Build knowledge-enhanced AI agents with Foundry IQ

1. Introduction

https://learn.microsoft.com/en-us/training/modules/introduction-foundry-iq/1-introduction

Introduction

Completed

- 2 minutes

Your organization is adopting AI agents to automate tasks and boost productivity. But you discover a fundamental challenge: how do you scale agents beyond simple, predefined tasks?

The real power of AI agents emerges when they can access your organization's knowledge. This knowledge includes your policies, procedures, product documentation, support articles, and domain expertise built over years.

Traditional AI agents have significant limitations. They can't access your private data, they're constrained by knowledge cutoff dates, and they generate generic responses without your company's context. When they lack factual grounding, they often create incorrect information.

If you want to build knowledge-enabled agents, you face complex engineering challenges. You need to connect to data sources, implement chunking strategies, build vector databases, and manage access controls. Each team tackles these same problems repeatedly.

Foundry IQ is Microsoft's unified knowledge platform that transforms how your AI agents access organizational data. Instead of rebuilding custom Retrieval Augmented Generation (RAG) pipelines for every project, you get a shared knowledge management system. Multiple agents can access the same knowledge bases, and improvements to those knowledge bases benefit every connected agent immediately.

Learning objectives

In this module, you learn to:

- Explain how RAG solves the knowledge problem by connecting agents to real-time information
- Describe how Foundry IQ provides a shared knowledge platform that multiple agents can access
- Configure data sources for knowledge bases including Azure AI Search, Blob Storage, SharePoint, and OneLake
- Configure agent instructions to control retrieval behavior and ensure consistent citations
- Test and monitor agent retrieval to maintain quality in production

Let's start by discovering how Retrieval Augmented Generation (RAG) transforms simple agents into powerful knowledge-enhanced assistants.

2. Understanding RAG for agents

https://learn.microsoft.com/en-us/training/modules/introduction-foundry-iq/2-understand-rag

Understanding RAG for agents

Completed

- 5 minutes

To understand why Foundry IQ represents such a significant advancement, we need to first examine the fundamental challenges that simple AI agents face in enterprise environments and how **Retrieval Augmented Generation** (**RAG**) addresses these limitations.

Simple AI agent limitations

Simple AI agents face significant challenges in enterprise environments. These limitations prevent them from providing the accurate, contextual responses that organizations need for critical business operations:

Limitation	Impact	Example
Knowledge cutoff dates	No access to recent information	Can't help with newly released features or updated policies
Private data access	Generic responses only	Missing company procedures, support knowledge, product specs
Lack of context	Irrelevant advice	Ignores specific security requirements or approval workflows
Fabricated responses	Compliance and security risks	Confident-sounding but incorrect information
Scalability issues	Duplicated engineering effort	Every team rebuilds the same RAG infrastructure

These challenges create real barriers to AI adoption in enterprise settings, where accuracy and reliability are non-negotiable.

How RAG solves these problems

Retrieval Augmented Generation (RAG) transforms agents by connecting them to organizational knowledge sources in real-time. This architectural approach fundamentally changes how agents access and use information, moving from static training data to dynamic knowledge retrieval.

The RAG process works in three coordinated steps:

1. **Retrieve**: System searches knowledge bases for relevant content related to the query
2. **Augment**: Combines retrieved content with the user's question to provide factual context
3. **Generate**: Agent creates response using both training data and retrieved information

Through this process, RAG delivers three critical advantages for enterprise AI:

- **Real-time updates** that keep agents current with policies and procedures without requiring retraining
- **Source transparency** that shows users exactly which documents informed each response to build trust and enable verification
- **Factual grounding** that anchors responses in actual organizational content to eliminate fabricated information and ensure compliance

While RAG solves the knowledge problem, building it requires significant technical expertise. This is where Microsoft Foundry IQ comes in. Foundry IQ provides a ready-made knowledge platform that eliminates the complexity of custom RAG implementations. Let's explore Foundry IQ in the next unit.

