Prepare Smart for Success Free Adobe AD0-E902 Exam Questions and Answers

Ready to pass faster? Grab free and updated Adobe Workfront Fusion Developer Professional exam PDF questions now. Get authentic AD0-E902 dumps packed with verified answers and secure your certification success with PrepBolt AD0-E902 exam pdf questions and answers.

Thank you for Downloading AD0-E902 exam PDF Demo

https://prepbolt.com/AD0-E902.html

QUESTIONS & ANSWERS

DEMO VERSION

(LIMITED CONTENT)

Question 1

Question Type: MultipleChoice

Refer to the exhibit.

In a scenario, there are two aggregators displaying different numbers in the bundle inspector: Aggregator 1 displays a 1, and Aggregator 2 displays a 10. What is a possible reason for this difference?

Options:

A- Aggregator 1's route is set to process the 1st bundle only, and Aggregator 2's route is set to process all bundles

B- The source module for the aggregators are different

C- Aggregator 2 displays a 10 in the bundle inspector because it is set to repeat 10 times

D- The scenario's router is set to only allow projects through the route to Aggregator 1 and only allows tasks through the route to Aggregator 2

Answer:

B

Explanation:

Step by Step Comprehensive Detailed Explanation:

Understanding the Scenario:

The diagram shows two routes, each leading to an aggregator module (Aggregator 1 and Aggregator 2).

The bundle inspector indicates different output counts for the two aggregators: Aggregator 1 displays 1, and Aggregator 2 displays 10.

Option Analysis:

A . Aggregator 1's route is set to process the 1st bundle only, and Aggregator 2's route is set to process all bundles:

Incorrect. Aggregators process data from their input modules based on their configuration. While filters or limits might be applied earlier in the flow, the scenario does not suggest that the aggregator's configuration explicitly limits bundles in this way.

B . The source module for the aggregators are different:

Correct. The two aggregators receive input from different modules. For example, Aggregator 1 might aggregate bundles from 'Projects,' which outputs a single bundle, while Aggregator 2 aggregates bundles from 'Issues,' which outputs 10 bundles. This explains the differing numbers in the bundle inspector.

C . Aggregator 2 displays a 10 in the bundle inspector because it is set to repeat 10 times:

Incorrect. Aggregators do not 'repeat' a specific number of times. Instead, they process the input bundles passed to them and output a result based on the aggregation logic.

D . The scenario's router is set to only allow projects through the route to Aggregator 1 and only allows tasks through the route to Aggregator 2:

Incorrect. While the router might direct specific bundles (e.g., 'Projects' to Aggregator 1 and 'Issues' to Aggregator 2), the difference in bundle counts is determined by the source modules, not by the router's configuration alone.

Why Source Modules Determine Bundle Counts:

Each aggregator processes data from a source module. If the source module outputs different numbers of bundles, the aggregators will show different bundle counts.

In this example, 'Projects' might output only 1 bundle (e.g., 1 project), while 'Issues' outputs 10 bundles (e.g., 10 issues).

How to Verify:

Inspect the source modules feeding into each aggregator. Check the number of bundles they produce during the scenario run.

Confirm that the aggregators are aggregating based on their respective inputs.

Reference: This explanation aligns with Workfront Fusion's handling of bundles and aggregation logic. Differences in bundle counts at the aggregator level are typically due to differences in the source module outputs, as shown in the scenario.

Question 2

Question Type: MultipleChoice

A Fusion user must archive the last five versions of a scenario for one year.

What should the user do?

Options:

A- Save the scenario frequently
B- Download the scenario blueprints
C- Clone the scenario anytime the design changes
D- Find previous versions using the History tab

Answer:

B

Explanation:

Step by Step Comprehensive Detailed Explanation:

Understanding the Requirement:

The user needs to archive the last five versions of a scenario for one year.

Archiving ensures there is a record of previous versions in case rollback or review is needed.

Option Analysis:

A . Save the scenario frequently:

Incorrect. While frequent saving ensures changes are not lost, it does not provide an archival mechanism for version history.

B . Download the scenario blueprints:

Correct. Downloading blueprints of the scenario allows the user to store version snapshots externally. Blueprints include the complete design and settings of the scenario, making them ideal for archival purposes.

C . Clone the scenario anytime the design changes:

Incorrect. Cloning creates duplicates of the scenario but does not inherently manage or track version history for archival purposes.

D . Find previous versions using the History tab:

Incorrect. The History tab only shows recent edits and logs but does not provide a long-term archiving solution.

Why Downloading Blueprints is Best:

External Storage: Blueprints can be downloaded and stored securely for long-term use.

