Develop an AI agent with Microsoft Agent Framework

1. Introduction

https://learn.microsoft.com/en-us/training/modules/develop-ai-agent-with-semantic-kernel/1-introduction

Introduction

Completed

- 2 minutes

AI agents are transforming how applications interact with users and automate tasks. Unlike traditional programs, AI agents use generative AI to interpret data, make decisions, and complete tasks with minimal human intervention. These agents use large language models to streamline complex workflows, making them ideal for automating business processes.

Developers can build AI agents using different tools, including the Microsoft Agent Framework. This open-source SDK simplifies the integration of AI models into applications. The Microsoft Agent Framework supports different types of agents from multiple providers, including Microsoft Foundry, Azure OpenAI, OpenAI, Microsoft Copilot Studio, and Anthropic agents. This module focuses on Microsoft Foundry Agents, which provide enterprise-grade capabilities using the Microsoft Foundry Agent Service.

Microsoft Foundry Agent Service is a fully managed service that enables developers to securely build, deploy, and scale high-quality extensible AI agents. Using the Foundry Agent Service, developers don't need to manage the underlying compute or storage resources. The Microsoft Agent Framework enables developers to quickly build agents on the Foundry Agent Service, supporting natural language processing and providing access to built-in tools in just a few lines of code.

While Foundry Agent Service provides a powerful foundation for building AI agents, the Microsoft Agent Framework offers more flexibility and scalability. If your solution requires multiple types of agents, using the Microsoft Agent Framework ensures consistency across your implementation.

Finally, if you're planning to develop multi-agent solutions, the framework's workflow orchestration features allow you to coordinate collaborative agents efficiently—a topic covered in more detail in a later module.

Suppose you need to develop an AI agent that automatically formats and emails expense reports for employees. Your AI agent can extract data from submitted expense reports, format them correctly, and send them to the appropriate recipients when you use the Microsoft Agent Framework. The tools and functions feature allows your AI agent to interact with APIs, retrieve necessary data, and complete tasks.

In this module, you learn about the core features of the Microsoft Agent Framework SDK. You also learn how to create your own AI agents and extend their capabilities with tool functions.

After completing this module, you're able to:

- Use the Microsoft Agent Framework to connect to a Microsoft Foundry project.
- Create Microsoft Foundry agents using the Microsoft Agent Framework.
- Integrate tool functions with your AI agent.

2. Understand Microsoft Agent Framework AI agents

https://learn.microsoft.com/en-us/training/modules/develop-ai-agent-with-semantic-kernel/2-understand-semantic-kernel-agents

Understand Microsoft Agent Framework AI agents

Completed

- 6 minutes

An AI agent is a program that uses generative AI to interpret data, make decisions, and perform tasks on behalf of users or other applications. AI agents rely on large language models to perform their tasks. Unlike traditional programs, AI agents can function autonomously, handling complex workflows and automating processes without requiring continuous human oversight.

AI Agents can be developed using many different tools and platforms, including the Microsoft Agent Framework. The Microsoft Agent Framework is an open-source SDK that enables developers to easily integrate the latest AI models into their applications. This framework provides a comprehensive foundation for creating functional agents that can use natural language processing to complete tasks and collaborate with other agents.

Microsoft Agent Framework core components

The Microsoft Agent Framework offers different components that can be used individually or combined.

- **Chat clients** - provide abstractions for connecting to AI services from different providers under a common interface. Supported providers include Azure OpenAI, OpenAI, Anthropic, and more through the `BaseChatClient` abstraction.
- **Function tools** - containers for custom functions that extend agent capabilities. Agents can automatically invoke functions to integrate with external APIs and services.
- **Built-in tools** - prebuilt capabilities including Code Interpreter for Python execution, File Search for document analysis, and Web Search for internet access.
- **Conversation management** - structured message system with roles (USER, ASSISTANT, SYSTEM, TOOL) and `AgentThread` for persistent conversation context across interactions.
- **Workflow orchestration** - supports sequential workflows, concurrent execution, group chat, and handoff patterns for complex multi-agent collaboration.

The Microsoft Agent Framework helps streamline the creation of agents and allows multiple agents to work together in conversations while including human input. The framework supports different types of agents from multiple providers, including Microsoft Foundry, Azure OpenAI, OpenAI, Microsoft Copilot Studio, and Anthropic agents.

What Is a Microsoft Foundry Agent?

Microsoft Foundry Agents provide enterprise-level capabilities using the Microsoft Foundry Agent Service. These agents offer advanced features for complex enterprise scenarios. Key benefits include:

- **Enterprise-level capabilities** – Built for Azure environments with advanced AI features including code interpreter, function tools integration, and Model Context Protocol (MCP) support.
- **Automatic tool invocation** – Agents can automatically call and execute tools, integrating seamlessly with Azure AI Search, Azure Functions, and other Azure services.
- **Thread and conversation management** – Provides built-in mechanisms for managing persistent conversation states across sessions, ensuring smooth multi-agent interactions.