3. Explore Foundry IQ

https://learn.microsoft.com/en-us/training/modules/introduction-foundry-iq/3-foundry-iq

Explore Foundry IQ

Completed

- 8 minutes

You now know how RAG solves the knowledge problem for AI agents. But here's the challenge: building a RAG system from scratch means configuring vector databases, implementing embedding pipelines, tuning retrieval algorithms, and maintaining search infrastructure. What if you need three different AI agents across your organization? You'd build three separate RAG systems.

There's a better approach.

What is Foundry IQ?

Foundry IQ is a managed knowledge platform for AI agents built on Azure AI Search. It provides the retrieval capabilities you learned about in RAG, but as a shared service that multiple agents can use.

Consider a typical scenario. Your organization has product documentation stored in SharePoint, customer policies in Azure Blob Storage, and training materials in OneLake. With traditional RAG, you'd index each data source separately for each agent. With Foundry IQ, you create knowledge bases once and connect any agent to them.

This matters because it shifts your focus from building infrastructure to designing agent experiences. You spend time improving what information your agents access, not how they access it.

How knowledge bases organize information

Knowledge bases in Foundry IQ organize information by business domain rather than technical storage location. This design reflects how people actually think about information.

Instead of agents searching "SharePoint Site A" or "Blob Container B," they search "Product Documentation" or "HR Policies." Each knowledge base brings together related information regardless of where it's stored.

For example, your Product Documentation knowledge base might include:

- Technical specifications from SharePoint
- API documentation from Azure Blob Storage
- Usage analytics from OneLake
- Support tickets from your existing search index

To agents, this appears as one unified knowledge source. To you, it means connecting data sources to knowledge bases rather than managing separate retrieval systems.

Connecting data sources

Foundry IQ connects to your existing storage through data source integrations. You point it at your SharePoint sites, Blob containers, or OneLake instances. Foundry IQ handles indexing, embedding generation, and search optimization automatically.

Here's what happens when you add a data source:

1. **Discovery**: Foundry IQ scans your storage location for documents
2. **Processing**: Documents are chunked and embedded for semantic search
3. **Indexing**: Content becomes searchable through the knowledge base
4. **Monitoring**: Changes to your documents trigger automatic reindexing

You configure this once per data source. Every agent connected to that knowledge base instantly benefits from updates.

Built-in retrieval intelligence

Remember the retrieval strategies you learned about in RAG? Foundry IQ implements these automatically. When an agent queries a knowledge base, the platform:

Analyzes the question to understand what information the agent needs. A question like "What's our return policy for damaged items?" requires different retrieval than "List all return policies."

Selects retrieval strategies based on the query. Simple factual questions use keyword search. Complex questions combine semantic search with query expansion.

Ranks results using relevance scoring. The most contextually appropriate information surfaces first, reducing the tokens needed for agent responses.

Provides citations so agents can reference source documents. This builds trust and lets users verify information.

This intelligence runs without custom code. You define what knowledge bases contain. Foundry IQ determines how to retrieve from them.

Connecting agents to knowledge

Let's see how simple it is to give an agent access to organizational knowledge. This example creates a support agent that can answer questions using product documentation:

```
 from azure.ai.projects import AIProjectClient
from azure.ai.projects.models import PromptAgentDefinition, MCPTool

project_client = AIProjectClient(endpoint=project_endpoint, credential=credential)

# Connect to the product documentation knowledge base
knowledge_tool = MCPTool(
    server_label="product-docs",
    server_url=f"{search_endpoint}/knowledgebases/product-documentation/mcp"
)

# Create an agent with knowledge access
agent = project_client.agents.create_version(
    agent_name="product-support-agent",
    definition=PromptAgentDefinition(
        model="gpt-4o-mini",
        instructions="Answer product questions using the knowledge base. Always cite your sources.
        tools=[knowledge_tool]
    )
)
```

The agent now retrieves information from the knowledge base just like it would use any other tool. You don't write retrieval logic or manage search infrastructure.