Restoration: A saved blueprint can be re-imported into Fusion to restore a scenario exactly as it was.

Implementation Steps:

Go to the scenario in Workfront Fusion.

Use the Download Blueprint option to save a copy of the scenario.

Label and organize the blueprints by version and date for easy retrieval later.

Question 3

Question Type: MultipleChoice

A customer wants all their Salesforce Opportunities to sync with their connected projects in Workfront -approximately 20,000+ projects.

After the admin sets a Workfront Fusion scenario to run each night and perform this action, the scenario is run once to test. After 40 minutes, it unexpectedly stops running.

Why did this occur?

Options:

A- Workfront has a limit to the number of API calls it can receive and stopped the scenario from running

B- Workfront Fusion occasionally times out if trying to process over 2000 records within a 40-minute period

C- Workfront Fusion has an execution timeout and likely stopped the scenario from running

D- The Workfront API stops integration webhooks if they are hit more than 2000 times in a 10 minute period

Answer:

C

Explanation:

Understanding the Issue:

The customer is syncing 20,000+ Salesforce Opportunities with Workfront projects using a scheduled Fusion scenario.

After running for 40 minutes, the scenario unexpectedly stops.

Why Option C is Correct:

Workfront Fusion Execution Timeout:

Fusion scenarios have a default execution timeout of 40 minutes per run.

If the scenario exceeds this time limit, Fusion automatically stops the execution to avoid resource overuse.

Handling Large Data Sets:

Scenarios involving large datasets (like syncing 20,000+ records) may require optimizations, such as breaking the data into smaller chunks using paginated requests or iterators.

In this case, the scenario stopped because the execution timeout was reached, not due to API limits or webhook restrictions.

Why the Other Options are Incorrect:

Option A ('Workfront API call limit'):

While Workfront does have API rate limits, they are generally generous and not the reason for the scenario stopping. Fusion scenarios are designed to manage API calls efficiently.

Option B ('Fusion times out if processing over 2000 records in 40 minutes'):

This is incorrect because Fusion does not have a hard limit on the number of records processed in 40 minutes. The timeout is time-based, not record-based.

Option D ('Workfront API stops webhooks after 2000 hits in 10 minutes'):

This does not apply to Fusion scenarios. Webhooks are separate from the API calls initiated by Fusion.

How to Resolve the Issue:

Split the Data: Use pagination or batch processing to divide the 20,000+ records into smaller chunks (e.g., 1,000 or 2,000 records per run).

Adjust Scheduling: Schedule the scenario to run more frequently with smaller batches, ensuring all records are synced over multiple runs.

Use Iterators: Add an Iterator module to loop through smaller subsets of data, preventing the scenario from exceeding the execution timeout.

Steps to Optimize the Scenario:

Add a Search Module to retrieve opportunities in smaller batches (e.g., using limits or pagination parameters).

Use a Repeater Module to process each batch iteratively.

Save the scenario and schedule it to run nightly or more frequently, depending on the sync requirements.

Reference and Supporting Documentation:

Adobe Workfront Fusion: Execution Timeout Limits

Workfront Community: Managing Large Data Sets in Fusion Scenarios

By optimizing the scenario to handle smaller batches of data, the admin can avoid the execution timeout issue and ensure successful syncing of Salesforce Opportunities with Workfront projects.

Question 4

Question Type: MultipleChoice

A web service provides the following array named "Colors":

Which expression returns the first ID in the array?

A.

B.

C.

Options:

A- Option A
B- Option B
C- Option C

Answer:

B

Explanation:

Understanding the Array and the Task:

Input Array (Colors):

[

{ 'ID': '22342', 'name': 'Red' },

{ 'ID': '33495', 'name': 'Blue' }

]

Goal: Extract the first ID from the array, which is '22342'.

Why Option B is Correct:

The expression get(map(2.Colors; ID); 1):

map(2.Colors; ID): Iterates over the array 2.Colors and extracts the ID field from each object. This creates a new array containing just the IDs: ['22342', '33495'].

get(...; 1): Retrieves the first element of the newly created array, which is '22342'.

Why the Other Options are Incorrect:

Option A (map(2.Colors; ID; ID; 1)):

This syntax is invalid because the additional ID and 1 parameters are misplaced. The map function requires only two arguments: the array and the field to map.

Option C (map(get(2.Colors; ID); 1)):

This incorrectly attempts to use get inside map. The get function does not return a field for mapping, so the syntax is invalid.

How the Expression Works:

Step 1: map(2.Colors; ID)

Extracts the ID field from each object in the Colors array.

Output: ['22342', '33495'].

Step 2: get(...; 1)

Retrieves the first element of the mapped array.

Output: '22342'.

