

Microsoft.AI-103.v2026-09-21.q29

|                                                                                                                                                             |                                        |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------|
| Exam Code:                                                                                                                                                  | AI-103                                 |
| Exam Name:                                                                                                                                                  | Developing AI Apps and Agents on Azure |
| Certification Provider:                                                                                                                                     | Microsoft                              |
| Free Question Number:                                                                                                                                       | 29                                     |
| Version:                                                                                                                                                    | v2026-09-21                            |
| # of views:                                                                                                                                                 | 102                                    |
| # of Questions views:                                                                                                                                       | 290                                    |
| <a href="https://www.freecram.net/torrent/Microsoft.AI-103.v2026-09-21.q29.html">https://www.freecram.net/torrent/Microsoft.AI-103.v2026-09-21.q29.html</a> |                                        |

NEW QUESTION: 1

You have a Microsoft Foundry project that contains a deployed chat model.

You have a Python service that sends API requests to the model. The service is integrated with an automated validation system that compares generated outputs against approved response patterns.

Stakeholders report that small wording differences are causing validation mismatches.

You need to update the request parameters to improve output stability. The solution must maximize reasoning quality.

How should you complete the Python code? To answer, select the appropriate options in the answer area.

NOTE: Each correct selection is worth one point.

```
message = client.messages.create(  
    model="deployment-name",  
    messages=[  
        {"role": "user", "content": "Summarize the release notes in 3 bullet points."}  
    ],  
    max_tokens=800,  
    temperature=  ,  
    thinking={"type": "enabled"},  
    output_config={"effort":  }  
)
```



Answer:

```

message = client.messages.create(
    model="deployment-name",
    messages=[
        {"role": "user", "content": "Summarize the release notes in 3 bullet points."}
    ],
    max_tokens=800,
    temperature=0,
    thinking={"type": "enabled"},
    output_config={"effort": "high"}
)

```

Explanation:

temperature = 0

output\_config = { " effort " : " high " }

The correct configuration is temperature = 0 and output\_config = { " effort " : " high " }. The requirement is to reduce small wording variations that are breaking automated validation. In chat completion requests, temperature controls sampling randomness. Microsoft's Azure OpenAI reference states that temperature ranges between 0 and 2, and that higher values make output more random while lower values make output more focused and deterministic. Therefore, the most stable setting from the available choices is 0, because it minimizes randomness and improves repeatability for validation-sensitive response patterns.

The solution must also maximize reasoning quality. The code already enables thinking with thinking={ " type

" : " enabled " }, so the remaining reasoning-quality control is the effort setting. Microsoft Foundry model guidance states that the effort parameter controls the quality/cost tradeoff and supports low, medium, and high effort levels. Selecting " high " maximizes reasoning quality among the available options.

Using temperature values of 1 or 2 would increase variability and make validation mismatches more likely.

Selecting low or medium effort would not meet the requirement to maximize reasoning quality. Reference topics: Microsoft Foundry model inference, chat model parameters, temperature, thinking, effort, and output stability.

## NEW QUESTION: 2

You are building a web app named App1 that generates responses by using a model deployed to a Microsoft Foundry project named Project1.

Before sending the prompts to the model, App1 must retrieve documents by using Azure AI Search.

You need to integrate Project1 and App1. The solution must meet the following requirements:

- \* Multiple client applications must use the same search configuration.
- \* A security policy must prevent key-based authentication.
- \* Administrative effort must be minimized.

What should you do?

**A.** Enable a managed identity for each application and call Azure AI Search directly.

- B.** Create a custom HTTP connection in Foundry and manually configure Azure AI Search endpoints per application.
- C.** Call Azure AI Search directly from each application by using Microsoft Entra authentication.
- D.** Configure an Azure AI Search connection in Project1 and reference the connection in each application.

**Answer:** ([SHOW ANSWER](#))

The correct solution is to configure an Azure AI Search connection in Project1 and reference that connection from each application. Microsoft Foundry project connections are intended to centralize external resource configuration for a project. The official Foundry guidance states that when adding a connection, you select the external service, such as Azure AI Search , and choose the authentication method for that resource.

This directly satisfies the requirement for multiple client applications to use the same search configuration instead of duplicating endpoints, indexes, and credentials in each app.

The security requirement is met by configuring the connection with keyless authentication. The Azure AI Search integration guidance specifies that a Foundry project connection requires the search endpoint and either key-based authentication or keyless authentication with Microsoft Entra ID . For keyless authentication, RBAC roles are assigned to the project's managed identity, eliminating hard-coded API keys.

Options A and C use secure identity-based access but force every client application to maintain its own Azure AI Search configuration, increasing administration and violating the shared-configuration requirement. Option B also increases administrative work and does not use the native Azure AI Search connection type. Reference topics: Microsoft Foundry project connections, Azure AI Search grounding, Microsoft Entra ID authentication, managed identities, and RBAC.

### NEW QUESTION: 3

You have a Microsoft Foundry project that generates product marketing images from text prompts.

After publishing several images, the legal team at your company identifies a competitor ' s logo on a sign in the background of an image.

You need to remove only the logo, while preserving the rest of the image.

What should you do?

- A.** Increase the prompt guidance strength.
- B.** Modify the original prompt to exclude brand names.
- C.** Apply a mask-based inpainting edit to the part of the image that contains the logo.
- D.** Rerun the prompt by using a different random seed.

**Answer:** ([SHOW ANSWER](#))

The correct answer is C because the requirement is a localized image edit: remove only the competitor logo while preserving the rest of the already generated image. Azure OpenAI image editing is designed for modifying existing images based on a text instruction, rather than regenerating the entire image from scratch.

Microsoft's Azure OpenAI image guidance states that the Image Edit API modifies existing images and requires an input image as part of the request. In a mask-based inpainting workflow, the mask identifies the exact region to change, allowing the model to replace only the logo area while retaining surrounding background, composition, lighting, and product content.

Increasing prompt guidance strength would affect adherence during generation, but it would not safely remove a specific logo from a completed image. Modifying the original prompt and regenerating may create a different image and does not guarantee preservation of the approved visual content. Rerunning with a different random seed also changes the image unpredictably and may introduce new brand or legal issues.

Mask-based inpainting is the minimal-change remediation method for post-generation brand cleanup.

Reference topics: Azure OpenAI image editing, inpainting, mask-guided edits, image generation governance, and computer vision solutions.