The shared knowledge advantage

The real value of Foundry IQ emerges when you scale beyond one agent. Imagine your organization needs:

- A support agent answering customer questions
- An employee assistant helping with HR policies
- A developer agent explaining API usage

With traditional RAG, you'd build and maintain three retrieval systems. With Foundry IQ, you create knowledge bases that multiple agents share:

- The **Product Documentation** knowledge base serves both the support agent and developer agent
- The **HR Policies** knowledge base serves only the employee assistant
- Each agent accesses exactly the knowledge it needs

When you improve a knowledge base by adding data sources or refining content, every connected agent benefits immediately. This is how organizations build consistent, scalable AI agent systems.

> **Note**
>
> Foundry IQ uses the Model Context Protocol (MCP) to connect agents to knowledge bases. MCP provides a standardized way for AI agents to access external tools and data sources securely.

4. Configure data sources for knowledge bases

https://learn.microsoft.com/en-us/training/modules/introduction-foundry-iq/4-data-requirements

Configure data sources for knowledge bases

Completed

- 9 minutes

Your knowledge base is only as good as the data it contains. Foundry IQ lets you connect to multiple data sources, enabling your AI agent to access the information it needs to answer questions accurately. You configure these sources when you set up your knowledge base, ensuring your agent has the right context for your specific use case.

Understanding which data source to use depends on where your data lives and how you need to access it. Foundry IQ supports six primary data source types:

Data Source	Access Type	Best For
Azure AI Search Index	Indexed	Enterprise search with custom pipelines

Data Source	Access Type	Best For
Azure Blob Storage	Direct	Document files in Azure Storage
Web	Real-time	Current, public information via Bing
SharePoint (Remote)	Real-time	Live SharePoint content with Microsoft 365 governance
SharePoint (Indexed)	Indexed	Advanced search on SharePoint with custom pipelines
OneLake	Direct	Unstructured data in Microsoft Fabric

With real-time sources, you get current information. With internal data sources like SharePoint or OneLake, you maintain security and governance while giving your agent access to proprietary knowledge.

Azure AI Search Index

Azure AI Search Index provides enterprise-scale search capabilities for your Foundry IQ knowledge base. This option is ideal when you already invested in Azure AI Search and want to use existing search indexes.

With this source, you connect directly to your Azure AI Search index, which can contain data from multiple origins that you've already processed and indexed. This becomes especially important when you need sophisticated search capabilities like semantic ranking, filters, or custom scoring profiles that Azure AI Search provides.

> **Tip**
>
> Learn more about Azure AI Search and how to create and manage search indexes for your knowledge bases.

Your agent can query this index to retrieve relevant information based on user questions. Key benefits include:

- **Semantic ranking** - Finds contextually relevant results, not just keyword matches
- **Custom scoring** - Prioritizes results based on your business logic
- **Faceted navigation** - Filters results by categories or attributes
- **Multi-language support** - Handles content in different languages

Azure Blob Storage

Azure Blob Storage lets you retrieve documents and files directly from your blob containers. You select specific containers or blobs, and Foundry IQ processes the content to make it available to your agent.

This source works well when you store documents in Azure Blob Storage. Common file types include:

- PDF documents
- Microsoft Word files (.docx)
- Text files (.txt)
- Markdown files (.md)
- HTML files

> **Note**
>
> Unlike Azure AI Search, which requires you to build and maintain an index, Blob Storage provides a more direct path from your files to your knowledge base.

Building on this concept, you can organize your blobs into containers based on topics or access levels, making it easier to manage what information your agent can access. This organization helps maintain data governance while keeping your knowledge base current.

Web

Web access grounds your agent with real-time content from the internet via Bing. Instead of relying only on static, internal data, your agent can search for current information when answering questions.

This becomes especially important when users ask about:

- Recent events or news
- Current pricing or availability
- Frequently changing information
- Topics outside your internal knowledge base

Important

With web grounding, you're relying on Bing's search results, which means less control over the specific sources your agent references. When accuracy and source verification are critical, consider using indexed, controlled data sources instead.

> **Tip**

> You can combine web grounding with internal data sources, using web access as a supplementary source when internal knowledge doesn't provide an answer.

