Adobe

AD0-E902 Exam

Adobe Workfront Fusion Professional

Questions & Answers
Demo

Version: 4.0

Question: 1

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?

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

Question: 2

A solution requested for a use case requires that the scenario is initiated with project updates. Which Workfront app module will start the scenario immediately?

A. Watch Events
B. Watch Record
C. Watch Field
D. Search

Answer: A

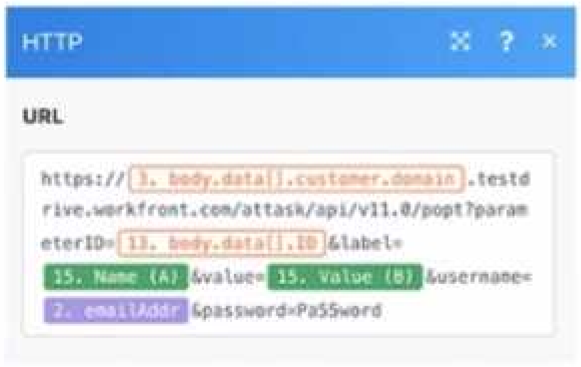
Explanation:

Understanding the Questio n:
The scenario must begin as soon as a project update occurs in Adobe Workfront.
The correct Workfront module should continuously monitor for specific changes (in this case, project updates) and trigger the scenario immediately.
Why Option A ("Watch Events") is Correct:
Watch Events Module: This module in Adobe Workfront Fusion is specifically designed to monitor events, such as updates to projects, tasks, or issues, and trigger scenarios as soon as those events occur.
Real-Time Triggering: The "Watch Events" module listens to the Workfront event stream and ensures the scenario starts immediately upon detecting relevant updates.
Example Use Case: Monitoring updates to a project’s status, such as changes in "Completion" or "Progress," to trigger notifications or integrations with other systems.
Why the Other Options are Incorrect:

Option B ("Watch Record"): This module monitors specific Workfront records (e.g., projects, tasks, issues) for new additions or modifications, but it does not initiate scenarios immediately when updates occur. It works better for periodic checks rather than real-time events.
Option C ("Watch Field"): This module monitors changes to specific fields within a Workfront object, but it is not designed for broader event monitoring like project updates. It is more suited for field-specific tracking.
Option D ("Search"): This module performs queries to find specific data in Workfront (e.g., searching for projects based on criteria), but it is not an event-driven module and does not automatically trigger scenarios.
Steps to Configure the Watch Events Module in Workfront Fusion:
In the Fusion scenario editor, add the Watch Events module as the first step in your scenario.
Configure the module:
Select Workfront Connection: Choose the authorized Workfront account.
Event Object: Specify the object type (e.g., Project, Task, Issue) to monitor.
Event Type: Select the type of event to watch, such as "Update" or "Change."
Save and activate the scenario.
How This Solves the Problem:
Using the Watch Events module ensures the scenario is event-driven and starts automatically when the desired project update occurs. This approach is both efficient and timely, meeting the requirement for immediate initiation.
Reference and Supporting Documentation:
[Adobe Workfront Fusion Official Documentation: Watch Events Module]
[Workfront Community Forum: Use Cases for Watch Events]

Question: 3

A user needs to dynamically create custom form field options in two customer environments.



```
https://3. body.data[].customer.domain.testd
rive.workfront.com/attask/api/v11.0/popt?param
eterID=13. body.data[].ID&label=
15. Name (A)&value=15. Value (B)&username=
2. emailAddr&password=Pa55word
```

Given this image, which type of Workfront module is referenced in the formula with the parameterID value?

