**Multi-agent Coordination Patterns(MCP)**

Multi-Agent Coordination Patterns (MCP) refer to structured methods or models used to enable multiple autonomous agents to work together efficiently within a shared environment. In a multi-agent system (MAS), several independent agents interact with each other to achieve individual or collective objectives. These agents may have different goals, capabilities, or knowledge about the environment, which makes coordination a crucial aspect of system design. MCPs provide standard solutions to recurring problems related to coordination, cooperation, and communication among agents.

A diagram of a system

AI-generated content may be incorrect.

**Definition**

A Multi-Agent Coordination Pattern is a general and reusable solution for handling the interaction and organization of multiple agents. It defines how agents share information, distribute tasks, resolve conflicts, and make decisions collectively. MCPs ensure that agents behave in a way that promotes system efficiency, stability, and goal achievement without centralized control in most cases.

**Objectives of MCP**

1. **Coordination:**  
   Ensures that agents perform their actions in an organized manner to avoid redundancy and conflicts. Coordination is necessary when tasks depend on one another or when shared resources are involved.
2. **Collaboration:**  
   Promotes joint effort among agents to achieve shared objectives that cannot be completed individually. Collaboration involves planning, information sharing, and sometimes negotiation.
3. **Negotiation:**  
   Agents may have different goals or priorities. MCPs provide negotiation mechanisms that help them reach agreements beneficial to the entire system.
4. **Communication:**  
   Enables agents to exchange data, updates, and intentions effectively. The communication strategy defines when, how, and what information is shared to maintain consistency and prevent information overload.
5. **Adaptability:**  
   In dynamic environments, agents must adapt to new conditions. MCPs provide frameworks for agents to adjust their behaviors based on changes in the environment or the actions of other agents.

A diagram of a data stream

AI-generated content may be incorrect.

**Common Multi-Agent Coordination Patterns**

**1. Master–Slave (Leader–Follower) Pattern**

In this pattern, one agent acts as the **leader** (or master) and others act as **followers** (or slaves). The leader is responsible for planning, decision-making, and assigning tasks to followers, who execute the instructions. This pattern ensures centralized control and simplifies coordination but reduces system autonomy.

**Advantages:**

* Simple and easy to implement.
* Effective when agents have unequal capabilities.
* Suitable for hierarchical organizations.

**Disadvantages:**

* Single point of failure (if the leader fails, the system may collapse).
* Limited flexibility and scalability.

**Example:**  
In a fleet of drones performing surveillance, one drone acts as the leader, assigning sectors to other drones and collecting their data.

A diagram of a company's organization

AI-generated content may be incorrect.

**2. Contract Net Protocol (CNP)**

This pattern is based on a **market-like bidding mechanism**. A manager agent announces a task, and other agents (contractors) bid to take it based on their capabilities and current workload. The manager selects the most suitable contractor for the task.

**Advantages:**

* Promotes distributed decision-making.
* Efficient task allocation based on agent specialization.
* Increases scalability and flexibility.

**Disadvantages:**

* High communication overhead due to frequent bidding.
* Decision-making delays if the network is large.

**Example:**  
In distributed manufacturing, each machine (agent) bids for production tasks based on its capacity and availability.

**3. Blackboard Pattern**

In this pattern, a shared data structure called a **blackboard** is used for communication. All agents read and write information to this common space instead of direct communication with each other. The blackboard acts as a mediator that stores the current state of the environment and partial results.

**Advantages:**

* Simplifies communication among agents.
* Promotes collaboration through shared knowledge.
* Facilitates modular design, as agents can be added or removed easily.

**Disadvantages:**

* Possible bottleneck due to centralized data storage.
* Requires synchronization mechanisms to prevent data inconsistency.

**Example:**  
In autonomous robotics, multiple robots update a shared map of the area on the blackboard to plan their paths collaboratively.

A diagram of a blackboard

AI-generated content may be incorrect.