Use Case in Workfront Fusion:

This approach is commonly used when processing arrays in Fusion scenarios, ensuring specific elements are accessed without additional looping or complex logic.

Reference and Supporting Documentation:

Adobe Workfront Fusion Functions Documentation

Workfront Community: Using Map and Get Functions

By combining map and get, this expression efficiently extracts the first ID from the array, ensuring correct and reliable results.

Question 5

Question Type: MultipleChoice

A query returns a partial list of possible values.

Which flow control module should be used to ensure all the possible results are queried?

Options:

A- Aggregator
B- Repeater
C- Iterator
D- Router

Answer:

B

Explanation:

Understanding the Requirement:

The query returns only a partial list of possible values.

The task is to ensure that all results are processed by iterating through multiple queries or pages of data.

Why Option B ('Repeater') is Correct:

The Repeater module is designed to repeat an operation a specified number of times or until a condition is met.

It is commonly used for querying paginated data or when a system limits the number of records returned in a single request.

In this case, the Repeater ensures all possible values are queried by making additional requests to retrieve subsequent pages or results.

Why the Other Options are Incorrect:

Option A ('Aggregator'):

The Aggregator combines multiple data bundles into a single output. It does not handle iterative queries or pagination.

Option C ('Iterator'):

The Iterator splits an array into individual items for processing. It does not handle querying for additional data or looping through requests.

Option D ('Router'):

The Router splits the flow of a scenario into multiple paths based on conditions. It is unrelated to iterative querying.

Steps to Configure the Repeater:

Add the Repeater module to your scenario.

Configure the number of repetitions or the condition to continue querying (e.g., based on the presence of additional data).

Link the Repeater to the module responsible for retrieving the data, ensuring it processes all available results.

How This Solves the Problem:

The Repeater module ensures that all possible results are queried by iteratively sending requests until no more data is available.

Reference and Supporting Documentation:

Adobe Workfront Fusion: Repeater Module Documentation

Workfront Community: Using Flow Control Modules

Question 6

Question Type: MultipleChoice

A Fusion user needs to connect Workfront with a third-party system that does not have a dedicated app connector in Fusion.

What should the user do to build this integration?

Options:

A- Determine the API structure and authentication protocols for the third-party system and then use the appropriate Universal Connector
B- Create a new connection to the third-party system in the connections area and then the Universal Connectors will be available for use
C- Use the Workfront Custom API module to set up the connection using API calls to the third-party system

Answer:

A

Explanation:

Understanding the Requirement:

If a third-party system does not have a dedicated app connector in Workfront Fusion, users can still build an integration using Universal Connectors.

Universal Connectors in Fusion allow users to configure custom API calls, enabling communication with systems that lack pre-built integrations.

Steps to Build the Integration:

Determine the API Structure: Review the third-party system's API documentation to understand the available endpoints, data formats (e.g., JSON, XML), and request/response structure.

Identify Authentication Protocols: Determine how the third-party system handles authentication (e.g., API keys, OAuth 2.0, Basic Auth).

Configure the Universal Connector: Use modules like HTTP Request or Webhook to make API calls to the third-party system based on the documented structure.

Why Not Other Options?

B . Create a new connection to the third-party system in the connections area and then the Universal Connectors will be available for use: Creating a new connection in the connections area is only applicable for predefined connectors, not for Universal Connectors, which require manual configuration for unsupported systems.

C . Use the Workfront Custom API module to set up the connection using API calls to the third-party system: The Workfront Custom API module is specifically designed for Workfront's own API, not for connecting to third-party systems.

Adobe Workfront Fusion Documentation: Using Universal Connectors for Custom Integrations

Experience League Community: Integrating Third-Party Systems Using Workfront Fusion Universal Modules

Question 7

Question Type: MultipleChoice

Refer to the exhibit.

This scenario shows a 1 in the bundle inspector for the Tasks module and a 23 in the bundle inspector for the Project module.

What does the number in the bundle inspector represent?

Options:

A- The number of seconds to process the module
B- The number of output bundles
C- The number of operations performed
D- The number of times a module has been edited

Answer:

B

Explanation:

Step by Step Comprehensive Detailed Explanation:

Understanding the Scenario:

In Workfront Fusion, each module in a scenario processes data and generates bundles as output.

The bundle inspector shows the number of bundles (data packets) output by a module during an execution.

Option Analysis:

A . The number of seconds to process the module:

This is incorrect. The number in the bundle inspector does not indicate time but rather the count of output bundles. Processing time is not displayed in this way.

B . The number of output bundles:

Correct. The number displayed in the bundle inspector represents how many bundles the module output during the execution. In the given example, the 'Tasks' module outputs 1 bundle, and the 'Project' module outputs 23 bundles.

