agents what tasks the agent is designed to handle.

Key elements of an Agent Skill include:

- ID: A unique identifier for the skill.
- Name: A human-readable name describing the skill.
- Description: A detailed explanation of what the skill does.
- Tags: Keywords for categorization and easier discovery.
- Examples: Sample prompts or use cases to illustrate the skill in action.
- Input/Output Modes: Supported data formats or media types (for example, text, JSON).

When defining a skill for your agent, consider the tasks it should perform, how to describe them clearly, and how other agents or clients might use them. For example, a simple "Hello World" skill could return a basic greeting in text format, whereas a blog-writing skill might accept a topic and return a suggested title or outline.

Agent Card

The Agent Card is like a digital business card for your agent. It's a structured document that a routing agent or client can retrieve to discover your agent’s capabilities and how to interact with it.

Key elements of an Agent Card include:

- Identity Information: Name, description, and version of the agent.
- Endpoint URL: Where the agent’s A2A service can be accessed.
- Capabilities: Supported A2A features such as streaming or push notifications.
- Default Input/Output Modes: The primary media types the agent can handle.
- Skills: A list of the agent’s skills that other agents can invoke.
- Authentication Support: Indicates if the agent requires credentials for access.

When creating an Agent Card, ensure it accurately represents your agent’s skills and endpoints. This allows clients or routing agents to discover the agent, understand what it can do, and interact with it appropriately.

Putting it together

Once an agent defines its skills and publishes an Agent Card:

- Other agents or clients can discover the agent automatically.
- Requests can be routed to the agent’s appropriate skill.
- Responses are returned in supported formats, enabling smooth collaboration across multiple agents.

For example, in a technical writer workflow, one agent could define skills for generating article titles, and another for creating outlines. The routing agent retrieves each agent’s card to discover these capabilities and orchestrates a workflow where a title generated by one agent feeds into the outline agent, producing a cohesive final response.

3. Implement an agent executor

https://learn.microsoft.com/en-us/training/modules/discover-agents-with-a2a/3-implement-agent-executor

Implement an agent executor

Completed

- 5 minutes

The **Agent Executor** is a core component of an A2A agent. It defines how your agent processes incoming requests, generates responses, and communicates with clients or other agents. Think of it as the bridge between the A2A protocol and your agent's specific business logic.

Understand the Agent Executor

The `AgentExecutor` interface handles all incoming requests sent to your agent. It receives information about the request, processes it according to the agent’s capabilities, and sends responses or events back through a communication channel.

Key responsibilities:

- Execute tasks requested by users or other agents.
- Stream responses or send individual messages back to the client.
- Handle task cancellation if supported.

Implement the interface

An Agent Executor typically defines two primary operations:

Execute

- Processes incoming requests and generates responses.
- Accesses request details (for example, user input, task context).
- Sends results back via an event queue, which may include messages, task updates, or artifacts.

Cancel

- Handles requests to cancel an ongoing task.
- May not be supported for simple agents.

The executor uses the **RequestContext** to understand the incoming request and an **EventQueue** to communicate results or events back to the client.

Request handling flow

Consider a "Hello World" agent workflow:

1. The agent has a small helper class that implements its core logic (for example, returning a string).
2. The executor receives a request and calls the agent’s logic.
3. The executor wraps the result as an event and places it on the event queue.
4. The routing mechanism sends the event back to the requester.

For cancellation, a basic agent might only indicate that cancellation isn't supported.

The Agent Executor is central to making your A2A agent functional. It defines how the agent executes tasks and communicates results, providing a standardized interface for clients and other agents. Properly implemented executors enable seamless integration and collaboration in multi-agent workflows.

4. Host an A2A server

https://learn.microsoft.com/en-us/training/modules/discover-agents-with-a2a/4-host-a2a-agent-server

Host an A2A server

Completed

- 5 minutes

Once your agent defines its skills and Agent Card, the next step is to host it on a server. Hosting makes your agent accessible to clients and other agents over HTTP, enabling real-time interactions and multi-agent workflows.

Hosting an agent allows it to:

- Expose its capabilities through its **Agent Card**, which clients and other agents can discover.
- Receive incoming A2A requests and forward them to your **Agent Executor** for processing.
- Manage task lifecycles, including streaming responses and stateful interactions.

Effectively, the server acts as a bridge between your agent's logic and the external world, ensuring it can participate in coordinated workflows.

Core components of the agent server

To host an agent, you need three essential components working together:

Agent Card

- Describes the agent's capabilities, skills, and input/output modes.
- Exposed at a standard endpoint (typically `/.well-known/agent-card.json`) so clients and other agents can discover your agent.
- Can include multiple versions or an "extended" card for authenticated users.