- **Secure enterprise integration** – Enables secure and compliant AI agent development with Azure CLI authentication, RBAC, and customizable storage options.

When you use Microsoft Foundry Agents, you get the full power of enterprise Azure capabilities combined with the features of the Microsoft Agent Framework. These features can help you create robust AI-driven workflows that can scale efficiently across business applications.

Agent framework core concepts

- **BaseAgent** - the foundation for all agents with consistent methods, providing a unified interface across all agent types.
- **Agent threads** - manage persistent conversation context and store conversation history across sessions using the `AgentThread` class.
- **Chat messages** - organized structure for agent communication using role-based messaging (USER, ASSISTANT, SYSTEM, TOOL) that enables smooth communication and integration.
- **Workflow orchestration** - supports sequential workflows, running multiple agents in parallel, group conversations between agents, and transferring control between specialized agents.
- **Multi-modal support** - allows agents to work with text, images, and structured outputs, including vision capabilities and type-safe response generation.
- **Function tools** - let you add custom capabilities to agents by including custom functions with automatic schema generation from Python functions.
- **Authentication methods** - supports multiple authentication methods including Azure CLI credentials, API keys, MSAL for Microsoft business authentication, and role-based access control.

This framework supports autonomous, multi-agent AI behaviors while maintaining a flexible architecture that lets you mix and match agents, tools, and workflows as needed. The design lets you switch between OpenAI, Azure OpenAI, Anthropic, and other providers without changing your code, making it easy to build AI systems—from simple chatbots to complex business solutions.

3. Create an Azure AI agent with Microsoft Agent Framework

https://learn.microsoft.com/en-us/training/modules/develop-ai-agent-with-semantic-kernel/3-create-azure-ai-agent

Create an Azure AI agent with Microsoft Agent Framework

Completed

- 7 minutes

Microsoft Foundry Agent is a specialized agent within the Microsoft Agent Framework, designed to provide enterprise-level conversational capabilities with seamless tool integration. It automatically handles tool calling, so you don't need to manually parse and invoke functions. The agent also securely manages conversation history using threads, which reduces the work of maintaining state. The Microsoft Foundry Agent supports many built-in tools, including code interpreter, file search, and web search. It also provides integration capabilities for Azure AI Search, Azure Functions, and other Azure services.

Creating a Microsoft Foundry Agent

A Microsoft Foundry Agent includes all the core capabilities you typically need for enterprise AI applications, like function execution, planning, and memory access. This agent acts as a self-contained runtime with enterprise-level features.

To use a Microsoft Foundry Agent:

1. Create a Microsoft Foundry project.
2. Add the project connection string to your Microsoft Agent Framework application code.
3. Set up authentication credentials.
4. Create a `ChatAgent` with an `AzureAIAgentClient`.
5. Define tools and instructions for your agent.

Here's the code that shows how to create a Microsoft Foundry Agent:

```
 from agent_framework import AgentThread, ChatAgent
from agent_framework.azure import AzureAIAgentClient
from azure.identity.aio import AzureCliCredential

def get_weather(
    location: Annotated[str, Field(description="The location to get the weather for.")],
) -> str:
    """Get the weather for a given location."""
    return f"The weather in {location} is sunny with a high of 25°C."

# Create a ChatAgent with Azure AI client
```

```
async with (
    AzureCliCredential() as credential,
    ChatAgent(
        chat_client=AzureAIAgentClient(async_credential=credential),
        instructions="You are a helpful weather agent.",
        tools=get_weather,
    ) as agent,
):
    # Agent is now ready to use
```

Once your agent is created, you can create a thread to interact with your agent and get responses to your questions. For example:

```
 # Create the agent thread for ongoing conversation
thread = agent.get_new_thread()

# Ask questions and get responses
first_query = "What's the weather like in Seattle?"
print(f"User: {first_query}")
first_result = await agent.run(first_query, thread=thread)
print(f"Agent: {first_result.text}")
```

Microsoft Foundry Agent key components

The Microsoft Agent Framework Microsoft Foundry Agent uses the following components to work:

- **AzureAIAgentClient** - manages the connection to your Microsoft Foundry project. This client lets you access the services and models associated with your project and provides enterprise-level authentication and security features.
- **ChatAgent** - the main agent class that combines the client, instructions, and tools to create a working AI agent that can handle conversations and complete tasks.
- **AgentThread** - automatically keeps track of conversation history between agents and users, and manages the conversation state. You can create new threads or reuse existing ones to maintain context across interactions.
- **Tools integration** - support for custom functions that extend agent capabilities. Functions are automatically registered and can be called by agents to connect with external APIs and services.
- **Authentication credentials** - supports Azure CLI credentials, service principal authentication, and other Azure identity options for secure access to Foundry Tools.
- **Thread management** - provides flexible options for thread creation, including automatic thread creation for simple scenarios and explicit thread management for ongoing conversations.