A. Custom API Call
B. Misc Action
C. Read Related Records
D. Search

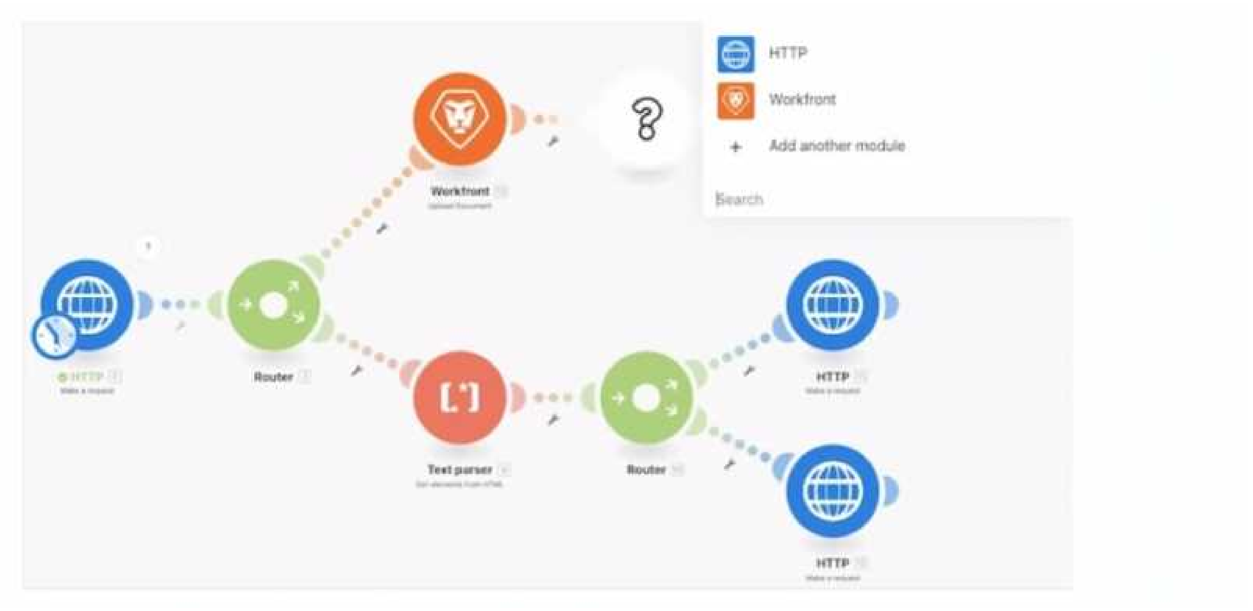
Answer: A

Explanation:

Understanding the Image and Context:
The image provided represents an HTTP module in Workfront Fusion with a URL that dynamically references various data points (e.g., parameterID, customer.domain, emailAddr).
The structure of the URL indicates a call to the Workfront API (/api/v1.0/), using parameters to pass dynamic data such as parameterID, username, and password.
Why Option A ("Custom API Call") is Correct:
The HTTP module shown in the image is a custom API call because it interacts with Workfront's API endpoints by passing dynamic parameters through the URL.
Custom API Call modules allow users to manually configure requests to endpoints in cases where no predefined Workfront Fusion module exists for the operation. This is evident in the example, where specific fields like parameterID, customer.domain, and others are manually mapped to the API URL.
Example Use Case: Dynamically creating custom form field options by sending a POST/PUT request to the Workfront API with specific parameters (like label and value) for each environment.
Why the Other Options are Incorrect:
Option B ("Misc Action"): This refers to predefined actions in Workfront Fusion for handling simple tasks. The HTTP module is not categorized under Misc Actions as it involves direct API interaction.
Option C ("Read Related Records"): This module is used to fetch data related to Workfront objects (e.g., related tasks or documents). It doesn’t allow dynamic parameter passing or URL customization as seen here.
Option D ("Search"): The Search module is used for querying Workfront objects based on specific criteria but does not involve making direct API calls or sending HTTP requests with custom parameters.
Steps to Configure a Custom API Call in Workfront Fusion:
Add the HTTP Module to your scenario.
Select the appropriate HTTP method (e.g., GET, POST, PUT). In this case, a POST or PUT method would be used to create or update custom form fields.
Enter the API endpoint in the URL field, as shown in the image.
Map dynamic values to the parameters by referencing fields from previous modules in the scenario.
For instance:
customer.domain: Extracted from prior steps.
parameterID, label, and value: Dynamically passed based on input data.
Authenticate the request using a username and password or an API token.
Test the module to ensure the API call works as expected.
How This Solves the Problem:
By using a Custom API Call (via the HTTP module), the user can dynamically interact with the Workfront API to create or modify custom form field options across multiple customer environments, passing the required parameters programmatically.
Reference and Supporting Documentation:
Adobe Workfront Fusion HTTP Module Documentation
Workfront API Documentation
Workfront Fusion Community Forum: Using HTTP Module for API Calls

Question: 4

Given this Fusion scenario, a user needs to access multiple fields from the Workfront module for mapped expressions in the HTTP PUT requests.