Microsoft SharePoint options

Foundry IQ provides two ways to connect to SharePoint, each with distinct advantages. The following table compares these approaches:

Feature	Remote	Indexed
Access method	Real-time queries	Preprocessed index
Response time	Depends on SharePoint	Faster
Maintenance	No index to maintain	Requires index updates
Advanced search	Limited	Full Azure AI Search capabilities
Data freshness	Always current	Depends on indexing schedule
Permission handling	Respects SharePoint permissions	Configured during indexing

SharePoint Remote

SharePoint Remote provides search capabilities with Microsoft 365 governance, retrieving content directly from SharePoint without preindexing. Your agent searches SharePoint sites and libraries in real-time when users ask questions.

Key benefits of remote access:

- No index maintenance required
- Always accesses current SharePoint content
- Automatically respects existing SharePoint permissions
- Simpler setup and configuration

> **Tip**
>
> Use SharePoint Remote when you need the simplest path to SharePoint data and don't require advanced search features.

SharePoint Indexed

SharePoint Indexed takes a different approach by indexing SharePoint content into Azure AI Search for custom pipelines. Unlike remote access, which queries SharePoint in real-time, indexing processes your SharePoint content in advance.

This preprocessing means faster response times and more sophisticated search capabilities. With indexed content, you can:

1. Apply custom analyzers for specialized terminology
2. Build enrichment pipelines with AI services
3. Combine SharePoint data with other sources
4. Create specialized search experiences

> **Note**
>
> Indexed SharePoint works best when you need advanced search features or when you're integrating SharePoint data with other sources in your Azure AI Search index.

Microsoft OneLake

Microsoft OneLake provides access to unstructured data stored in your Microsoft Fabric data lakehouse. You connect to OneLake to retrieve files and documents stored in your lakehouse, making this data available to your knowledge base.

> **Tip**
>
> Learn more about Microsoft Fabric OneLake and how it serves as a unified data lake for your organization.

This option matters when your organization uses Microsoft Fabric for data analytics and storage. Common use cases include:

- **Business intelligence reports** - Reference analytical findings in agent responses
- **Data documentation** - Provide context about datasets and metrics
- **Analytical findings** - Share insights from data science work
- **Research outputs** - Make research accessible through conversational AI

With this connection, your agent can reference this information when answering business questions, providing data-driven responses grounded in your organization's analytical work.

Choose the right data source

Selecting the appropriate data source depends on several factors. Use this decision guide:

If your data is...	And you need...	Choose...
In SharePoint	Simple setup, always current	SharePoint Remote
In SharePoint	Advanced search, custom pipelines	SharePoint Indexed
Files in Azure	Direct file access	Azure Blob Storage
In Microsoft Fabric	Data lakehouse content	OneLake
Already indexed	Existing Azure AI Search investment	Azure AI Search Index
Public, current information	Real-time web content	Web

Important

You can combine multiple sources in a single knowledge base. For example, use internal SharePoint data as the primary knowledge base while enabling web grounding for current events or supplementary information.

5. Configure retrieval with Foundry IQ

https://learn.microsoft.com/en-us/training/modules/introduction-foundry-iq/5-configure-retrieval

Configure retrieval with Foundry IQ

Completed

- 8 minutes

You've built a knowledge base and optimized how content gets indexed. Now comes the critical step: configuring how agents retrieve and use that knowledge.

This is where many implementations fail. You can have perfectly indexed content with excellent semantic ranking, but if your agent doesn't know when or how to use the knowledge base, users get

inconsistent results.

The retrieval behavior problem

Consider what happens without proper configuration. You ask your agent "What's our vacation policy?" Three different behaviors might occur:

Behavior	Example Response	Problem
Answers from training data	"Most companies offer 2-3 weeks of vacation annually"	Generic information, not your actual policy
Searches but doesn't cite	"You get 15 days of PTO annually"	Correct but unverifiable, no accountability
Searches, cites, and grounds	"You receive 15 days of paid time off annually 【doc_id:1†Employee Handbook 2024】"	✓ This is what you want

Only the third behavior is acceptable for enterprise agents. The first provides wrong information. The second lacks accountability. You need agents that consistently retrieve, cite, and stay grounded in your knowledge base.