**4. Mediator or Broker Pattern**

In this approach, a **mediator agent** handles communication and coordination between agents. Instead of agents interacting directly, they send messages to the mediator, who forwards them to appropriate recipients or manages the flow of information.

**Advantages:**

* Reduces direct dependencies among agents.
* Centralized communication management simplifies monitoring.
* Useful in dynamic environments where agents frequently join or leave.

**Disadvantages:**

* The mediator becomes a single point of failure.
* Increases dependency on the mediator’s efficiency.

**Example:**  
In customer service systems, a broker agent connects customer agents to service agents based on request type and availability.

**5. Coalition Formation Pattern**

In this pattern, agents **form temporary groups or coalitions** to solve specific problems collaboratively. Once the task is completed, the coalition dissolves. Coalition formation involves negotiation and benefit-sharing among members.

**Advantages:**

* Enhances problem-solving capability through cooperation.
* Efficient resource utilization by grouping compatible agents.
* Encourages flexible and dynamic team formation.

**Disadvantages:**

* Complex negotiation and coalition management.
* Possible conflicts in benefit distribution.

**Example:**  
In smart city systems, agents representing transport, energy, and communication sectors may form a coalition to manage traffic during peak hours.

**6. Consensus-Based Pattern**

In this approach, all agents work together to reach a **collective decision or consensus**. It is commonly used in distributed control systems where agents must agree on values like direction, speed, or task priority.

**Advantages:**

* Promotes decentralized control.
* Increases fault tolerance and system robustness.
* Ensures uniformity in agent decisions.

**Disadvantages:**

* High communication cost.
* Delays in reaching consensus in large networks.

**Example:**  
In sensor networks, all sensors agree on a common environmental reading through consensus algorithms.

**Real-World Applications of MCP**

1. **Healthcare:**  
   Coordination among medical agents for scheduling, diagnosis, and treatment planning. For instance, in hospital automation, doctor agents, nurse agents, and equipment agents collaborate to optimize patient care.
2. **Mobility and Transportation:**  
   Used in autonomous vehicle coordination where multiple self-driving cars communicate to maintain safe distances, optimize routes, and avoid collisions.
3. **Customer Service Systems:**  
   Chatbots and virtual assistants act as agents coordinated by mediator or broker patterns to respond efficiently to user queries.
4. **Retail and E-commerce:**  
   Agents coordinate pricing, inventory management, and customer recommendations. For example, recommendation agents collaborate with inventory agents to suggest available products.
5. **Smart Cities:**  
   Agents coordinate in domains like energy distribution, waste management, and traffic control to optimize city operations.