### NEW QUESTION: 4

Note: This section contains one or more sets of questions with the same scenario and problem. Each question presents a unique solution to the problem. You must determine whether the solution meets the stated goals. More than one solution in the set might solve the problem. It is also possible that none of the solutions in the set solve the problem.

After you answer a question in this section, you will NOT be able to return. As a result, these questions do not appear on the Review Screen.

You have a multimodal AI generative model that accepts image uploads and uses extracted image text to generate responses.

You discover that users can upload unsafe images and embed hidden instructions into images to manipulate the model.

You need to implement controls to mitigate the risk.

Solution: You configure image moderation to block unsafe content before processing the images.

Does this meet the goal?

**A.** Yes

**B.** No

**Answer:** ([SHOW ANSWER](#))

The solution does not fully meet the goal. Image moderation is appropriate for one part of the risk: blocking unsafe image content before the image is processed. Azure AI Content Safety provides image APIs that detect harmful content, and its harm categories and severity levels can be used to classify and block objectionable image content. This addresses unsafe photos, but it does not address hidden instructions embedded in images.

The second risk is prompt manipulation through extracted image text. After OCR extracts text from the uploaded image, that text becomes untrusted third-party content supplied to a generative model. Microsoft defines document attacks as malicious instructions embedded in third-party content, where the objective is to cause the model to execute unintended commands or alter intended behavior. Prompt Shields are the control designed to detect user prompt attacks and document attacks, including indirect attacks that come from uploaded or referenced content. Therefore, image moderation alone is incomplete. A complete mitigation would combine image moderation for harmful visual content with Prompt Shields for document attacks, and optionally Spotlighting, so extracted or embedded text is treated as lower trust. Reference topics: Azure AI Content Safety, image moderation, Prompt Shields, document attacks, indirect prompt injection, and multimodal safety.

#### **NEW QUESTION: 5**

You have an app named App1 that uses a Microsoft Foundry multimodal model deployment.

App1 runs optical character recognition (OCR) on uploaded images and appends the OCR output to the prompt as additional context.

Some uploaded images contain embedded text.

You need to prevent potentially malicious instructions from being processed by the model.

What should you use?

**A.** protected material text

**B.** prompt shields for user prompts

**C.** image moderation

**D.** prompt shields for documents

**Answer:** **D** ([LEAVE A REPLY](#))

The correct answer is D. prompt shields for documents . The OCR text is extracted from uploaded images and appended as additional context, so it is third-party content rather than a direct trusted user instruction.

Microsoft Foundry Prompt Shields distinguish between user prompt attacks , which are malicious instructions directly supplied as user prompts, and document attacks , which are hidden or embedded instructions in third-party content such as documents, emails, webpages, or grounded data. Microsoft also notes that Prompt Shields analyze indirect attacks embedded in input documents or images.

This scenario is an indirect prompt injection pattern: an uploaded screenshot may contain text that says to ignore the system prompt or perform unauthorized actions. Because the OCR output is being passed to the multimodal model as contextual content, the appropriate protection is Prompt Shields for documents, which scans externally sourced context for malicious embedded instructions before generation. Protected material text detects known copyrighted text in model outputs, not prompt injection. Image moderation detects harmful visual content categories, not malicious instructions in extracted context. Prompt Shields for user prompts are less precise here because the risk comes from document-like content derived from the uploaded image.

Reference topics: Prompt Shields, document attacks, indirect prompt injection, multimodal safety, OCR- derived context, and Foundry guardrails.

#### **NEW QUESTION: 6**

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

Answer: (SHOW ANSWER)

The correct configuration is an API key security scheme . For Microsoft Foundry Agent Service OpenAPI tools, the OpenAPI specification must declare authentication through the components.securitySchemes section and use a scheme of type apiKey when the external service expects a key in a header. Microsoft's OpenAPI tool guidance states that API key authentication requires updating the OpenAPI spec security schemes with one scheme of type apiKey, and the tool then uses the associated project connection to supply the key value at runtime. This allows the key stored in Project1's connection to be injected automatically when the tool is invoked.

A header parameter defined separately for each operation is not the correct approach because credentials should not be modeled as ordinary operation parameters. The Foundry guidance explicitly indicates that parameters requiring the API key should be removed from the OpenAPI spec because the API key is stored and passed through a connection. A Bearer token security scheme is used for bearer-token-style authorization, not a generic weather API key passed in a custom HTTP header. Azure Key Vault is a secret store, but the scenario already stores the key in a Foundry project connection. Reference topics: Microsoft Foundry Agent Service, OpenAPI tools, project connections, API key authentication, and OpenAPI security schemes.

NEW QUESTION: 7

You have a Microsoft Foundry project that contains a workflow for a customer support triage process.

You have an Ask a question node that stores user responses in a local variable named Var01.

You need to create the following Power Fx expressions:

- \* An if/else condition expression that ensures that Var01 contains a value
  - \* A Send message expression that returns the stored user response in uppercase
- How should you configure the expressions? To answer, select the appropriate options in the answer area.

NOTE: Each correct selection is worth one point.

The screenshot shows a configuration interface for Power Fx expressions. It has two main sections: 'If/else condition expression:' and 'Send message expression:'. Each section has a dropdown menu. The 'If/else condition expression' dropdown is open, showing three options: 'IsBlank(Local.Var01)', 'IsEmpty(Local.Var01)', and 'Not(IsBlank(Local.Var01))'. The 'Send message expression' dropdown is also open, showing three options: '{Local.Var01}', '{Upper(Local.Var01)}', and '{Upper(Var01)}'. A 'Microsoft' logo is visible at the bottom of the interface.

Answer:



Explanation:

If/else condition expression: `Not(IsBlank(Local.Var01))`

Send message expression: `{Upper(Local.Var01)}`

The correct if/else condition is `Not(IsBlank(Local.Var01))` because the workflow must continue only when the Ask a question node has captured a nonblank value. In Power Fx, `IsBlank` checks whether a value is blank, while `Not` reverses that Boolean result; therefore, `Not(IsBlank(Local.Var01))` evaluates to true only when `Var01` contains user input. `IsEmpty(Local.Var01)` is not the correct choice because `IsEmpty` checks whether a table contains records, not whether a text variable has a value. The Power Fx formula reference distinguishes these functions by defining `IsBlank` for blank values and `IsEmpty` for empty tables.