These components work together to let you create enterprise-level agents with instructions to define their purpose and get responses from AI models while maintaining security, scalability, and conversation context for business applications.

4. Add tools to Azure AI agent

https://learn.microsoft.com/en-us/training/modules/develop-ai-agent-with-semantic-kernel/4-add-plugins-to-agent

Add tools to Azure AI agent

Completed

- 5 minutes

In the Microsoft Agent Framework, tools allow your AI agent to use existing APIs and services to perform tasks it couldn't do on its own. Tools work through function calling, allowing AI to automatically request and use specific functions. The framework routes the request to the appropriate function in your codebase and returns the results back to the large language model (LLM) so it can generate a final response.

To enable automatic function calling, tools need to provide details that describe how they work. The function's input, output, and purpose should be described in a way that the AI can understand, otherwise, the AI can't call the function correctly.

How to use tools with Microsoft Foundry Agent

The Microsoft Agent Framework supports both custom function tools and built-in tools that are ready to use out of the box.

Built-in tools

Microsoft Foundry Agents come with several built-in tools that you can use immediately:

- **Code Interpreter** - executes Python code for calculations, data analysis, and more
- **File Search** - searches through and analyzes documents
- **Web Search** - retrieves information from the internet

These tools are automatically available and don't require any extra setup.

Custom function tools

When creating custom tools for your Microsoft Foundry Agent, you need to understand several key concepts:

1. **Function definition and annotations**

   Create your tool by defining a regular Python function with proper type annotations. Use `Annotated` and `Field` from Pydantic to provide detailed descriptions that help the AI understand the function's purpose and how to use its parameters. The more descriptive your annotations, the better the AI can understand when and how to call your function.

2. **Adding tools to your agent**

   Pass your custom functions to the ChatAgent during creation using the `tools` parameter. You can add a single function or a list of multiple functions. The framework automatically registers these functions and makes them available for the AI to call.

3. **Tool invocation through conversation**

   Once your tools are registered with the agent, you don't need to manually invoke them. Instead, ask the agent questions or give it tasks that would naturally require your tool's functionality. The AI automatically determines when to call your tools based on the conversation context and the tool descriptions you provided.

4. **Multiple tools and orchestration**

   You can add multiple tools to a single agent, and the AI automatically chooses which tool to use based on the user's request. The framework handles the orchestration, calling the appropriate functions and combining their results to provide a comprehensive response.

Best practices for tool development

- **Clear descriptions**: Write clear, detailed descriptions for your functions and parameters to help the AI understand their purpose
- **Type annotations**: Use proper Python type hints to specify expected input and output types
- **Error handling**: Implement appropriate error handling in your tool functions to gracefully handle unexpected inputs
- **Return meaningful data**: Ensure your functions return data that the AI can effectively use in its responses
- **Keep functions focused**: Design each tool to handle a specific task rather than trying to do too many things in one function

By following these concepts, you can extend your Microsoft Foundry Agent with both built-in and custom tools, allowing it to interact with APIs and perform advanced tasks. This approach makes your AI more powerful and capable of handling real-world applications efficiently.

5. Exercise - Develop an Azure AI agent with the Microsoft Agent Framework SDK

https://learn.microsoft.com/en-us/training/modules/develop-ai-agent-with-semantic-kernel/5-exercise

Exercise - Develop an Azure AI agent with the Microsoft Agent Framework SDK

Completed

- 30 minutes

Now you're ready to build an agent with the Microsoft Agent Framework. In this exercise, you use the Microsoft Agent Framework SDK to create an AI agent that creates an expense claim email.

Note

To complete this exercise, you will need a Microsoft Azure subscription. If you don't already have one, you can sign up for one.

If you need to set up your computer for this exercise, you can use this setup guide and then follow the exercise instructions linked below. Note that the setup guide is designed for multiple development exercises, and may include software that is not required for this specific exercise. Additionally, due to the range of possible operating systems and setup configurations, we can't provide support if you choose to complete the exercise on your own computer.

Launch the exercise and follow the instructions.

Launch Exercise

Tip

After completing the exercise, if you're finished exploring Azure AI Agents, delete the Azure resources that you created during the exercise.

6. Knowledge check

https://learn.microsoft.com/en-us/training/modules/develop-ai-agent-with-semantic-kernel/6-knowledge-check

Knowledge check

Completed

- 3 minutes

7. Summary

https://learn.microsoft.com/en-us/training/modules/develop-ai-agent-with-semantic-kernel/7-summary

Summary

Completed

- 2 minutes

In this module, you learned how the Microsoft Agent Framework enables developers to build AI agents. You learned about the components and core concepts of the Microsoft Agent Framework. You also learned how to create custom tools to extend your agent's capabilities. By applying these concepts and skills, you can use the Microsoft Agent Framework to create dynamic, adaptable AI solutions that enhance user interactions and automate complex tasks.