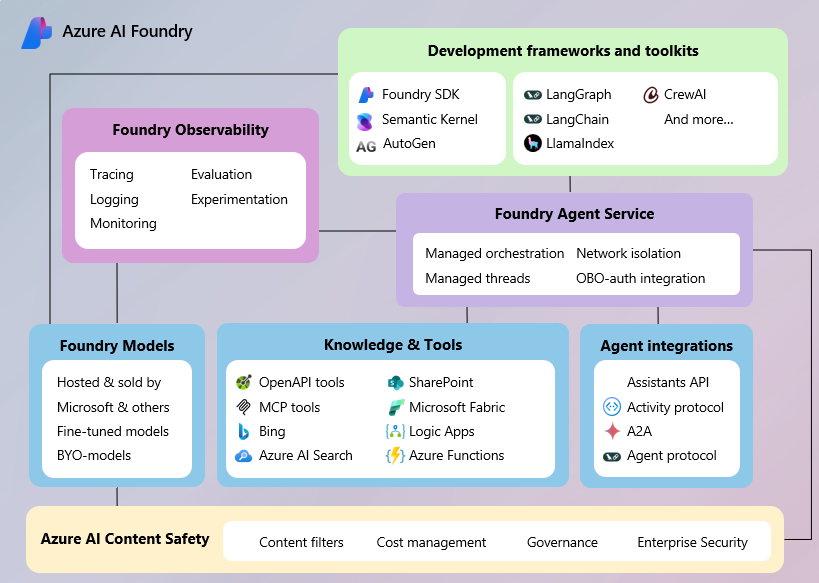
**Conclusion**

Multi-Agent Coordination Patterns provide a structured way to design, manage, and optimize the interaction among agents in complex systems. By defining how agents communicate, share resources, and make decisions collectively, MCPs improve efficiency, flexibility, and scalability. The choice of pattern depends on the system’s requirements — whether it needs centralized control, distributed decision-making, or cooperative behavior. As multi-agent systems continue to evolve, MCPs play an essential role in enabling intelligent, adaptive, and autonomous coordination across diverse domains such as robotics, smart cities, healthcare, and e-commerce.

**Azure AI Foundry Agent Service**

Large language models (LLMs) opened the door to a new type of automation with systems that can understand unstructured data, make decisions, and generate content. In practice, it can be difficult for businesses to move beyond demos and into production. LLMs can drift, be incorrect, and lack accountability. Without visibility, policy enforcement, and orchestration, these models are difficult to trust in real business workflows.

**Azure AI Foundry** is designed to change that. It's a platform that combines models, tools, frameworks, and governance into a unified system for building intelligent agents. At the center of this system is **Azure AI Foundry Agent Service**, enabling the operation of agents across development, deployment, and production.



AI Foundry Agent Service connects the core pieces of Azure AI Foundry such as models, tools, and frameworks into a single runtime. It manages threads, orchestrates tool calls, enforces content safety, and integrates with identity, networking, and observability systems to ensure agents are secure, scalable, and production-ready.

By abstracting away infrastructure complexity and enforcing trust and safety by design, AI Foundry Agent Service makes it easy to move from prototype to production with confidence.