The Send message expression must be `{Upper(Local.Var01)}`. Microsoft Foundry workflow guidance uses the same pattern for an Ask a question node saved as `Var01`, followed by a Send message action that outputs

`{Upper(Local.Var01)}`. The `Local.` prefix is required because the response is stored in a local workflow variable, and `Upper()` converts the stored text response to uppercase. Reference topics: Microsoft Foundry workflows, Power Fx expressions, local variables, if/else branching, and Send message actions.

### NEW QUESTION: 8

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 locations within the source document.

What should you do?

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

Answer: ([SHOW ANSWER](#))

The correct answer is A. Enable estimateFieldSourceAndConfidence . Azure Content Understanding document analyzers support an opt-in confidence and grounding capability for field extraction. Microsoft documentation states that to opt in for confidence and grounding, you set estimateFieldSourceAndConfidence = true in the analyzer configuration, or configure estimateSourceAndConfidence = true for specific fields.

This enables each extracted field to include a confidence score and references back to the original document source location.

This directly satisfies both stakeholder requirements: source grounding allows users to verify where the extracted value came from in the scanned PDF, and the confidence score supports downstream automation rules, such as sending low-confidence fields to manual review. Microsoft's analyzer improvement guidance describes confidence scoring as a value between 0 and 1 and grounding as references or citations for extracted outputs to the original document content.

Generative extraction does not guarantee per-field confidence and source grounding. enableSegment is used for document segmentation, not confidence scoring. Labeled samples can improve extraction quality, but they do not by themselves enable confidence and grounding output. Reference topics: Content Understanding document analyzers, field extraction, confidence scoring, grounding, and RAG ingestion.

### NEW QUESTION: 9

Note: This section contains one or more sets of questions with the same scenario and problem. Each question presents a unique solution to the problem. You must determine whether the solution meets the stated goals. More than one solution in the set might solve the problem. It is also possible that none of the solutions in the set solve the problem.

After you answer a question in this section, you will NOT be able to return. As a result, these questions do not appear on the Review Screen.

You have a multimodal AI generative model that accepts image uploads and uses extracted image text to generate responses.

You discover that users can upload unsafe images and embed hidden instructions into images to manipulate the model.

You need to implement controls to mitigate the risk.

Solution: You configure a prompt shield for documents.

Does this meet the goal?

A. Yes

B. No

Answer: ([SHOW ANSWER](#))

The solution does not fully meet the goal. A prompt shield for documents is the correct control for the embedded-instruction portion of the scenario. Microsoft defines Prompt Shields as protection against prompt manipulation, including attacks embedded in third-party or document-like content that is supplied to a generative model. OCR-extracted text from uploaded images is untrusted contextual content, so document attack protection is appropriate for detecting hidden instructions that attempt to override the model's intended behavior.

However, the scenario contains two separate risks: unsafe image uploads and hidden instructions embedded in images. Prompt Shields for documents address indirect prompt injection, but they do not classify or block harmful visual content in the uploaded image itself. Azure AI Content Safety image moderation is the control that scans images for harmful content categories such as sexual content, violence, hate, and self-harm with severity levels that can be used for blocking decisions.

Therefore, document prompt shielding alone is incomplete. A complete mitigation would combine image moderation for unsafe images with Prompt Shields for document attacks, and optionally Spotlighting for lower-trust third-party content. Reference topics: Content Safety image moderation, Prompt Shields, document attacks, indirect prompt injection, and multimodal safety.

### NEW QUESTION: 10

You have a Microsoft Foundry project named Project1.

Project1 contains an application that processes PDF vendor invoices.

You need to configure Azure Document Intelligence in Foundry Tools to generate a Markdown output that preserves the sections and table structure of the PDFs. The solution must minimize development effort.

What should you do?

A. Increase the confidence threshold.

B. Configure output=figures when you analyze the PDF.

C. Configure content=markdown when you analyze the document.

D. Set the output\_content\_format=ContentFormat. MARKDOWN value.

Answer: D (LEAVE A REPLY)

The correct answer is D. Set the output\_content\_format=ContentFormat.MARKDOWN value . Azure Document Intelligence Layout API can return extracted document content in Markdown format, preserving semantic structure such as headings, paragraphs, sections, tables, and other layout elements. Microsoft's Document Intelligence layout guidance shows the Python SDK pattern for analyzing a document with the prebuilt-layout model and setting output\_content\_format=ContentFormat.MARKDOWN in the begin\_analyze\_document call. The Markdown output is returned in the top-level content section of the analysis result.

This minimizes development effort because the service produces structure-preserving Markdown directly, rather than requiring custom post-processing to reconstruct sections and table formatting from raw OCR spans. Microsoft's Markdown output documentation states that specifying Markdown output produces semantically structured content that maintains paragraphs, headings, tables, and other document elements in their proper hierarchy.

Option A only changes validation behavior and does not generate Markdown. Option B requests figures, not structured Markdown. Option C uses an incorrect parameter name; the documented SDK setting is output\_content\_format, not content. Reference topics: Azure Document Intelligence Layout API, Markdown output, PDF analysis, table extraction, and Foundry Tools document processing.

NEW QUESTION: 11

You have a Microsoft Foundry project that processes procurement documents submitted by suppliers.

You need to implement two pipelines by using Azure Content Understanding in Foundry Tools. The solution must meet the following requirements:

- \* Include a pipeline named Pipeline1 that supports cost-effective, high-volume processing of standalone PDF invoices.
- \* Include a pipeline named Pipeline2 that supports cross-document validation by using multi-step reasoning and reference data.

How should you configure each pipeline? To answer, drag the appropriate configurations to the correct pipelines. Each configuration 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.

Configurations

Multi-file task in pro mode

Multi-file task in standard mode

Single-file task in pro mode

Single-file task in standard mode

Answer Area

Pipeline1: Configuration

Pipeline2: Configuration



Answer:

### Configurations

- Multi-file task in pro mode
- Multi-file task in standard mode
- Single-file task in pro mode
- Single-file task in standard mode

### Answer Area

- Pipeline1 | Single-file task in standard mode
- Pipeline2 | Multi-file task in pro mode



### Explanation:

Pipeline1: Single-file task in standard mode

Pipeline2: Multi-file task in pro mode

Pipeline1 should use a single-file task in standard mode because the workload is high-volume processing of standalone PDF invoices. Azure Content Understanding standard mode is intended for individual files that require straightforward structured extraction, and Microsoft describes it as minimizing cost and latency for broad, data-centric processing scenarios. This makes it the best fit for cost-effective invoice extraction where each PDF can be processed independently.