Controlling retrieval with instructions

Agent instructions determine retrieval behavior. Think of them as the contract between you and the agent about how it should use knowledge bases.

Here's a basic approach that produces inconsistent results:

```
agent = project_client.agents.create_version(
    agent_name="hr-assistant",
    definition=PromptAgentDefinition(
        model="gpt-4o-mini",
        instructions="Answer HR questions using the knowledge base.",
        tools=[knowledge_tool]
    )
)
```

This instruction is too vague. "Using the knowledge base" doesn't specify *when* to use it or *how* to present results. The agent might search or might not. It might cite sources or might not.

Writing effective retrieval instructions

Effective instructions specify three critical behaviors:

1. **When to retrieve**: Tell the agent to always use the knowledge base, never rely on training data
2. **How to cite**: Specify the exact format for source attribution
3. **What to do when unsure**: Define fallback behavior when information isn't found

Here's how this looks in practice:

```
 retrieval_instructions = """You are a helpful HR assistant.

CRITICAL RULES:
- You must ALWAYS search the knowledge base before answering any question
- You must NEVER answer from your own knowledge or training data
- Every answer must include citations in this format: 【doc_id:search_id†source_name】
- If the knowledge base doesn't contain the answer, respond with "I don't have that information in

Your role is to provide accurate, verifiable information from company documentation."""

agent = project_client.agents.create_version(
    agent_name="hr-assistant",
    definition=PromptAgentDefinition(
        model="gpt-4o-mini",
        instructions=retrieval_instructions,
        tools=[knowledge_tool]
    )
)
```

These instructions create consistent behavior. The agent knows exactly when to search, how to format responses, and what to do when information isn't available.

Testing retrieval behavior

Instructions alone aren't enough. You need to verify that agents actually behave as configured. This requires systematic testing with different query types.

Setting up a test conversation

Create a conversation session and send test queries:

```
 openai_client = project_client.get_openai_client()
conversation = openai_client.conversations.create()

# Test query that should trigger retrieval
response = openai_client.responses.create(
    conversation=conversation.id,
    input="How many vacation days do I get?",
```

```
    extra_body={"agent": {"name": agent.name, "type": "agent_reference"}}
)

print(response.output_text)
```

What to test

Your test queries should cover different retrieval scenarios:

Query Type	Example Questions	Expected Behavior
Straightforward factual	"What is our vacation policy?" "Where can I find the security guidelines?"	Direct retrieval with citations
Questions requiring synthesis	"What are the differences between our leave types?" "How do I request time off?"	Multiple document retrieval, synthesized answer with multiple citations
Questions outside knowledge base	"What's the weather like today?" "Tell me about machine learning"	Graceful fallback ("I don't have that information...")
Ambiguous questions	"What about benefits?" "Tell me more about that"	Clarifying questions or focused search on most relevant topic

Evaluating response quality

Good responses demonstrate four characteristics:

1. **Grounding** - Information comes from knowledge base, not training data
2. **Citation** - Every factual claim includes source references
3. **Relevance** - Retrieved content actually answers the question
4. **Completeness** - All necessary information is provided, not just fragments

When you find responses that don't meet these criteria, adjust your instructions. Add more specific rules. Clarify edge cases. Iterate until behavior becomes consistent.

Retrieval strategies for different agent types

Different agent purposes require different retrieval approaches. A customer support agent needs different behavior than an internal research assistant.