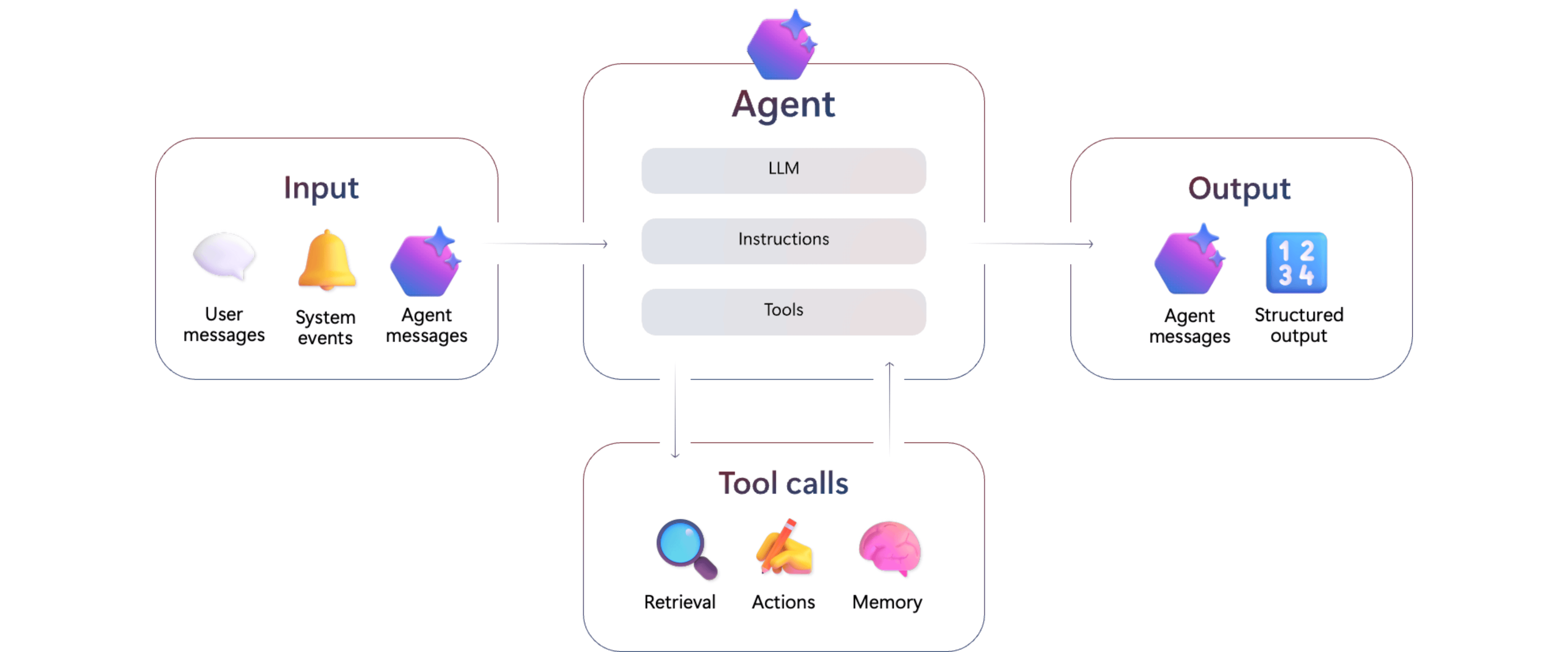
**What is an AI Agent?**

Agents make decisions, invoke tools, and participate in workflows. Sometimes independently, sometimes in collaboration with other agents or humans. What sets agents apart from assistants is autonomy: assistants support people, agents complete goals. They are foundational to real process automation.

Agents created using AI Foundry are not monoliths. They are composable units. Each with a specific role, powered by the right model, and equipped with the right tools, and deployed within a secure, observable, and governable runtime.

Each agent has three core components:

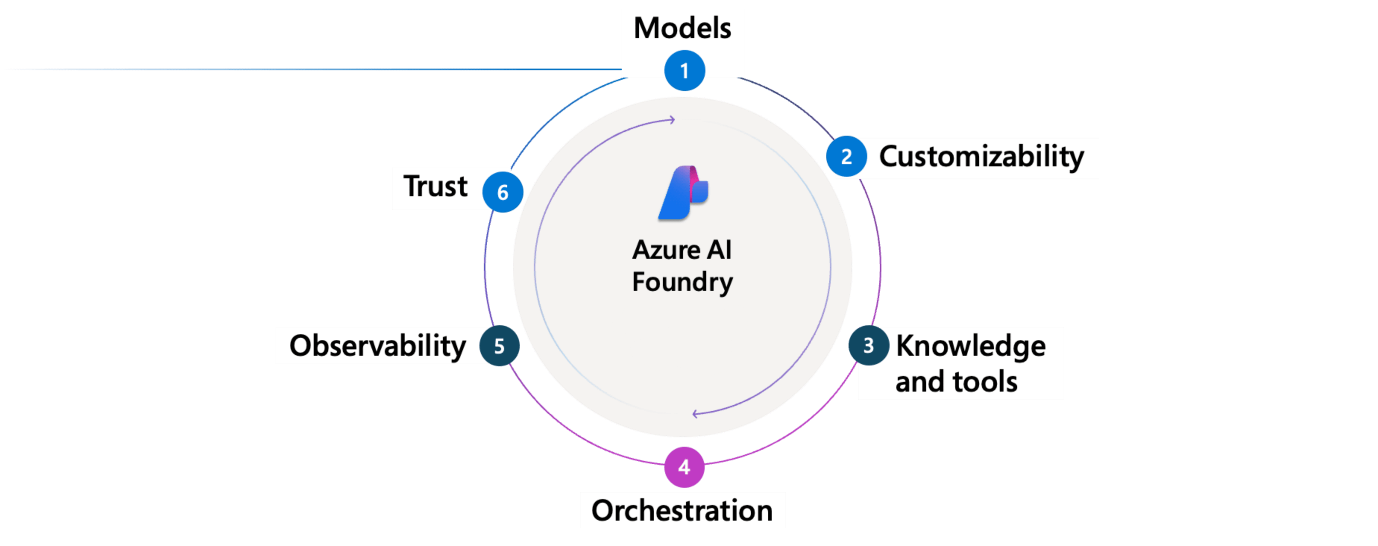
* **Model (LLM)**: Powers reasoning and language understanding
* **Instructions**: Define the agent’s goals, behavior, and constraints
* **Tools**: Let the agent retrieve knowledge or take action



Agents receive unstructured inputs such as user prompts, alerts, or messages from other agents. They produce outputs in the form of tool results or messages. Along the way, they may call tools to perform retrieval, or trigger actions.