Pipeline2 should use a multi-file task in pro mode because the requirement includes cross-document validation, multi-step reasoning, and reference data. Microsoft guidance states that pro mode is designed for advanced scenarios requiring multi-step reasoning and cross-file analysis, including processing multiple input files in a single request, validating or enriching data across documents, and using reference data to guide extraction and validation.

Single-file pro mode would add unnecessary capability for Pipeline1 and would not optimize for cost-effective high-volume standalone processing. Multi-file standard mode does not meet the pro-mode requirement for reference-data-based reasoning. Reference topics: Azure Content Understanding standard mode, pro mode, single-file tasks, multi-file tasks, field extraction, and procurement document validation.

### NEW QUESTION: 12

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

**Answer:** ([SHOW ANSWER](#))

The correct role is Cognitive Services OpenAI User . The application is using DefaultAzureCredential, so az login only proves the developer's Microsoft Entra identity and enables token acquisition. It does not by itself grant authorization to the model deployment. Azure OpenAI and Microsoft Foundry separate authentication from authorization; Microsoft Entra ID provides token-based authentication, while Azure RBAC controls whether the signed-in principal can perform data-plane actions such as model inference. Microsoft's Foundry guidance states that Microsoft Entra ID supports granular RBAC and that data-plane operations include runtime usage such as chat completions and embedding generation.

For Azure OpenAI resources, the Cognitive Services OpenAI User role specifically allows users to make inference API calls with Microsoft Entra ID against deployed models, while preventing higher-privilege actions such as creating deployments, copying keys, fine-tuning, or managing the resource. The other roles are not least privilege. Contributor grants broad management-plane permissions. Cognitive Services User is broader and less specific than the Azure OpenAI inference role. Cognitive Services Data Reader is read-oriented and does not provide the required model inference data action. Reference topics: Microsoft Foundry authentication and authorization, Azure RBAC, Microsoft Entra ID keyless authentication, Azure OpenAI v1 API, and least-privilege model inference.

**NEW QUESTION: 13**

You have a Microsoft Foundry project that contains an agent. The agent accepts user-uploaded screenshots and uses a multimodal chat model. Some screenshots contain potentially malicious embedded text. You need to prevent a prompt injection attack and ensure that third-party content is treated as lower trust. How should you configure prompt shields for document attacks? To answer, select the appropriate options in the answer area. NOTE: Each correct selection is worth one point.

Prompt shields action:

- Disable the shield.
- Set action to block.
- Set action to annotate.

Additional mitigation:

- Enable Spotlighting.
- Create a custom blocklist.
- Use optical character recognition (OCR) to extract the text from the images first.

**Answer:**

Prompt shields action:

- Disable the shield.
- Set action to block.
- Set action to annotate.

Additional mitigation:

- Enable Spotlighting.
- Create a custom blocklist.
- Use optical character recognition (OCR) to extract the text from the images first.

Explanation:  
Prompt shields action: Set action to block.  
Additional mitigation: Enable Spotlighting.

The correct configuration is to set the Prompt Shields document attack action to block and enable Spotighting . Prompt Shields in Microsoft Foundry are designed to detect attempts to manipulate model behavior through adversarial input. Microsoft distinguishes document attacks from direct user prompt attacks: document attacks are malicious instructions embedded in third-party content such as documents, webpages, emails, or other externally supplied material. In this scenario, the embedded text inside uploaded screenshots is third-party content and can attempt to override the agent's instructions. Setting the action to block prevents detected document-attack content from being processed normally, which is required because the goal is prevention rather than passive logging or annotation. Spotighting is the additional mitigation because it marks or transforms document content so the model treats it as lower trust than system and user instructions. Microsoft's Foundry guidance describes Spotighting as a Prompt Shields subfeature that helps protect against indirect or embedded document attacks by tagging input documents with special formatting to indicate lower trust. A custom blocklist is insufficient for unknown attacks, and OCR alone only extracts the malicious text; it does not mitigate prompt injection. Reference topics: Prompt Shields, document attacks, guardrails, Spotighting, multimodal safety, and prompt injection defense.

**NEW QUESTION: 14**

You have a Microsoft Foundry project that contains a high-traffic agent. After a recent update, operational costs increase significantly. Monitoring confirms that the volume of user traffic to the agent remains unchanged. You suspect that changes to the request or response characteristics are causing the increase. You need to identify whether the additional costs are driven by the model input size, the model output size, or expanded tool usage. Which observability capability should you use?

- A. evaluation metrics
- B. token usage
- C. latency
- D. run success rate

**Answer: B (LEAVE A REPLY)**

The correct capability is token usage . In Microsoft Foundry observability, token consumption is the primary signal for diagnosing model-cost changes when request volume is unchanged. Token usage lets you distinguish whether costs increased because prompts became larger, retrieved or tool-provided context expanded, responses became longer, or agent execution added more model calls. Microsoft Foundry monitoring dashboards track operational metrics such as token consumption, latency, error rates, and quality scores, and the agent monitoring dashboard is specifically intended to help analyze token usage, latency, success rates, and evaluation outcomes for production traffic. This directly matches the scenario because the issue is not more traffic, but changed request or response characteristics. Input tokens reveal whether the prompt, chat history, grounding data, or tool outputs being sent to the model increased. Output tokens reveal whether the model is generating longer completions. Expanded tool usage can also increase cost indirectly by adding more tool results, intermediate calls, and context into subsequent model requests; Foundry tracing and observability capture tool usage and token consumption for agent runs. Evaluation metrics assess response quality and safety, not cost drivers. Latency identifies performance delays, and run success rate measures reliability. Reference topics: Microsoft Foundry observability, agent monitoring dashboard, token consumption, cost analysis, tool usage, and production monitoring.

**NEW QUESTION: 15**

You have a Microsoft Foundry project that contains three agents as shown in the following table.

| Name        | Description                                            |
|-------------|--------------------------------------------------------|
| TriageAgent | Classifies incoming customer requests                  |
| PolicyAgent | Answers policy questions by searching internal content |

ActionAgent

Creates or updates tickets by calling an HTTP API

You need to orchestrate the agents to ensure that the customer requests meet the following requirements:

- \* Support a deterministic, step-based process that uses conditional branching and shared state across the agents.
- \* Optionally trigger a ticket action based on the triage result.