Request Handler

- Routes incoming requests to the appropriate methods on your **Agent Executor** (for example, `execute` or `cancel`).
- Manages the task lifecycle using a **Task Store**, which tracks tasks, streaming data, and resubscriptions.
- Even simple agents require a task store to handle interactions reliably.

Server Application

- Built using a web framework (Starlette in Python) to handle HTTP requests.
- Combined with an ASGI server (like Uvicorn) to start listening on a network interface and port.
- Exposes the agent card and request handler endpoints, enabling clients to interact with your agent.

Set up the A2A agent server

1. Define your agent’s skills and Agent Card.
2. Initialize a request handler that links your **Agent Executor** with a **Task Store**.
3. Set up the server application, providing the Agent Card and request handler.
4. Start the server using an ASGI server (Uvicorn) to make it accessible on the network.
5. Once running, the agent listens for incoming requests and responds according to its defined skills.

A "Hello World" agent may expose a basic greeting skill. Once hosted, it can respond to any requests sent to its endpoint. A more complex agent can serve multiple skills or an extended Agent Card for authenticated users.

Hosting an A2A agent combines the Agent Card, request handler, and agent executor to make it available for client and agent interactions. This setup ensures tasks are managed correctly and responses are delivered reliably, enabling your agent to participate in multi-agent workflows.

5. Connect to your A2A agent

https://learn.microsoft.com/en-us/training/modules/discover-agents-with-a2a/5-connect-to-a2a-agent

Connect to your A2A agent

Completed

- 5 minutes

Once your A2A agent server is running, the next step is understanding how a client can interact with it. A client acts as the bridge between your application and the agent server.

The client responsibilities include:

- Discovering the Agent Card, which contains metadata about the agent and its endpoints.
- Sending requests to the agent for processing.
- Receiving and interpreting the agent's responses, which can be either direct messages or task-based results.

Connect to your agent server

- The client must know the **base URL** of the server.
- The client typically retrieves the Agent Card from a well-known endpoint on the server.
- Once the Agent Card is obtained, the client can be initialized with it, establishing a connection ready to send messages.

Send requests to the agent

There are two main types of requests a client can make:

- **Non-Streaming Requests:** The client sends a message and waits for a complete response. This type of request is suitable for simple interactions or when a single response is expected.
- **Streaming Requests:** The client sends a message and receives responses incrementally as the agent processes the request. This type of request is useful for long-running tasks or when you want to update the user in real-time.

In both cases, requests usually include a `role` (fo example, user) and the message content. More complex agents may return **task objects** instead of immediate messages, allowing for task tracking or cancellation.

Handle the agent response

Agent responses may include:

- **Direct messages:** Immediate outputs from the agent, such as text or structured content.
- **Task-based responses:** Objects representing ongoing tasks, which may require follow-up calls to check status or retrieve results.

Clients should be prepared to handle both response types and interpret the returned data appropriately.

Interacting with the agent

- Each request should be uniquely identifiable, often using a generated ID.
- Streaming responses are asynchronous and may provide partial results before the final output.
- Simple agents may return messages directly, while more advanced agents may manage multiple tasks simultaneously.

Connecting a client to your agent server involves fetching the Agent Card, establishing a connection, sending requests, and handling responses. By grasping these core concepts, you can confidently interact with your remote agent, whether you're sending simple messages or managing complex tasks.

6. Exercise - Connect to remote Azure AI Agents with the A2A protocol

https://learn.microsoft.com/en-us/training/modules/discover-agents-with-a2a/6-exercise

Exercise - Connect to remote Azure AI Agents with the A2A protocol

Completed

- 30 minutes

If you have an Azure subscription, you can complete this exercise to develop an A2A client-server application that interacts with remote agents.

Note

If you don't have an Azure subscription, you can sign up for an account, which includes credits for the first 30 days.

Launch the exercise and follow the instructions.

Launch Exercise

7. Module assessment

https://learn.microsoft.com/en-us/training/modules/discover-agents-with-a2a/7-knowledge-check

Module assessment

Completed

- 3 minutes

8. Summary

https://learn.microsoft.com/en-us/training/modules/discover-agents-with-a2a/8-summary

Summary

Completed

- 1 minute

In this module, you learned how to connect Python clients to Azure AI Agents using the Agent-to-Agent (A2A) protocol.

By running an A2A server and connecting your client, you explored how agents are discovered and communicated with dynamically using the Agent Card. You learned about executors, which handle agent requests, and how messages—both streaming and non-streaming—flow between clients and agents. Understanding these concepts allows you to build flexible, discoverable agent networks that can delegate tasks and respond to requests across distributed environments.