**How do agents in AI Foundry work?**

Think of Azure AI Foundry as an assembly line for intelligent agents. Like any modern factory, it brings together different specialized stations, each responsible for shaping part of the final product. Instead of machines and conveyor belts, the Agent Factory uses models, tools, policies, and orchestration to build agents that are secure, testable, and production-ready. Here’s how the factory works step by step:



**1. Models**

The assembly line starts by selecting a model that gives your agent its intelligence. Choose from a growing catalog of large language models including GPT-4o, GPT-4, GPT-3.5 (Azure OpenAI), and others like Llama. This is the reasoning core of the agent that powers its decisions.

**2. Customization**

Next, shape that model to fit your use case. Customize your agent with fine-tuning, distillation, or domain-specific prompts. This step allows you to encode agent behavior, role-specific knowledge, and patterns from prior performance using data captured from real thread content and tool results.

**3. AI Tools**

Then, equip your agent with tools. These let it access enterprise knowledge (such as Bing, SharePoint, Azure AI Search) and take real-world actions (via Logic Apps, Azure Functions, OpenAPI, and more). This enhances the agent's ability to expand its capabilities.

**4. Orchestration**

Next, the agent needs coordination. [Connected agents](https://learn.microsoft.com/en-us/azure/ai-foundry/agents/how-to/connected-agents) orchestrates the full lifecycle such as handling tool calls, updating thread state, managing retries, and logging outputs.

**5. Observability**

Finally, agents are tested and monitored. AI Foundry can capture logs, traces, and evaluations at every step. With full thread-level visibility and Application Insights integration, teams can inspect every decision and continuously improve agents over time.

**6. Trust**

It's important to ensure agents are suitable and reliable for the workload they're assigned to. AI Foundry applies enterprise-grade trust features including identity via Microsoft Entra, RBAC, content filters, encryption, and network isolation. You choose how and where your agents run - using platform-managed or bring-your-own infrastructure.

The result? An agent that's ready for production: reliable, extensible, and safe to deploy across your workflows.

**Why Use Azure AI Foundry Agent Service?**

Azure AI Foundry Agent Service provides a production-ready foundation for deploying intelligent agents in enterprise environments. Here's how it compares across key capabilities:

| **Capability** | **Azure AI Foundry Agent Service** |
| --- | --- |
| **1. Visibility into conversations** | Full access to structured [threads](https://learn.microsoft.com/en-us/azure/ai-foundry/agents/concepts/threads-runs-messages#threads), including both user↔agent and agent↔agent messages. Ideal for UIs, debugging, and training |
| **2. Multi-agent coordination** | Built-in support for agent-to-agent messaging. |
| **3. Tool orchestration** | Server-side execution and retry of [tool calls](https://learn.microsoft.com/en-us/azure/ai-foundry/agents/how-to/tools/overview) with structured logging. No manual orchestration required. |
| **4. Trust and safety** | Integrated [content filters](https://learn.microsoft.com/en-us/azure/ai-foundry/openai/how-to/content-filters) help prevent misuse and mitigate prompt injection risks (XPIA). all outputs are policy-governed. |
| **5. Enterprise integration** | Bring your own [storage](https://learn.microsoft.com/en-us/azure/ai-foundry/agents/how-to/use-your-own-resources#use-an-existing-azure-cosmos-db-for-nosql-account-for-thread-storage), [Azure AI Search index](https://learn.microsoft.com/en-us/azure/ai-foundry/agents/how-to/use-your-own-resources#use-an-existing-azure-ai-search-resource), and [virtual network](https://learn.microsoft.com/en-us/azure/ai-foundry/agents/how-to/virtual-networks) to meet compliance needs. |
| **6. Observability and debugging** | Threads, tool invocations, and message traces are [fully traceable](https://learn.microsoft.com/en-us/azure/ai-foundry/how-to/develop/trace-agents-sdk); [Application Insights integration](https://learn.microsoft.com/en-us/azure/ai-foundry/agents/how-to/metrics) for telemetry |
| **7. Identity and policy control** | Built on Microsoft Entra with full support for RBAC, audit logs, and enterprise conditional access. |