The solution must minimize development effort.

What should you include in the solution?

- A.** separate agent runs coordinated in the application code
- B.** a multi-agent group chat session
- C.** a workflow
- D.** threads and runs without a workflow

**Answer: C (LEAVE A REPLY)**

The correct answer is a workflow . Microsoft Foundry workflows are designed to orchestrate agents and business logic as declarative, predefined sequences of actions. The official workflow guidance states that workflows are ideal when you need to orchestrate multiple agents in a repeatable process, add branching logic such as if/else, and handle variables without writing application orchestration code. This directly matches the requirement for a deterministic, step-based process with conditional branching and shared state.

In this scenario, TriageAgent can classify the request first, the workflow can store the triage result, and conditional logic can determine whether to invoke PolicyAgent, ActionAgent, or both. The ticket action is optional, so it should be triggered through a workflow condition based on the triage output. This minimizes development effort because the branching, sequencing, and variable handling are managed in the Foundry workflow rather than being manually implemented across separate runs in application code.

A group chat session is better for dynamic agent handoff, not a strict deterministic process. Threads and runs or separate app-coordinated calls require more custom orchestration. Reference topics: Microsoft Foundry workflows, multi-agent orchestration, conditional branching, variable handling, and agent-driven workflows.

### NEW QUESTION: 16

You have a Microsoft Foundry project that contains an agent named PaymentAgent.

PaymentAgent includes a function tool that issues customer refunds by using an external API.

You are creating a workflow in YAML.

You need to ensure that the workflow pauses for human approval and continues with the refund step only after approval is granted.

How should you complete the workflow definition? To answer, select the appropriate options in the answer area.

NOTE: Each correct selection is worth one point.

steps:

- id: propose\_refund

type: agent  
agent: PaymentAgent

- id: approval

type:

- ask\_question
- basic\_chat
- data\_transformation

- id: execute\_refund

type: agent  
agent: PaymentAgent

condition:

- approval == "approved"
- propose\_refund.output != null
- true

Answer:

steps:

Microsoft

- id: propose\_refund

type: agent

agent: PaymentAgent

- id: approval

type:

ask\_question

basic\_chat

data\_transformation

- id: execute\_refund

type: agent

agent: PaymentAgent

condition:

approval == "approved"

propose\_refund.output != null

true

Explanation:

type: ask\_question

condition: approval == " approved "

The approval step must use type: ask\_question because the workflow must pause and wait for a human response before the refund execution proceeds. Microsoft Foundry workflows support human-in-the-loop patterns where the workflow asks the user a question and awaits input before continuing; this pattern is explicitly intended for approval requests and clarifying questions. The workflow guidance also identifies workflows as declarative sequences that orchestrate agents and business logic, including branching logic and human-in-the-loop steps.

The refund execution step must use condition: approval == " approved " so that the second invocation of PaymentAgent runs only when the approval response matches the required approval value. Using true would always execute the refund, bypassing the approval control. Using propose\_refund.output != null would only confirm that the first agent step produced output; it would not prove that a human approved the refund.

data\_transformation is also incorrect for the approval node because it sets or parses values rather than pausing for user input. Reference topics: Microsoft Foundry workflows, human-in-the-loop workflow pattern, YAML workflow editing, agent orchestration, conditional execution, and workflow approval gates.

**Valid AI-103 Dumps** shared by EduDump.com for Helping Passing AI-103 Exam! EduDump.com now offer the **newest AI-103 exam dumps**, the EduDump.com AI-103 exam **questions have been updated** and **answers have been corrected** get the **newest** EduDump.com AI-103 dumps with Test Engine here: <https://www.edudump.com/exams/Microsoft/AI-103/premium/> (159 Q&As Dumps, **35%OFF** Special Discount Code: **freecram**)

**NEW QUESTION: 17**

You have a Microsoft Foundry project that contains an agent.

You need to process mixed-format documents that contain scanned text, tables, and multicolumn layouts. The extracted content must preserve the document structure and be converted into the Markdown format for downstream reasoning.

What should you configure first?

- A. an Azure Content Understanding in Foundry Tools analyzer
- B. a generative chat completion request
- C. an Azure Language in Foundry Tools text analysis model deployment
- D. an Azure OpenAI Responses API call that uses a multimodal model

Answer: (SHOW ANSWER)

The correct answer is A. an Azure Content Understanding in Foundry Tools analyzer . In Azure Content Understanding, an analyzer is the reusable configuration that defines what content type is processed, which elements are extracted, and how the output is structured. Microsoft's analyzer guidance states that analyzers define extraction for text, layout, tables, fields, and output formats such as Markdown and JSON. This makes the analyzer the first required configuration step before downstream reasoning or agent orchestration can use the extracted content.

This matches the requirement because the documents contain scanned text, tables, and multicolumn layouts.

Content Understanding document analysis is designed to transform unstructured documents into structured, machine-readable output while preserving document structures and relationships. Its Markdown representation converts unstructured documents into GitHub Flavored Markdown while maintaining content and layout for downstream use.

A generative chat completion request or Azure OpenAI Responses API call could reason over extracted content, but it is not the correct first step for OCR, layout preservation, and Markdown conversion. Azure Language focuses on text analysis after content has already been extracted. Reference topics: Content Understanding analyzers, document extraction, OCR, layout analysis, Markdown output, and downstream reasoning.

NEW QUESTION: 18

You have a Microsoft Foundry project that contains an agent.

The agent uses tools to retrieve internal content and call external APIs. The agent is configured to let the model decide when to call the tools.

You need to publish the agent for a compliance workflow. The solution must meet the following requirements:

- \* Each workflow run must include a retrieval step before generating a response.
- \* Tool calls must authenticate by using the published agent's own identity.
- \* Tool access must use an identity isolated from other project resources.
- \* Tool access must support audit tracing.

What should you do? To answer, select the appropriate options in the answer area.

NOTE: Each correct selection is worth one point.

Set tool\_choice to:

auto

none

required

Configure the tool to authenticate by:

Storing API keys in prompts

Using the shared project agent identity

Using a distinct agent identity bound to the client application

Microsoft

Answer:



existing images by submitting an input image plus a prompt. For masked edits, the mask explicitly defines the part of the image the model is allowed to change; Microsoft states that the mask parameter defines the area to edit and must match the input image dimensions.

text\_to\_image would create a new image from a prompt and cannot guarantee preservation of the original image. image\_variation generates related variants rather than targeted removals.

image\_to\_image with high strength can regenerate broader areas and may alter unrelated visual details. Mask-based inpainting is the built-in editing control that limits modification to the selected region. Reference topics: Azure OpenAI image editing, mask inpainting, image edit API, input image, mask parameter, and computer vision image generation workflows.

