PassQueen
a.k.a dumpsqueen.com

Adobe Workfront Fusion Professional

Adobe AD0-E902

Version Demo

Total Demo Questions: 7

Total Premium Questions: 71

Buy Premium PDF

https://passqueen.com

support@passqueen.com

Topic Break Down

Topic	No. of Questions
Topic 1, Workfront Fusion Fundamentals	9
Topic 2, Scenario Development	40
Topic 3, Scenario Testing and Troubleshooting	17
Topic 4, Administration and Maintenance	5
Total	71

QUESTION NO: 1

REST APIs commonly implement CRUD operations, including Create, Read, Update, and Delete.

Which two actions is an object ID generally required to be provided as an input? Choose two.

A. Update

B. Delete

C. Respond

D. Create

ANSWER: A B

Explanation:

Update and **Delete** generally require an object ID because both operations act on one already-existing, specific resource. An update request must identify the exact record whose fields should be changed. Likewise, a delete request must unambiguously identify the record to remove. In REST-style APIs, this identifier is commonly supplied in the endpoint path, such as `/task/{ID}`, although an API can alternatively accept it as a request parameter or in a request body depending on its design.

This aligns with Adobe Workfront API usage: Workfront objects have unique IDs, and API operations that retrieve or modify a particular object use that ID to target the intended record. In Fusion, when configuring calls to Workfront or another REST API, mapping the ID returned by an earlier search, create, or trigger module is therefore a standard pattern before performing an update or deletion. Adobe documents object IDs and API operations in its Workfront API basics, and its REST API reference describes endpoints and object-specific operations in the Adobe Workfront API overview.

QUESTION NO: 2

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

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: D

Explanation:

Search is the module type referenced to supply the dynamic `parameterID` value. Custom form fields are represented by Parameter records in Workfront, and the identifier of a given field can differ between customer environments even when the fields have the same name or business purpose. A Search module can locate the appropriate Parameter record in the current environment using a stable criterion, such as its name or another identifying attribute. Its returned record ID can then be mapped into the `parameterID` field or request parameter when creating the associated custom-field option.

This approach makes the scenario portable: rather than hard-coding an ID that only exists in one environment, the scenario resolves the current environment's Parameter ID at runtime and uses that result for the option-creation request. Adobe's Workfront Fusion documentation describes Search as the Workfront module used to retrieve records that match specified search criteria, while the Workfront API documentation explains that API operations use object identifiers to associate records and perform actions. See Adobe Workfront modules in Fusion and Adobe Workfront API basics.

QUESTION NO: 3

A cloud-storage module returns an array containing several file collections. Each file must be uploaded as a separate document to the same Workfront project.

Which module should be placed before the Workfront Upload Document module?

A. Array Aggregator

B. Iterator

C. Router

D. Repeater

ANSWER: B

Explanation:

An **Iterator** is correct because it converts an array into individual bundles. Each emitted bundle contains one file collection, allowing the Upload Document module to run once for each file and map that file's name and content into the upload fields.

An Array Aggregator combines multiple bundles and would move processing in the opposite direction. A Router creates separate routes but does not divide an array into file-level bundles. A Repeater runs a route a specified number of times but does not inherently supply each file collection from the array.

QUESTION NO: 4

A third-party application returns an array of line-item collections. Each collection contains an `sku` field. A downstream module requires one text value containing all SKU values separated by commas.

Which mapping expression should the user use?

A. `join(1.lineItems[]; ",")`

B. `map(1.lineItems[]; "sku")`

C. `join(map(1.lineItems[]; "sku"); ",")`

D. `concat(1.lineItems[]; "sku"; ",")`

ANSWER: C

Explanation:

`join(map(1.lineItems[]; "sku"); ",")` first extracts the `sku` value from every collection in the `lineItems` array, then joins the resulting array into one comma-separated text value. `join` alone cannot select a field from each collection, and `map` alone returns an array rather than the required text string.

QUESTION NO: 5

A scenario must translate a source user's department and job role into a destination job role. The mapping changes infrequently and must remain available across scenario executions. A Fusion data store will contain the approved mappings.

Which two actions should the developer take? (Choose two.)

A. Create a unique key from the source department and job role values

B. Use a Get a record module to retrieve the mapping by its key

C. Use an Array Aggregator to retain the mapping between executions

D. Use a Router with one route for every possible department and role combination

ANSWER: A B

Explanation:

Create a unique composite key and use a Get a record module to retrieve the matching mapping record. A composite key, such as a consistently constructed department-and-role value, identifies the specific mapping required for each source user. The data store can then retain the corresponding destination role as record data between scenario executions.

A Get a record module retrieves the approved mapping at runtime. An Array Aggregator is used to combine bundles during an execution and is not persistent storage. Hard-coding role logic in a Router makes ongoing maintenance dependent on scenario edits rather than controlled data-store records.

QUESTION NO: 6

A Fusion scenario updates project conditions each night, and should set the project condition to At Risk if there are any high priority open issues on the project. The scenario retrieves all open projects and cycles through the projects. For each project with issues, it retrieves all associated open issues, iterates through them and sets the project condition to At Risk if the issue is high priority or On Target if it is not.

A user notices that Fusion is updating the progress condition multiple times, once for each issue in the project.

How can the developer ensure the project is updated only once?

A. Change the issue search module to result set of First Matching

B. Record Add an Ignore error directive as an error handler route for the update module

C. Create a separate scenario to update the overall project condition

D. Apply the Run Once flow control function

E. Add an Array Aggregator grouped by project ID, then evaluate the aggregated issues and update the project.

ANSWER: E

Explanation:

Use an Array Aggregator to collect the issue bundles for each project, then perform the project update after the aggregation. In Fusion, an iterator or search that returns multiple issues produces a separate bundle for each issue, so a downstream Update Project module runs once per issue by default. An aggregator changes that processing pattern by combining the issue bundles into one aggregated output bundle for the selected source module and grouping context. The update module can therefore execute once for each project rather than once for every related issue.

After aggregation, the scenario should evaluate the aggregated issue data to determine whether any issue has high priority. If at least one high-priority open issue is present, map *At Risk* to the project condition; otherwise map *On Target*. Configuring the aggregation so that issues are grouped by project ID is essential: it preserves a distinct one-update result for each project while preventing issue-level updates. This also makes the evaluation reflect the complete set of open issues before the project condition is written.

Adobe documents that aggregators merge multiple bundles into a single bundle, and its bundle-processing documentation explains why modules ordinarily execute separately for each incoming bundle. See the Adobe Fusion Aggregator module reference and Adobe documentation on bundles.

QUESTION NO: 7

A scenario calls a third-party API for each bundle. Some requests fail with a temporary service-unavailable response, while later requests to the same API succeed. The failed work must not be lost, and repeated unhandled errors must not deactivate the scenario.

Which two actions should the developer take? (Choose two.)

A. Add an error-handler route to the API module and use a Break directive

B. Enable storage of incomplete executions in the scenario settings

C. Add an error-handler route to the API module and use an Ignore directive

D. Increase the maximum number of cycles in the scenario settings

ANSWER: A B

Explanation:

An error-handler route with a Break directive retains the failed bundle as an incomplete execution instead of leaving it as an unhandled failure. The scenario must also be configured to store incomplete executions so the retained work can be reviewed and resolved or retried after the third-party service recovers.

An Ignore directive would discard the failed operation and allow the scenario to continue without completing the required API call. Increasing the maximum number of cycles affects workload per execution, not recovery from a temporary service failure.