**Get started with Foundry Agent Service**

To get started with Foundry Agent Service, you need to create an Azure AI Foundry project in your Azure subscription.

Start with the [environment setup](https://learn.microsoft.com/en-us/azure/ai-foundry/agents/environment-setup) and [quickstart](https://learn.microsoft.com/en-us/azure/ai-foundry/agents/quickstart) guide if it's your first time using the service.

1. You can create a project with the required resources.
2. After you create a project, you can deploy a compatible model such as GPT-4o.
3. When you have a deployed model, you can also start making API calls to the service using the SDKs.

**What are tools in Azure AI Foundry Agent Service?**

To empower your AI agent with grounded data or the capability to take actions and automating workflows, the Foundry Agent service provides a wide range of built-in tools, such as Grounding with Bing Search, Azure AI Search, Azure Logic Apps, as well as third-party partner tools, such as Tripadvisor. This page is designed to provide an overview of tools provided in the Foundry Agent Service.

**Knowledge tools**

To keep your AI agent informed with richer context from various data sources. The Foundry Agent Service has covered a wide range of data types:

* **private data**: Azure AI Search, File Search, Microsoft Fabric, and more
* **public web data**: Grounding with Bing Search
* **licensed data**: Tripadvisor, Morningstar
* **unstructured data**: Azure AI Search, File Search
* **structured data**: Microsoft Fabric and more

**Action tools**

To streamline workflows with your AI agent with capabilities to take actions. The Foundry Agent Service provides different action tools for you with different level of flexibility, control, and ease of integration:

* **Deep Research tool**: Web-based integrated deep research pipeline with the o3-deep-research model and Grounding with Bing Search.
* **Azure Logic Apps**: Low-code / no-code solution to add a workflow to your AI Agent
* **OpenAPI Spec tool**: Bring an existing OpenAPI specification of a service API you want to add to your AI agent, with no or minor changes.
* **MCP tool**: Bring an existing Model Context Protocol (MCP) endpoint that you want to add to your AI agent.
* **Function calling**: Write your own custom, stateless functions to define the expected behaviors.
* **Azure Functions**: Write and manage your own custom, stateful functions.
* **Browser Automation**: Perform real-world browser tasks through natural language prompts.

**How does a tool work in the Foundry Agent Service?**

Tools are optional capabilities you can add to your AI agent for AI models to decide and pick based on the user query and context. When a user sends a query, the AI model identifies the intent with the context and potentially rewrites the user query. Then the AI model decides which tools to be called for each run. For example, if you add both the Grounding with Bing Search tool and the Azure AI Search tool to your agent and ask "*what is the weather in Seattle today?*", the model will identify your intent to ask about real-time information and more likely to invoke the Grounding with Bing Search tool.

You can add tools at the agent, thread, or run level. By providing tools at a narrower level, the tool resources will **override** tool resources at a broader level. For example, tool resources at the run level override tool resources at thread level. Currently, you can add multiple tools but you can add **one instance of each** of the following tools: File Search, Azure AI Search, Grounding with Bing Search, Grounding with Bing Custom Search, Microsoft Fabric, and other tools under knowledge section.