### NEW QUESTION: 20

You have a Microsoft Foundry project that contains an agent. The agent generates summaries from retrieved policy documents.

You need to improve response completeness. The solution must be implemented in the logic of the application code before responses are returned.

What should you do?

- A. Decrease the value of the temperature parameter.
- B. Add a reflection pass before the responses are returned.
- C. Switch to Retrieval Augmented Generation (RAG).
- D. Enable response streaming.

**Answer: B (LEAVE A REPLY)**

The correct answer is B. Add a reflection pass before the responses are returned . A reflection pass is an application-orchestration step in which the generated summary is reviewed before final delivery, typically by asking the model or an evaluator step to check whether the answer covers the retrieved policy evidence and to revise the response when important details are missing. This directly addresses response completeness in application logic before the response is returned. The Microsoft Learn study guide explicitly includes Implement model reflection and Apply prompt engineering techniques to improve responses under optimization and operationalization of generative AI solutions.

This is also consistent with Microsoft Foundry agentic-loop guidance, which identifies reflection and planning cycles as patterns for multi-step reasoning in production agent systems.

Completeness is a response-quality property: Azure AI evaluation defines completeness as whether a response contains all necessary and relevant information with respect to ground truth.

Option C is not correct because the scenario already says the agent generates summaries from retrieved policy documents, which is already a grounded retrieval pattern. Option A mainly reduces randomness, not missing content. Option D improves delivery experience, not answer completeness. Reference topics: model reflection, prompt engineering, agentic loops, response evaluation, and grounded generative AI solutions.

### NEW QUESTION: 21

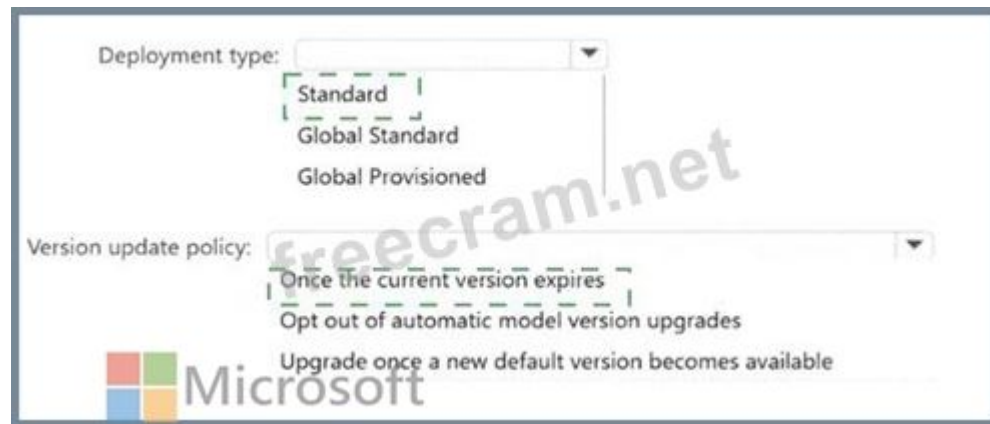
You need to configure the model deployment for Agent1 to meet the technical requirements.

What should you configure? To answer, select the appropriate options in the answer area.

NOTE: Each correct selection is worth one point.



**Answer:**



Explanation:

Deployment type: Standard

Version update policy: Once the current version expires

The correct deployment type is Standard . The case study specifies that Project1 is deployed in an EU Azure region and that model-processed data must remain within the EU. It also requires scalable, high-throughput generative AI workloads that dynamically handle variable customer support traffic without reserved throughput capacity. In Microsoft Foundry Models, Standard is a pay-per-token deployment type that processes data in a single Azure region, while Global Standard can process requests across regions and Global Provisioned uses reserved provisioned throughput. Microsoft's deployment-type guidance identifies Standard as single-region, pay-per-token, whereas Global Provisioned is cross-region with reserved capacity.

The correct version update policy is Once the current version expires . This keeps Agent1 on the selected model version during its supported lifecycle, which supports stable and consistent responses, but still preserves continuity by automatically moving to a supported replacement when the current version is retired.

Microsoft's model versioning guidance states that this policy updates only when the current model version expires, while upgrading when a new default is available changes the deployment sooner and opting out can cause the deployment to stop working after retirement. Reference topics: deployment types, regional data processing, model versioning, throughput capacity, and stable production deployments.

## NEW QUESTION: 22

You have an Azure Speech in Foundry Tools resource that hosts a custom speech to text model deployed to a custom endpoint. An agent uses the endpoint to perform real-time speech recognition.

You are approaching the expiration date of the custom speech to text model.

What is the expected behavior when the model expires?

- A.** Speech recognition requests will fall back to the most recent base model for the same locale.
- B.** Speech recognition requests will continue to use the expired custom model until the model is removed manually.
- C.** Speech recognition requests will return a 4xx error until a new custom model is deployed.
- D.** The custom model will be deleted automatically when the model expires.

**Answer: (SHOW ANSWER)**

The correct answer is A. Speech recognition requests will fall back to the most recent base model for the same locale . Microsoft's custom speech model lifecycle guidance states that when a model expires, it is no longer available for transcription. For the custom endpoint route, speech recognition requests fall back to the most recent base model for the same locale. The documentation also warns that recognition results might still be returned, but the transcription may no longer reflect the domain-specific adaptation of the custom model.

This distinguishes real-time custom endpoint behavior from batch transcription behavior. Batch transcription requests that specify an expired model fail with a 4xx error, but that is not the route described in this question.

The agent is using a custom endpoint for real-time recognition, so fallback to the latest base model is the expected behavior. The model is not automatically deleted merely because it expires, and it does not continue to use the expired custom model indefinitely. The operational recommendation is to update the endpoint's model before expiration by redeploying the endpoint with a newer custom model. Reference topics: Azure Speech custom speech model lifecycle, custom endpoints, model expiration, real-time speech recognition, and endpoint redeployment.

### NEW QUESTION: 23

You need to configure Agent1 to meet the security and compliance requirements.

What should you use?