Customer-facing support agents

These agents need high accuracy and must never provide uncertain information:

```
 support_instructions = """You provide customer support using our product documentation.

Rules:
- Search the knowledge base for every product question
- Cite documentation for all technical answers
- If documentation doesn't cover a question, say "Let me connect you with a specialist" rather tha
- Focus on official product information, not general knowledge"""
```

Internal research assistants

These agents can synthesize across documents and provide broader context:

```
 research_instructions = """You help employees research topics across company documentation.

Rules:
- Search all relevant knowledge bases for comprehensive answers
- Synthesize information from multiple sources when helpful
- Always cite all sources used
- Indicate confidence level when synthesizing across documents
- Suggest related topics that might be useful"""
```

Specialized domain experts

These agents focus deeply on specific knowledge domains:

```
 compliance_instructions = """You are a compliance documentation assistant.

Rules:
- Only answer questions about compliance policies and procedures
- Always cite the specific policy document and section
- If a question involves interpretation or legal advice, refer to the compliance team
- Keep answers strictly factual based on written policies
- Note the effective date of any policy you reference"""
```

The pattern is consistent: define the agent's scope, specify retrieval requirements, establish citation standards, and handle edge cases explicitly.

Moving from testing to production

Once your test queries produce consistent, high-quality results, you're ready to deploy. But production introduces new challenges.

Monitor actual usage patterns. Users ask questions differently than your test scenarios. Some questions hit edge cases you didn't anticipate. Others reveal gaps in your knowledge base content.

Track these patterns:

- **Citation frequency** - Are agents consistently citing sources?
- **Fallback frequency** - How often do agents say "I don't know"?
- **Query types** - What categories of questions appear most often?
- **Retrieval accuracy** - Do retrieved documents actually contain answers?

Use this data to refine instructions, improve knowledge base content, and adjust search configurations. Retrieval quality improves through iteration based on real-world usage.

The combination of clear instructions, systematic testing, and ongoing monitoring creates reliable knowledge retrieval that scales across your organization's agents.

6. Knowledge check

https://learn.microsoft.com/en-us/training/modules/introduction-foundry-iq/6-knowledge-check

Knowledge check

Completed

- 3 minutes

7. Exercise - Integrate an AI agent with Foundry IQ

https://learn.microsoft.com/en-us/training/modules/introduction-foundry-iq/7-exercise

Exercise - Integrate an AI agent with Foundry IQ

Completed

- 35 minutes

If you have an Azure subscription, you can explore Foundry IQ in Microsoft Foundry for yourself.

> **Note**
>
> If you don't have an Azure subscription, and you want to explore Microsoft Foundry, you can sign up for an account, which includes credits for the first 30 days.

Launch the exercise and follow the instructions.

Launch Exercise

8. Summary

https://learn.microsoft.com/en-us/training/modules/introduction-foundry-iq/8-summary

Summary

Completed

- 1 minute

You've learned how Foundry IQ transforms AI agents from simple chatbots into knowledge-enhanced enterprise tools capable of accessing your organization's information.

RAG solves the knowledge problem

Retrieval Augmented Generation addresses the fundamental limitations of AI agents by connecting them to real-time knowledge sources. Instead of relying only on training data, RAG-enabled agents retrieve relevant information, augment queries with factual context, and generate responses grounded in your organizational content. This provides real-time updates, source transparency, and factual grounding.

Foundry IQ provides a shared knowledge platform

Foundry IQ eliminates the need to build custom RAG infrastructure for every agent. You create knowledge bases organized by business domain, connect data sources from SharePoint, Azure Blob Storage, OneLake, or existing Azure AI Search indexes, and any agent can access them. When you improve a knowledge base, every connected agent benefits immediately. This shared approach scales knowledge access across your organization.

Data quality determines retrieval effectiveness

You improve retrieval through three key techniques:

- **Scoring profiles** boost specific fields or attributes to surface more relevant results
- **Semantic ranking** uses AI models to understand meaning and context beyond keywords
- **Custom analyzers** handle specialized content like HTML, product codes, or technical terminology

These techniques work together to transform basic search into intelligent retrieval tailored to your content.

Instructions control agent behavior

Effective agent instructions specify when to retrieve (always use the knowledge base), how to cite (exact format for source attribution), and what to do when unsure (graceful fallback). Test different query types to verify consistent behavior. Monitor production usage to identify patterns and refine your configuration.

Next steps

Start with a high-value knowledge domain where accurate, cited information provides immediate business impact. Build your first knowledge base, configure data quality settings, create an agent with clear retrieval instructions, and test systematically before deploying to users.