When a user sends a query to the agent, it will create a [thread, run, and message](https://learn.microsoft.com/en-us/azure/ai-foundry/agents/concepts/threads-runs-messages). For each run, the AI model decides what tools to invoke based on the user intent and available tool resources. Based on the tool outputs, the AI model might decide to invoke another tool or call the same tool again to get more context. For example, when you use Grounding with Bing Search tool, you might see multiple Bing Search queries when [tracing a thread](https://learn.microsoft.com/en-us/azure/ai-foundry/how-to/develop/trace-agents-sdk). This means the AI model actually calls the Grounding with Bing Search tool multiple times with different queries to get more information. If you want to learn more about what tools are called and how the AI model invokes them, check the run step details.

There are various ways to influence how your AI agent invokes tools:

* The tool\_choice parameter: Most deterministic way of controlling which (if any) tool is called by the model. By default, it is set to auto, which means the AI model will decide. If you want to **force** the model to call a specific tool, you can provide the specification of this tool, for example

PythonCopy

run = project\_client.agents.runs.create\_and\_process(

thread\_id=thread.id,

agent\_id=agent.id,

tool\_choice={"type": "bing\_grounding"} # specify the tool to use

)

* The instructions parameter: Nondeterministic. Use the instructions to help the AI model understand your use case and the purposes of each tool. You want to tell the AI model what information or actions each tool can do. For example "*use the AI Search tool <tool\_name> for product related information, use the Fabric tool <tool\_name> for sales related information*." Sometimes the user query can be responded by the model's base knowledge or by the tools, you want to provide instructions like "*use the tool outputs to generate a response, don't use your own knowledge.*"

**Prerequisites**

* [A created agent](https://learn.microsoft.com/en-us/azure/ai-foundry/agents/quickstart)
* Make sure your AI model has enough Tokens-Per-Minute (TPM) allocated. We recommend having a minimum of 30k TPM. You can change the TPM allocation by going to **models + endpoints** in the [AI Foundry portal](https://ai.azure.com/?cid=learnDocs) and edit your model.

**Built-in tools**

The Foundry Agent Service provides the following built-in tools. You can use them with the REST API, SDK, and Azure AI Foundry portal.

| **Tool** | **Description** |
| --- | --- |
| [Azure AI Search](https://learn.microsoft.com/en-us/azure/ai-foundry/agents/how-to/tools/azure-ai-search) | Use an existing Azure AI Search index to ground agents with data in the index, and chat with your data. |
| [Azure Functions](https://learn.microsoft.com/en-us/azure/ai-foundry/agents/how-to/tools/azure-functions) | Leverage your Azure Functions to create intelligent, event-driven applications. |
| [Browser Automation](https://learn.microsoft.com/en-us/azure/ai-foundry/agents/how-to/tools/browser-automation) | Perform real-world browser tasks through natural language prompts. |
| [Code Interpreter](https://learn.microsoft.com/en-us/azure/ai-foundry/agents/how-to/tools/code-interpreter) | Enable agents to write and run Python code in a sandboxed execution environment. |
| [Deep Research (preview)](https://learn.microsoft.com/en-us/azure/ai-foundry/agents/how-to/tools/deep-research) | Use OpenAI's advanced agentic research capability for analysis and reasoning. |
| [File Search](https://learn.microsoft.com/en-us/azure/ai-foundry/agents/how-to/tools/file-search) | Augment agents with knowledge from outside its model, such as proprietary product information or documents provided by your users. |
| [Function calling](https://learn.microsoft.com/en-us/azure/ai-foundry/agents/how-to/tools/function-calling) | Describe the structure of functions you create to an agent and have them be called when appropriate during the agent's interactions with users. |
| [Grounding with Bing Search](https://learn.microsoft.com/en-us/azure/ai-foundry/agents/how-to/tools/bing-grounding) | Enable your agent to use Grounding with Bing Search to access and return information from the internet. |
| [Grounding with Bing Custom Search (preview)](https://learn.microsoft.com/en-us/azure/ai-foundry/agents/how-to/tools/bing-custom-search) | Enhance your Agent response with selected web domains |
| [Model Context Protocol (preview)](https://learn.microsoft.com/en-us/azure/ai-foundry/agents/how-to/tools/model-context-protocol) | Give the agent access to tools hosted on an existing MCP endpoint |
| [Microsoft Fabric (preview)](https://learn.microsoft.com/en-us/azure/ai-foundry/agents/how-to/tools/fabric) | Integrate your agent with the [Microsoft Fabric data agent](https://go.microsoft.com/fwlink/?linkid=2312815) to unlock powerful data analysis capabilities. |
| [OpenAPI 3.0 Specified tool](https://learn.microsoft.com/en-us/azure/ai-foundry/agents/how-to/tools/openapi-spec) | Connect your Azure AI Agent to external APIs using functions with an OpenAPI 3.0 specification. |