- A. prompt shields
- B. Personally Identifiable Information (PII) Detection
- C. self-harm content filtering
- D. violence content filtering

**Answer: (SHOW ANSWER)**

The correct answer is B. Personally Identifiable Information (PII) Detection . The case study states that Agent1 must never reveal customer information , even if a document containing customer data is added accidentally to the product sheet repository in storage1. This is a privacy and compliance control requirement, so the appropriate capability is PII Detection.

Azure Language PII Detection is a Foundry Tools capability that identifies, classifies, and redacts sensitive information across text, conversations, and native documents. Microsoft states that PII Detection can be used to implement privacy controls, reduce sensitive data exposure, and support compliance requirements. In this scenario, PII Detection should be applied to retrieved product-sheet content and generated responses so customer names, contact details, identifiers, and other sensitive values are not exposed to users.

Prompt Shields are important for a separate requirement: protecting Agent1 from malicious instructions hidden in documents or embedded text. Microsoft describes Prompt Shields for documents as protection against hidden instructions embedded in external content. However, the option that directly satisfies the requirement to prevent disclosure of customer information is PII Detection. Self-harm and violence filters address harmful-content categories, not privacy leakage.

### NEW QUESTION: 24

You have an application that processes scanned PDF invoices. The invoices have varied layouts and include multipage tables.

You have a pipeline that uses optical character recognition (OCR) and extracts totals and invoice numbers.

The results are often

incorrect because the document structure is ignored.

You need to implement a solution that provides OCR, layout analysis, and template-generalizing field extraction. The solution must NOT require training a custom model. The solution must minimize administrative effort.

What should you include in the solution?

- A. an Azure Machine Learning model
- B. Azure Content Understanding in Foundry Tools
- C. Azure Language in Foundry Tools

**Answer: (SHOW ANSWER)**

The correct answer is Azure Content Understanding in Foundry Tools . The scenario requires more than basic OCR because scanned invoices have varied layouts and multipage tables.

Content Understanding is designed for intelligent document processing and provides OCR, layout detection, table extraction, field extraction, confidence scores, and grounding in a managed service. Microsoft describes Content Understanding as a service that transforms unstructured content into structured outputs and supports invoice processing by extracting and validating fields from complex documents.

This also meets the requirement to avoid training a custom model. Content Understanding includes prebuilt and domain-specific analyzers, including invoice and procurement-style document processing, and Microsoft states that these analyzers provide structured extraction without custom training. It generalizes across visual template variations by using semantic document categories rather than requiring separate models per invoice layout.

Azure Machine Learning would increase administrative effort because it requires model development, training, deployment, and monitoring. Azure Language is optimized for text analytics tasks such as classification and entity extraction after text is available, but it does not provide document layout analysis or multipage table structure extraction. Reference topics: Content Understanding, intelligent document processing, OCR, layout analysis, analyzers, field schemas, and structured extraction.

### NEW QUESTION: 25

Note: This section contains one or more sets of questions with the same scenario and problem. Each question presents a unique solution to the problem. You must determine whether the solution meets the stated goals. More than one solution in the set might solve the problem. It is also possible that none of the solutions in the set solve the problem.

After you answer a question in this section, you will NOT be able to return. As a result, these questions do not appear on the Review Screen.

You have a Microsoft Foundry project that contains an agent. The agent generates summaries from retrieved policy documents.

Users report that some responses omit required regulatory clauses, even when the clauses are present in the retrieved content.

You need to improve response completeness.

Solution: You increase the value of the max\_tokens parameter.

Does this meet the goal?

**A.** Yes

**B.** No

**Answer:** ([SHOW ANSWER](#))

The solution does not meet the goal. Increasing max\_tokens only raises the maximum number of tokens the model is allowed to generate. Microsoft's Azure OpenAI reference defines max\_tokens as the maximum number of tokens allowed for the generated answer, and the quota guidance notes that increasing it can help when responses are being truncated.

In this scenario, the problem is not described as output truncation. The required regulatory clauses are already present in the retrieved policy documents, but the agent omits them during summarization. That is a response completeness issue: Microsoft Foundry RAG evaluator guidance defines response completeness as the recall aspect of the response, meaning the response should not miss critical information compared with expected content or ground truth.

A larger token budget might permit a longer answer, but it does not force the model to identify, verify, or include each mandatory clause. It can also increase cost and latency. The appropriate control is a reflection or completeness verification pass that checks the draft against the retrieved policy clauses and regenerates or revises the response when required content is missing.

Reference topics: RAG response completeness, model output limits, max\_tokens, reflection, and response validation.

## NEW QUESTION: 26

You have a customer support agent that uses the Microsoft Foundry Agent Service.

Sometimes, customers return to a session days later to continue the same support case, and the agent must resume with the full historical context. The agent must provide the following:

- \* Multi-turn continuity within the session

- \* Cross-session continuity for the same case

- \* Access to the full interaction history, including user messages, agent messages, tool calls, and tool outputs

You need to ensure that the agent automatically reloads the complete history on each new turn.

What should you do?

**A.** Persist only the final model response stored in the client application and prepend the response to future prompts.

**B.** Enable memory summarization on the agent definition to persist the context automatically.

**C.** Create and reuse a conversation by storing the conversation's ID and supplying the ID on subsequent requests.

**Answer:** ([SHOW ANSWER](#))

The correct approach is to create and reuse a conversation by storing the conversation's ID and supplying that ID on subsequent requests . In Microsoft Foundry Agent Service, conversations are durable objects with unique identifiers that can be reused across sessions. The official runtime guidance states that conversations store items, including messages, tool calls, tool outputs, and other data, and are intended for multi-turn continuity, cross-session continuity, and inspection of what happened over time. This directly satisfies the requirement to resume the same support case days later with the full historical context.

Persisting only the final model response is insufficient because it loses the full interaction chain, especially tool calls and tool outputs that may be essential to case state. Memory summarization is also not the best fit because the requirement asks for the complete history, not a compressed representation that may omit details.

Reusing the conversation ID allows Foundry to maintain the conversation server-side so the next turn can reuse prior context without the client manually rebuilding prompts. Reference topics: Foundry Agent Service runtime components, conversations, conversation items, multi-turn continuity, cross-session continuity, and tool output history.

### NEW QUESTION: 27

You need to configure Agent1 to answer customer questions about only the Contoso products. The solution must meet the business requirements.

What should you do?

- A. Apply top-p sampling.
- B. Modify the system message instructions.
- C. Add few-shot examples.
- D. Increase the value of the temperature parameter.

