MCD-LEVEL-2$^{Q\&As}$

MuleSoft Certified Developer - Level 2 (Mule 4)

Pass Mulesoft MCD-LEVEL-2 Exam with 100% Guarantee

QUESTION 1

An organization uses CloudHub to deploy all of its applications.

How can a common-global-handler flow be configured so that it can be reused across all of the organization\\'s deployed applications?

A. Create a Mule plugin project Create a common-global-error-handler flow inside the plugin project. Use this plugin as a dependency in all Mute applications. Import that configuration file in Mute applications.

B. Create a common-global-error-handler flow in all Mule Applications Refer to it flow-ref wherever needed.

C. Create a Mule Plugin project Create a common-global-error-handler flow inside the plugin project. Use this plugin as a dependency in all Mule applications

D. Create a Mule daman project. Create a common-global-error-handler flow inside the domain project. Use this domain project as a dependency.

Correct Answer: C

To configure a common-global-handler flow that can be reused across all of the organization\\'s deployed applications, the developer should create a Mule Plugin project, create a common-global-error-handler flow inside the plugin project, and use this plugin as a dependency in all Mule applications. This way, the developer can import the common-global-error-handler flow in any application that needs it and avoid duplicating the error handling logic. References:https://docs.mulesoft.com/mule-runtime/4.3/error-handling#global-error-handler

QUESTION 2

A Mule application includes a subflow containing a Scatter.Gather scope. Within each log of the Scatter.Gatter. an HTTP connector calls a PUT endpoint to modify records in different upstream system. The subflow is called inside an Unit successful scope to retry if a transitory exception is raised.

A technical spike is being performed to increase reliability of the Mule application.

Which steps should be performed within the Mule flow above the ensure idempontent behavior?

A. Change the PUT requests inside the Scatter-Gather to POST requests

B. Ensure an error-handling flow performs corrective actions to roll back all changes if any leg of the Scatter-Gather fails

C. Remove the Put requests from the Scatter-Getter and perform them sequentially

D. None, the flow already exhibits idempotent behavior

Correct Answer: B

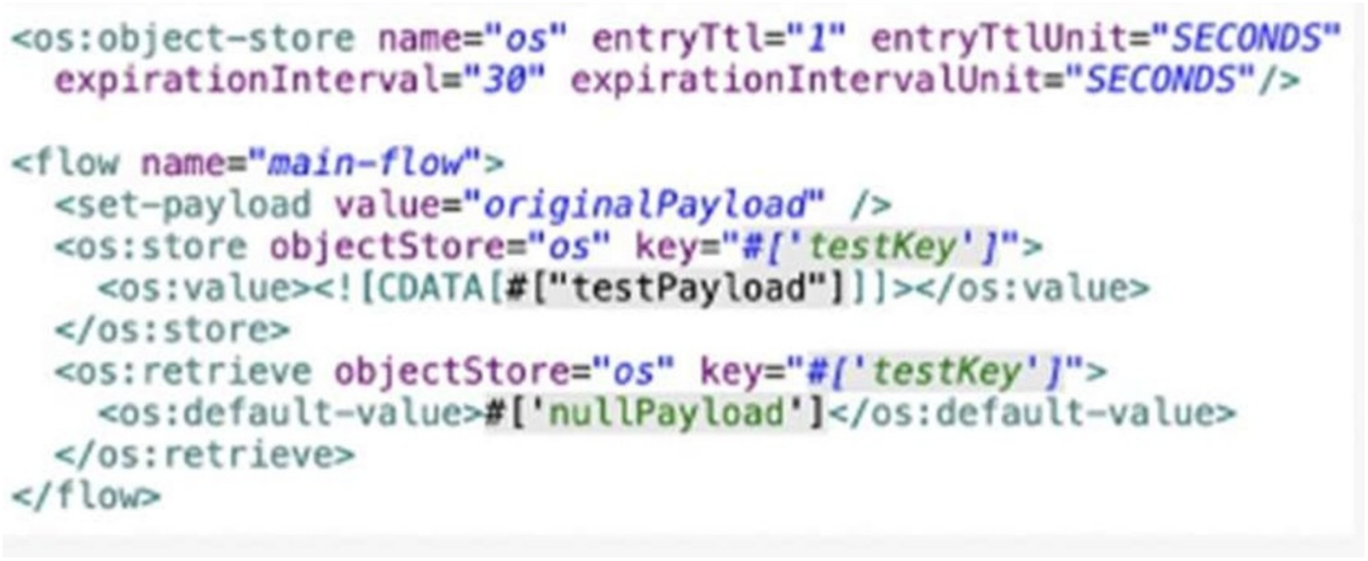
To ensure idempotent behavior within a Mule flow that contains a subflow with a Scatter-Gather scope, the developer should ensure an error-handling flow performs corrective actions to roll back all changes if any leg of the Scatter-Gather fails. Idempotency means that multiple identical requests have the same effect as a single request. Therefore, if one of the HTTP requests inside the Scatter-Gather fails, the error-handling flow should undo any changes made by other successful requests to ensure consistency and avoid partial updates. References: https://docs.mulesoft.com/mule-runtime/4.3/scatter-gather-concepthttps://docs.mulesoft.com/mule-runtime/4.3/error-handling

QUESTION 3

Refer to the exhibit.