Which action should the user take?

A. Set Multiple Variables module after the Workfront module. Get Multiple Variables between the Text Parser and the bottom Router.
B. Set Multiple Variables module after the Workfront module. Get Variable modules just before each HTTP module for the specific variables needed in each bottom path.
C. Set Variable module after the Workfront module. Get Variable modules just before each HTTP module in the bottom paths.

Answer: A

Explanation:

Understanding the Scenario:
The image represents a Workfront Fusion scenario with a Workfront module, HTTP modules, and routers splitting the execution path.
The goal is to reuse multiple fields from the Workfront module (e.g., data extracted or processed earlier in the flow) as mapped expressions in HTTP PUT requests located in the bottom paths.
Why Option A is Correct:
Set Multiple Variables Module: This module allows you to define and store multiple variables at a single point in the scenario (e.g., after the Workfront module). These variables can then be reused throughout subsequent steps in the scenario.
Get Multiple Variables Module: By placing this module between the Text Parser and the bottom Router, you can retrieve all previously stored variables, ensuring consistent access across all branches of the flow. This avoids redundancy and ensures the data is easily accessible for each HTTP request in the bottom paths.
Why the Other Options are Incorrect:
Option B ("Set Multiple Variables after Workfront, Get Variables before each HTTP module"): This is

partially correct but less efficient. Adding multiple Get Variable modules before each HTTP request results in repetitive configuration and increases maintenance complexity.
Option C ("Set Variable and Get Variable for each HTTP module"): Using individual Set and Get Variable modules increases duplication. This approach requires separate variables for each data point, which is inefficient compared to using the Set/Get Multiple Variables module for multiple fields at once.
Steps to Configure the Solution:
After the Workfront module:
Add a Set Multiple Variables module.
Define all the fields required for the HTTP PUT requests as variables, mapping them from the Workfront module outputs.
Between the Text Parser and the bottom Router:
Add a Get Multiple Variables module.
Retrieve the previously stored variables, ensuring they are accessible for all paths.
In the HTTP modules on each bottom path:
Use the retrieved variables for mapping in the PUT requests.
How This Solves the Problem:
This approach centralizes variable management, making it easier to access and modify data as needed.
It avoids redundancy, as variables are defined once and reused across all paths, reducing the risk of errors and ensuring consistency.
Reference and Supporting Documentation:
Adobe Workfront Fusion: Variables Module Overview
Workfront Community: Efficient Use of Variables in Fusion

Question: 5

In a scenario that searches for recently completed tasks, a user notices the following input and output for a date transformation.
Input: March 3, 2021 10:34 AM Output: March 1, 2021 10:34 AM
Which expression produces this date transformation?

A. subDays(now,2)
B. addHours(now; -48)
C. addDays(today; -4)

Answer: A

Explanation:

Understanding the Date Transformation:
Input: March 3, 2021, 10:34 AM
Output: March 1, 2021, 10:34 AM
The transformation subtracts 2 days from the input date without altering the time.
Why Option A is Correct:
subDays(now,2) subtracts exactly 2 days from the given date and time.
It preserves the time component of the input (10:34 AM) while shifting the date backward by 2 days,

which matches the given output.
Why the Other Options are Incorrect:
Option B ("addHours(now; -48)"): While subtracting 48 hours also results in a 2-day difference, this approach directly modifies the time. The resulting time could shift if the operation crosses daylight saving changes or edge cases with leap seconds. It is less reliable compared to subDays.
Option C ("addDays(today; -4)"): This would subtract 4 days, which does not match the transformation shown in the example.
Reference and Supporting Documentation:
Adobe Workfront Fusion: Date and Time Functions
Workfront Community: Using Date and Time Expressions