



Vendor: Microsoft

Exam Code: AI-103

Exam Name: Azure AI App and Agent Developer Associate

Version: DEMO

QUESTION 1

You have a Microsoft Foundry project that contains a model deployment.

You have an application that calls the deployment by using the Azure OpenAI v1 API and DefaultAzureCredential.

The developers at your company receive HTTP 403 errors when they send inference requests, even after running az login.

You need to ensure that the developers can perform model inference. The solution must follow the principle of least privilege.

Which role-based access control (RBAC) role should you assign to the developers?

- A. Cognitive Services User
- B. Cognitive Services OpenAI User
- C. Contributor
- D. Cognitive Services Data Reader

Answer: B

Explanation:

To resolve the HTTP 403 Forbidden errors when making inference calls via the Azure OpenAI v1 API and DefaultAzureCredential, users must be assigned the Cognitive Services OpenAI User built-in Azure RBAC role.

Why This Happens

Running az login successfully authenticates the user with Microsoft Entra ID, but it does not grant data-plane access permissions. By default, standard control-plane roles (like Reader or Foundry User) only allow users to view project metadata or manage settings, not send prompts to the model deployment endpoint itself.

Recommended Role Definition

Role Name: Cognitive Services OpenAI User

Permissions Granted: This role provides the absolute minimum privileges required to execute chat completions, embeddings, and general inference tasks (Microsoft.CognitiveServices/accounts/OpenAI/deployments/search/action and Microsoft.CognitiveServices/accounts/OpenAI/deployments/causalLanguageModeling/action). It does not allow users to deploy new models, view access keys, or alter configurations

Scope Placement: Assign this role to the users (or a Microsoft Entra ID Group) at the Azure OpenAI resource level or the Resource Group level containing your Microsoft Foundry infrastructure.

Reference:

<https://learn.microsoft.com/en-us/azure/foundry-classic/openai/how-to/managed-identity>

QUESTION 2

You have a Microsoft Foundry project that contains an agent. The agent has a Model Context Protocol (MCP) tool that queries a knowledge base stored in Azure AI Search.

Some agent runs return answers from the base model without invoking the knowledge base, which results in responses without grounded citations.

You are provided with the following code snippet that runs the agent.

```
run = project_client.agents.runs.create_and_process(  
    thread_id=thread.id,  
    agent_id=agent.id  
)
```

You need to add the correct `tool_choice` parameter to the code to deterministically force the agent to invoke the MCP tool on each run.

What should you add?

- A. `tool_choice={"required"}`
- B. `tool_choice={"auto"}`
- C. `tool_choice={"type":"knowledge_base"}`
- D. `tool_choice={"type":"mcp"}`

Answer: A

Explanation:

To deterministically force the agent to invoke your Model Context Protocol (MCP) tool on every run, you must pass `tool_choice="required"` into the `run_create_and_process` method.

The 'required' tool choice: Setting this parameter to 'required' forces the underlying Azure OpenAI model to invoke one of your available tools on every response, ensuring the agent doesn't guess answers from the base model.

Reference:

<https://learn.microsoft.com/en-us/azure/foundry/agents/concepts/tool-best-practice>

QUESTION 3

Drag and Drop Question

You have a Microsoft Foundry project that contains a customer support agent grounded in internal documentation.

After a recent update, users report the following issues:

- Some answers are unsupported by retrieved documents.
- A small number of responses are flagged for policy violations.

You need to evaluate each issue.

Which observability signals should you use for each issue? To answer, drag the appropriate observability signals to the correct issues. Each observability signal may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.

NOTE: Each correct selection is worth one point.

Observability signals	Answer Area
☐ Groundedness evaluation metrics	Unsupported responses: ☐ Observability signal
☐ Latency breakdown traces	Policy violations: ☐ Observability signal
☐ Risk and safety metrics	
☐ Token usage analytics	

Answer:

Observability signals	Answer Area
☐ Groundedness evaluation metrics	Unsupported responses: ☐ Groundedness evaluation metrics
☐ Latency breakdown traces	Policy violations: ☐ Risk and safety metrics
☐ Risk and safety metrics	
☐ Token usage analytics	

QUESTION 4

You have a Microsoft Foundry project named Project1 that contains an agent. The agent uses an OpenAPI 3.0 specification to call an external weather service.

The weather service requires a key to be passed in an HTTP header. The key value is stored as a connection in Project1.

You need to ensure that the key value from the connection is included automatically whenever the OpenAPI tool is invoked.

What should you configure in the OpenAPI specification?

- A. a header parameter defined for each operation
- B. an Azure Key Vault connection
- C. an API key security scheme
- D. a Bearer token security scheme

Answer: C

Explanation:

To ensure Microsoft Foundry automatically injects the API key from your project connection whenever the OpenAPI tool is invoked, your OpenAPI 3.0 specification must explicitly include a securitySchemes component mapping to the exact header name, and a global or operation-level security requirement referencing that scheme. The orchestrator matches the name field in the specification against the key stored inside your project's custom connection.