A Mute Object Store is configured with an entry TTL of one second and an expiration interval of 30 seconds.

What is the result of the flow if processing between os\\'store and os:retrieve takes 10 seconds?


```
<os:object-store name="os" entryTtl="1" entryTtlUnit="SECONDS"
  expirationInterval="30" expirationIntervalUnit="SECONDS"/>

<flow name="main-flow">
  <set-payload value="originalPayload" />
  <os:store objectStore="os" key="#['testKey']">
    <os:value><![CDATA[#["testPayload"]]]></os:value>
  </os:store>
  <os:retrieve objectStore="os" key="#['testKey']">
    <os:default-value>#['nullPayload']</os:default-value>
  </os:retrieve>
</flow>
```

A. nullPayload

B. originalPayload

C. OS:KEY_NOT_FOUND

D. testPayload

Correct Answer: A

The result of the flow is nullPayload if processing between os:store and os:retrieve takes 10 seconds. This is because the entry TTL of the object store is one second, which means that any stored value expires after one second and is removed from the object store. The expiration interval of 30 seconds only determines how often the object store checks for expired values, but it does not affect the TTL. Therefore, when os:retrieve tries to get the value after 10 seconds, it returns nullPayload because the value has already expired and been removed.
References:https://docs.mulesoft.com/object-store/osv2-faq#how-does-the-time-to-live-work

QUESTION 4

A scatter-gather router is configured with four routes:Route A, B, C and D.

Route C false.

A. Error,errorMesage.payload.results [`2\\']

B. Payload failures[`2\\']

C. Error,errorMessage,payload.failures[`2\\']

D. Payload [`2\\']

Correct Answer: C

The result of accessing route C failure is Error,errorMessage,payload.failures[`2\\']. This is because a scatter-gather router returns an aggregated message that contains an array of results from each route and an array of failures from each route. The failures array contains error objects with information about each failed route execution. To access route C failure, which is the third route (index 2), the developer needs to use Error.errorMessage.payload.failures[`2\\'] expression.

References:https://docs.mulesoft.com/mule-runtime/4.3/scatter-gather-reference#scatter-gather-output

---

QUESTION 5

A developer deploys an API to CloudHub and applies an OAuth policy on API Manager. During testing, the API response is slow, so the developer reconfigures the API so that the out-of-the-box HTTP Caching policy is applied first, and the OAuth API policy is applied second.

What will happen when an HTTP request is received?

A. In case of a cache hit, both the OAuth and HTTP Caching policies are evaluated; then the cached response is returned to the caller

B. In case of a cache it, only the HTTP Caching policy is evaluating; then the cached response is returned to the caller

C. In case of a cache miss, only the HTTP Caching policy is evaluated; then the API retrieves the data from the API implementation, and the policy stores the data to be cached in Object Store

D. In case of a cache miss, both the OAuth and HTTP Cachingpolicies are evaluated; then the API retrieves the data from the API implementation, and the policy does not store the data in Object Store

Correct Answer: B

When an HTTP request is received and the HTTP Caching policy is applied first, it checks if there is a cached response for that request in Object Store. If there is a cache hit, meaning that a valid cached response exists, then only the HTTP Caching policy is evaluated and the cached response is returned to the caller without invoking the OAuth policy or the API implementation. If there is a cache miss, meaning that no valid cached response exists, then both the HTTP Caching policy and the OAuth policy are evaluated before invoking the API implementation.
References:https://docs.mulesoft.com/api-manager/2.x/http-caching-policy#policy-ordering

---

QUESTION 6

The Center for Enablement team published a common application as a reusable module to the central Nexus repository.

How can the common application be included in all API implementations?

A. Download the common application from Naxus and copy it to the src/main/resources folder in the API

B. Copy the common application\\'s source XML file and out it in a new flow file in the src/main/mule folder

C. Add a Maven dependency in the PCM file with multiple-plugin as

D. Add a Maven dependency in the POM file with jar as

Correct Answer: D

To include a common application as a reusable module in all API implementations, the developer should add a Maven dependency in the POM file with jar as . This way, the developer can reuse Mule code from another application by packaging it as a JAR file and adding it as a dependency in the POM file of the API implementation. The classifier element specifies that it is a JAR file. References:https://docs.mulesoft.com/mule-runtime/4.3/mmp-concept#add-a-maven-dependency-to-the-pom-file

QUESTION 7

Which configurations are required for HTTP Listener to enable mTLS authentication?