**Non-Microsoft tools**

The following tools are authored by third-party partners. Use the links below to view the documentation and code samples.

| **Tool** | **Description** |
| --- | --- |
| [Auquan](https://github.com/azure-ai-foundry/foundry-samples/tree/main/samples/microsoft/python/getting-started-agents/3p-tools/auquan) | AI-powered workflow automation for institutional finance |
| [Celonis](https://github.com/azure-ai-foundry/foundry-samples/tree/main/samples/microsoft/python/getting-started-agents/3p-tools/Celonis) | Celonis delivers Process Intelligence to accelerate enterprise AI at scale |
| [InsureMO Insurance Quotation](https://github.com/azure-ai-foundry/foundry-samples/tree/main/samples/microsoft/python/getting-started-agents/3p-tools/InsureMO) | Action APIs for insurance quotations for Car, Home, and Travel |
| [LEGALFLY](https://github.com/azure-ai-foundry/foundry-samples/tree/main/samples/microsoft/python/getting-started-agents/3p-tools/legalfly) | Legal insights grounded in trusted sources from your jurisdiction. |
| [LexisNexis](https://github.com/azure-ai-foundry/foundry-samples/tree/main/samples/microsoft/python/getting-started-agents/3p-tools/LexisNexis) | Seamless access to LexisNexis content. |
| [MiHCM](https://github.com/azure-ai-foundry/foundry-samples/tree/main/samples/microsoft/python/getting-started-agents/3p-tools/MiHCM) | seamless integration with MiHCM's HR functionalities |
| [Morningstar](https://github.com/azure-ai-foundry/foundry-samples/tree/main/samples/microsoft/python/getting-started-agents/3p-tools/Morningstar) | Access up-to-date investment research and data such as analyst research, expert commentary, and essential Morningstar data. |
| [Trademo](https://github.com/azure-ai-foundry/foundry-samples/tree/main/samples/microsoft/python/getting-started-agents/3p-tools/Trademo_Global_trade) | Provide latest duties and past shipment data for trade between multiple countries |
| [Tripadvisor](https://github.com/azure-ai-foundry/foundry-samples/tree/main/samples/microsoft/python/getting-started-agents/3p-tools/Tripadvisor) | Get travel data, guidance and reviews |

**Best Practices**

**Use system instruction to help model invoke the right tool**

In order for the model to understand which tools to use, you want to provide detailed instruction for the model to describe when and how to use the tool. You might want to consider providing the following information:

* Primary Objective: what is the objective of this agent? what is the goal of related tasks? what are the expected outcomes?
* Your responsibilities: what tasks you expect the agent to perform. For example, calling Grounding with Bing Search tool to get the latest information about local events.
* Inputs you may receive: what inputs do you expect the agent to receive?
* For each tool:
  + The tool name
  + A description of the tool
  + Triggers: when do you expect this tool to be called? What type of information will be searched? What will queries contain?
  + An example of a query

For example, you might provide tool instructions like the following for the Grounding with Bing Search tool:

Grounding with Bing Search tool

* Use: Gather external trends or news to enrich the post with real-time insights.
* Trigger this when:
  + The user asks to reference recent data or competitive context.
  + Example: "Can you reference the latest industry trends?" or "What are competitors doing?".