1. Required OpenAPI 3.0 Configuration

You must add both the components.securitySchemes block and the security block to your specification file:

openapi: 3.0.0

```
info:
  title: External Weather Service
  version: 1.0.0
paths:
  /weather:
    get:
      operationId: getWeather
      responses:
        '200':
          description: Successful weather retrieval
```

1. Define the security scheme in the components section

components:

```
*-> securitySchemes:
  weatherApiKey:          # Arbitrary logical identifier for this scheme
*->   type: apiKey
      in: header
      name: X-Weather-API-Key # MUST match the "key" name configured in your Foundry
      Connection
```

2. Apply the security requirement globally (or inside individual operations)

```
security:
  - weatherApiKey: []      # Instructs Foundry to enforce this scheme on the API requests
```

in: header: Explicitly instructs the Foundry proxy layer to attach the credential value to the HTTP request headers (rather than as a query parameter).name: This string value is the exact HTTP header key (e.g., X-Weather-API-Key or Authorization). Crucially, this value must identically match the "Key" property given to the secret in your Microsoft Foundry Custom Connection.

security: Actively triggers the authentication workflow for the tool's endpoints. Without this block, Microsoft Foundry treats the API call as anonymous and strips out connection values.

Reference:

<https://learn.microsoft.com/en-us/azure/foundry/agents/how-to/tools/openapi>

QUESTION 5

You have a Microsoft Foundry project that serves a high-volume chat app.

Most requests are simple FAQs, but some require advanced reasoning.

You need to reduce costs and latency for common queries, without degrading the quality of the responses to complex questions.

What should you do?

- A. Route all the requests to a smaller model.
- B. Use a model cascade that routes the requests to different models.
- C. Increase the value of the max_tokens parameter for all the requests.
- D. Route all the requests to the most capable model.

Answer: B

Explanation:

One should absolutely use a model cascade to route requests to different models based on complexity. This architectural pattern is highly effective for high-volume chat applications because

it directly addresses the trade-off between operational cost, API latency, and response quality.

Using a model cascade router ensures that your high-volume Microsoft Foundry application scales efficiently by reserving expensive computational power exclusively for queries that actually require advanced cognitive processing.

Note:

1. Analyze Request Complexity

Implement a lightweight intent classifier or routing layer at the entry point of your Microsoft Foundry project. This router quickly inspects incoming user prompts using basic heuristic keyword matching, semantic embeddings, or a highly optimized, fast model (like Phi-3 or GPT-4o-mini) to categorize the query as either a "Simple FAQ" or a "Complex Reasoning" request.

2. Route to the Optimal Tier Tier 1 (Fast & Cheap):

Route standard, predictable FAQ requests to a smaller, cost-effective model or a local cache/vector database lookup. This keeps latency in milliseconds and drastically lowers token costs.

Tier 2 (Advanced Reasoning): Route multi-step logic, coding, or highly contextual queries to a frontier model (like GPT-4o).

3. Implement Fallback Logic Design the cascade to be dynamic. If the smaller Tier 1 model generates a response with low confidence, or if the user asks a follow-up question that invalidates the simple FAQ status, seamlessly upgrade the conversation loop to the Tier 2 model.

Reference:

<https://medium.com/@sujathamudadla1213/what-is-the-primary-purpose-of-a-model-cascade-in-machine-learning-0b145a7bc6e2>

QUESTION 6

You have a Microsoft Foundry project that contains an agent.

The agent uses Azure AI Search as the retriever.

You plan to ingest PDF into an Azure AI Search index to ensure that the agent can ground responses in texts in both documents and embedded images.

Users require citations that link to the source files.

You need to ensure that during indexing, the images are extracted into a structure that can be used as input for the built-in optical character recognition (OCR) skill.

Which indexing approach should you use?

- A. an indexer to extract image data into a normalized_images collection
- B. a Shaper skill to restructure the OCR input
- C. a skillset to run the OCR skill directly against the content field of the index
- D. the outputFieldMappings parameter to write image data to a searchable field

Answer: A

Explanation:

The best configuration step is an indexer to extract image data into a normalized_images collection.

Cracking Foundation: In Azure AI Search, document cracking automatically separates textual

data from visual content. To make embedded images available to image-processing skills like OCR, the indexer must be explicitly configured with an `imageAction` configuration property (such as `generateNormalizedImages` or `generateNormalizedImagePerPage`).

Input Structure for OCR: This indexer step extracts embedded images from the PDF and restructures them into an internal `normalized_images` array (accessed via the path `/document/normalized_images/*`). The built-in OCR skill strictly requires this normalized image collection format as its input payload.

Incorrect:

[Not C]

Content Field Limitations: The `/document/content` field generated during document cracking holds only the raw text extracted from the file. It does not contain the binary data or structural properties of embedded images.

Targeting Errors: Pointing an OCR skill directly at the raw text content field would fail to analyze the actual images, leaving the agent unable to ground answers in visual elements like diagrams or embedded infographics.

Reference:

<https://docs.azure.cn/en-us/search/tutorial-skillset>

QUESTION 7

You have a Microsoft Foundry project that contains an agent.

The knowledge source for the agent is a set of scanned PDF troubleshooting guides stored in Azure Blob Storage. The guide pages contain two-column layouts and tables.