C . The number of operations performed:

This is incorrect. The bundle inspector displays the number of output bundles, not operations. While operations may be a result of processing bundles, they are tracked separately in Fusion reports.

D . The number of times a module has been edited:

This is incorrect. Editing history is not displayed in the bundle inspector.

Explanation of Bundle Inspector:

Each module processes input data and generates output bundles.

These numbers in the bundle inspector indicate how many bundles the module is outputting in the current run of the scenario.

For example, if a 'Search' module retrieves 23 records, the bundle inspector will show 23, meaning the module outputs 23 bundles.

Context of the Given Image:

The 'Tasks' module processes and outputs 1 bundle.

The 'Project' module processes 1 input bundle (from 'Tasks') and outputs 23 bundles.

Reference: This information is consistent with Workfront Fusion documentation, which explains the bundle inspector's function during scenario execution. The bundle inspector is used to monitor data processing and ensure expected outputs from modules.

Question 8

Question Type: MultipleChoice

What are two required elements of a test case? (Choose two.)

Options:

A- Expected outcome of test
B- Name of test owner
C- Clear procedure for completing the test
D- Source code being tested

Answer:

A, C

Explanation:

A . Expected Outcome of Test

A test case must clearly state what the expected outcome is, providing a standard against which the results can be measured.

This ensures testers can validate whether the scenario behaves as intended.

C . Clear Procedure for Completing the Test

A well-defined procedure outlines the exact steps required to execute the test, ensuring consistent and repeatable testing.

This reduces ambiguity and helps identify whether errors are due to the scenario configuration or improper test execution.

Why Not Other Options?

B . Name of Test Owner: While helpful for accountability, the name of the test owner is not a required component of the test case itself.

D . Source Code Being Tested: Fusion scenarios do not typically involve source code. Instead, the focus is on workflow execution and configuration, making this element irrelevant.

Workfront Training Materials: Test Case Design Best Practices

Adobe Workfront Fusion Documentation: Testing and Debugging Scenarios

Question 9

Question Type: MultipleChoice

A global customer has end users who input date and currency data into fields in inconsistent formats. Despite continued training efforts, this continues to be an issue. Unfortunately, the third-party service that the end users are using does not support forcing a required field format. These input mistakes do not happen frequently, but they currently stop the scenario from executing after the default three retries.

In Fusion, which action can the admin take to ensure that errors are corrected after they occur in a scenario?

Options:

A- Select storing of Incomplete Executions in the scenario settings. The customer admin can then filter and search the execution history to resolve errors as they occur.
B- Use the switch module to catch dates not in the required format and convert the common misused patterns into the appropriate format to prevent the scenario from stopping after three failed executions.
C- Set up an error handling path that will catch errors in the end user's inputs and message the users in an email update that they need to try again.

Answer:

A

Explanation:

Understanding the Scenario:

The issue involves end users inputting inconsistent date and currency formats that result in errors in a Workfront Fusion scenario.

The default behavior of Fusion stops the scenario after three retries due to input mismatches or invalid formats.

Why Option A is Correct:

Storing Incomplete Executions: In Adobe Workfront Fusion, enabling 'Store incomplete executions' in the scenario settings allows administrators to capture incomplete runs without fully stopping the entire process. This feature stores all relevant data, even from incomplete runs, allowing administrators to identify and correct input issues manually.

Error Troubleshooting: By reviewing incomplete executions, admins can pinpoint where the scenario failed, resolve the issue, and potentially reprocess those incomplete records, preventing complete scenario stoppage.

Why Option B is Incorrect:

The 'switch module' can handle specific input variations, but it is not a comprehensive solution for handling unexpected or inconsistent formats entered by end users. While it might mitigate some errors, it does not address the root cause and can miss unanticipated input patterns.

Why Option C is Incorrect:

Setting up an error handling path to notify users and request corrections adds an additional manual step for users but does not resolve the problem efficiently within Fusion. Moreover, this solution does not prevent the scenario from halting after retries.

Steps to Enable Storing Incomplete Executions:

Navigate to the scenario in Adobe Workfront Fusion.

Open the Scenario Settings by clicking the gear icon.

Enable the option Store Incomplete Executions under Execution settings.

Save the settings and activate the scenario.

How This Solves the Problem:

Enabling this setting ensures that no data is lost when the scenario fails due to inconsistent inputs. Admins can access the incomplete executions through the scenario execution history, apply necessary corrections, and retry specific records or steps as needed.

Reference and Supporting Documentation:

Adobe Workfront Fusion Official Documentation: Scenario Settings

Workfront Community: Handling Incomplete Executions