**Answer:** ([SHOW ANSWER](#))

The correct answer is B. Modify the system message instructions . The case study states that Agent1 answers general questions about Contoso products and that the business requirement is for Agent1 to answer questions only about the products sold by Contoso . This is a behavioral boundary for the agent, so it should be implemented in the highest-priority instructions that define the agent's role, allowed scope, and refusal behavior.

Microsoft Foundry guidance states that a system message is used to steer model behavior, define the assistant' s role and boundaries, and add safety or quality constraints for the scenario. The system message should instruct Agent1 to answer only when the question concerns Contoso products, use the configured Contoso product documentation as grounding, and politely refuse or redirect questions about non-Contoso products.

Top-p sampling and temperature control randomness, not business-domain scope. Increasing temperature would make responses less deterministic. Few-shot examples can support desired behavior, but examples alone are weaker than explicit system-level instructions for defining operating boundaries. Reference topics:

system message design, prompt engineering, agent instructions, response constraints, and grounded generative AI behavior.

### NEW QUESTION: 28

You need to configure Agent1 to answer customer questions about only the Contoso products. The solution must meet the business requirements.

What should you do?

- A. Apply top-p sampling.
- B. Modify the system message instructions.
- C. Add few-shot examples.
- D. Increase the value of the temperature parameter.

**Answer:** B ([LEAVE A REPLY](#))

The correct answer is B. Modify the system message instructions . The case study states that Agent1 answers general questions about Contoso products and that a business requirement is for Agent1 to answer questions only about products sold by Contoso. This requirement defines the agent's allowed domain and refusal boundary, so it must be expressed in the agent's system-level instructions. Microsoft Foundry guidance states that system messages steer Azure OpenAI chat model behavior and are used to define the assistant's role, boundaries, output format, and safety or quality constraints.

The system message should instruct Agent1 to answer only Contoso-product questions, use Contoso product documentation when available, and decline questions about non-Contoso products. This directly enforces the intended business scope at the highest instruction level. Few-shot examples can reinforce desired behavior but are not the primary control for defining mandatory operating boundaries. Top-p sampling and temperature are decoding controls; they influence randomness and diversity, not whether the agent restricts answers to a specific product domain. Increasing temperature would likely reduce consistency. Reference topics: Microsoft Foundry agent instructions, system message design, prompt engineering, response boundaries, and grounded generative AI behavior.

### NEW QUESTION: 29

Your company is piloting a customer support agent in a Microsoft Foundry project name Project1. Project1 is connected to an existing Application Insights resource, and the company ' s support team reviews runs in the Traces tab.

The Foundry Agent Service is configured to perform the following actions:

\* Retrieve the Application Insights connection string by calling

project\_client.telemetry.get\_application\_insights\_connection\_string()).

- \* Call `configure_azure_monitor(connection_string=...)` to enable telemetry.

A separate LangChain service configured to use OpenTelemetry and has the following configurations:

- \* Uses `AzureAIOpenTelemetryTracer(connection_string=..., enable_content_recording=False)`
- \* Passes the tracer by using `config={ " callbacks " :[azure_tracer]}`

Company policy has the following requirements:

- \* Telemetry from LangChain and OpenTelemetry must be distinguishable within the same Application Insights resource.
- \* Secrets and credentials must NOT be stored in prompts, tool arguments, or span attributes.

For each of the following statements, select Yes if the statement is true. Otherwise, select No.

NOTE: Each correct selection is worth one point.

Answer Area

| Statements                                                                                                        | Yes                   | No                    |
|-------------------------------------------------------------------------------------------------------------------|-----------------------|-----------------------|
| The LangChain service will appear in Traces without configuring a tracer.                                         | <input type="radio"/> | <input type="radio"/> |
| Setting different OTEL_SERVICE_NAME values separates the services in Application Insights.                        | <input type="radio"/> | <input type="radio"/> |
| When using <code>enable_content_recording=False</code> , prompts and tool data will be captured in the telemetry. | <input type="radio"/> | <input type="radio"/> |

Answer:

Answer Area

| Statements                                                                                                        | Yes                              | No                               |
|-------------------------------------------------------------------------------------------------------------------|----------------------------------|----------------------------------|
| The LangChain service will appear in Traces without configuring a tracer.                                         | <input type="radio"/>            | <input checked="" type="radio"/> |
| Setting different OTEL_SERVICE_NAME values separates the services in Application Insights.                        | <input checked="" type="radio"/> | <input type="radio"/>            |
| When using <code>enable_content_recording=False</code> , prompts and tool data will be captured in the telemetry. | <input type="radio"/>            | <input checked="" type="radio"/> |

Explanation:

The LangChain service will appear in Traces without configuring a tracer: No Setting different OTEL\_SERVICE\_NAME values separates the services in Application Insights: Yes When using `enable_content_recording=False`, prompts and tool data will be captured in the telemetry: No

The first statement is No because a separate LangChain or LangGraph application must emit telemetry through the configured tracing integration. Microsoft's LangChain tracing guidance states that you configure `AzureAIOpenTelemetryTracer`, attach it to the runnable or agent through callbacks, and then inspect the emitted traces in Azure Monitor. The troubleshooting guidance also states that missing LangChain or LangGraph spans are caused by tracing callbacks not being attached to the run.

The second statement is Yes . In OpenTelemetry, OTEL\_SERVICE\_NAME maps to the service.name resource attribute. Azure Monitor Application Insights uses cloud role names to represent separate services, and Microsoft states that when multiple services emit to the same Application Insights resource, cloud role names must be set so services are represented properly. The third statement is No . enable\_content\_recording=False is specifically used to redact message content and tool call arguments from traces. Microsoft also advises disabling content recording in production and not storing secrets, credentials, or tokens in prompts or tool arguments. Reference topics: Microsoft Foundry tracing, LangChain tracing, OpenTelemetry service naming, Application Insights, and secure telemetry configuration.

**Valid AI-103 Dumps** shared by EduDump.com for Helping Passing AI-103 Exam! EduDump.com now offer the **newest AI-103 exam dumps**, the EduDump.com AI-103 exam **questions have been updated** and **answers have been corrected** get the **newest** EduDump.com AI-103 dumps with Test Engine here: <https://www.edudump.com/exams/Microsoft/AI-103/premium/> (159 Q&As Dumps, **35%OFF** Special Discount Code: **freecram**)