You use Azure Content Understanding in Foundry Tools to process the PDFs.

You plan to ingest the processed content into an index for Retrieval Augmented Generation (RAG) and store extracted fields for downstream automation.

Stakeholders must be able to verify where each extracted field value came from in the original PDF and route low-reliability extractions for manual review.

You need to ensure that the Content Understanding document analyzer output includes a per-field confidence score and source grounding to locations within the source document.

What should you do?

- A. Set `enableSegment` to true.
- B. Provide labeled samples.
- C. Enable `estimateFieldSourceAndConfidence`.
- D. Configure the analyzer to use generative extraction for all fields.

Answer: C

Explanation:

To fulfill all your requirements using Azure Content Understanding in Foundry Tools, you need to configure a custom document analyzer with specific flags, set up an index ingestion pipeline, and build a downstream human-in-the-loop validation rule.

*-> 1. Enable Confidence Scores and Source Grounding

To force the analyzer to provide per-field confidence metrics and precise layout/bounding box

coordinates for verification, you must opt-in to the estimate FieldSourceAndConfidence parameter within your configuration.

Option A (Global): Set estimateFieldSourceAndConfidence = true in the main analyzer config to evaluate all fields.

Option B (Field-Level): Set estimateSourceAndConfidence = true under individual field schemas. This ensures the generated JSON response populates the bounding box coordinates, page numbers, and a confidence score 0.0 to 1.0 for every extracted entity.

2. Configure Document Extraction for Two-Column & Table Layouts

3. Build the Ingestion Pipeline (RAG vs. Automation Dual-Path)

4. Implement Threshold Routing and Source Verification

Reference:

<https://learn.microsoft.com/en-us/azure/ai-services/content-understanding/document/overview>

QUESTION 8

You are building a speech processing solution in Microsoft Foundry for a customer support platform.

The platform will transcribe live phone calls, so that supervisors at your company can view call transcripts and detect issues while the calls are in progress. The call audio will arrive as a continuous stream from the telephony system.

You need to ensure that the call transcripts appear within only a few seconds of the audio stream.

What should you do?

- A. Use text to speech by using a custom neural voice.
- B. Use speech translation to generate the transcripts into multiple languages.
- C. Run a batch transcription job on recorded audio files.
- D. Use real-time speech to text to process streaming audio input.

Answer: D

Explanation:

You should use Real-Time Speech-to-Text (Streaming STT) for this project.

Why Real-Time STT is Required

Low Latency: It transcribes audio within seconds.

Continuous Input: It processes live, ongoing audio streams.

Live Monitoring: Supervisors need immediate text to flag issues.

Reference:

<https://learn.microsoft.com/en-us/azure/ai-services/speech-service/get-started-speech-to-text>

QUESTION 9

You are creating an agent workflow in a Microsoft Foundry project to support natural voice interactions.

The agent must receive continuous audio input, convert the input into text for reasoning, and then return spoken responses to a user. The workflow must meet the following requirements:

- Support turn-taking dynamics, where the agent begins to generate the speech output before the user finishes speaking.
- Operate with low latency to maintain conversational experience.

You need to enable both speech to text and text to speech in a real-time agent interaction.

What should you do?

- A. Use batch transcription to convert the audio input and return text responses from the agent.
- B. Use real-time speech to text for incoming audio and text to speech for agent responses.
- C. Use an embeddings model to encode the audio, and then decode the audio into text and speech.
- D. Use speech translation to convert the audio into another language and return the translated text.

Answer: B

Explanation:

To achieve low latency and natural turn-taking dynamics in this specific Microsoft Foundry workflow, the best approach is to use real-time speech-to-text for incoming audio and text-to-speech for agent responses.

Low Latency: Streaming, real-time Speech-to-Text (STT) and Text-to-Speech (TTS) pipelines process audio chunks concurrently. This allows the system to analyze text and prepare responses while the user is still speaking.

Turn-Taking Support: Real-time STT systems utilize voice activity detection (VAD) and continuous streaming. This enables the agent to instantly detect pauses, interruptions, or trailing speech to naturally shift conversational turns.

Direct Reasoning Compatibility: Because your workflow requires converting input into text for reasoning (such as prompting a Large Language Model), a highly optimized text-based pipeline fits seamlessly without extra translation layers.

Reference:

<https://learn.microsoft.com/en-us/azure/foundry/openai/concepts/priority-processing>

Thank You for Trying Our Product

Lead2pass Certification Exam Features:

- ★ More than **99,900** Satisfied Customers Worldwide.
- ★ Average **99.9%** Success Rate.
- ★ **Free Update** to match latest and real exam scenarios.
- ★ **Instant Download** Access! No Setup required.
- ★ Questions & Answers are downloadable in **PDF** format and **VCE** test engine format.
- ★ Multi-Platform capabilities - **Windows, Laptop, Mac, Android, iPhone, iPod, iPad**.
- ★ **100%** Guaranteed Success or **100%** Money Back Guarantee.
- ★ **Fast**, helpful support **24x7**.



View list of all certification exams: <http://www.lead2pass.com/all-products.html>



10% Discount Coupon Code: ASTR14