A. Set an appropriate reconnection strategy and use persistent connections for the listener

B. Set an appropriate keystore configuration and use persistent connections for the listener

C. Set an appropriate keystore and truststoreconfiguration for the listener

D. Set an appropriate truststore configuration and reconnection strategy for the listener

Correct Answer: C

To enable mTLS authentication for HTTP Listener, the developer needs to set an appropriate keystore and truststore configuration for the listener. The keystore contains the certificate and private key of the Mule application that are used to prove its identity to clients. The truststore contains the certificates of trusted clients that are allowed to access the Mule application. References:https://docs.mulesoft.com/mule-runtime/4.3/tls-configuration#mutual-authentication

QUESTION 8

The flow is invoicing a target API. The API\\'s protocol is HTTPS. The TLS configuration in the HTTP Request Configuration global element is set to None. A web client submits a request to http:localhost:8081/vehicles. If the certificate of the target API is signed by a certificate authority (CA), what is true about the HTTP Request operation when the flow executes?

A. The HTTP Request operation will succeed if the CA\\'S certificate is present in the JRE\\'s default keystore

B. The HTTP Request operation will succeed if the CA\\'s certificate is present in the JRE\\'s default truststore.

C. The HTTP Request operation will always succeed regardless of the CA

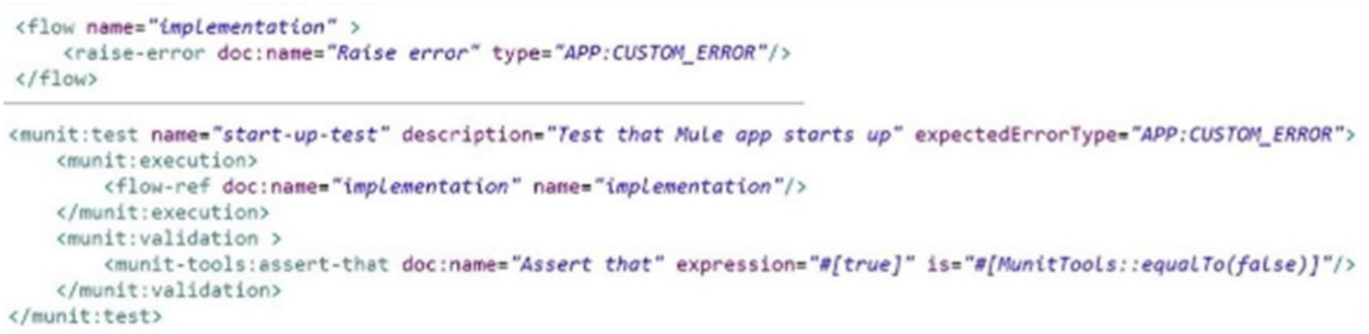
D. The HTTP Request operation will always fail regardless of the CA

Correct Answer: B

The HTTP Request operation will use the default truststore of the JRE to validate the certificate of the target API. If the CA\\'s certificate is present in the truststore, the operation will succeed. Otherwise, it will fail with a handshake exception. References:https://docs.mulesoft.com/mule-runtime/4.3/tls-configuration#tls-default

QUESTION 9

Refer to the exhibit.



```
<flow name="implementation" >
    <raise-error doc:name="Raise error" type="APP:CUSTOM_ERROR"/>
</flow>
```

```
<munit:test name="start-up-test" description="Test that Mule app starts up" expectedErrorType="APP:CUSTOM_ERROR">
    <munit:execution>
        <flow-ref doc:name="implementation" name="implementation"/>
    </munit:execution>
    <munit:validation >
        <munit-tools:assert-that doc:name="Assert that" expression="#[true]" is="#[MunitTools::equalTo(false)]"/>
    </munit:validation>
</munit:test>
```

The flow name is `\\'implementation\\'\\' with code for the MUnit test case.

When the MUnit test case is executed, what is the expected result?

A. The test case fails with an assertion error

B. The test throws an error and does not start

C. The test case fails with an unexpected error type

D. The test case passes

Correct Answer: A

Based on the code snippet and MUnit test case below, when the MUnit test case is executed, the expected result is that the test case fails with an assertion error. This is because the assert-equals processor compares two values for equality, and fails if they are not equal. In this case, the expected value is `Hello World\\', but the actual value returned by the implementation flow is `Hello Mule\\'. Therefore, the assertion fails and an error is thrown. References:https://docs.mulesoft.com/munit/2.3/assert-equals-processor

QUESTION 10

An API has been developed and deployed to CloudHub Among the policies applied to this API is an allowlist of IP addresses. A developer wants to run a test in Anypoint Studio and does not want any policies applied because their workstation is not included in the allowlist.

What must the developer do in order to run this test locally without the policies applied?

A. Create a properties file specifically for local development and set the API instance ID to a value that is not used in API Manager

B. Pass in the runtime parameter `\\'-Danpow.platform.gatekeeper=disabled\\'\\'

C. Deactivate the API in API Manager so the Autodiscovery element will not find the application when it runs in Studio

D. Run the test as-s, with no changes because the Studio runtime will not attempt to connect to API Manager

Correct Answer: A

To run a test locally without the policies applied, the developer should create a properties file specifically for local development and set the API instance ID to a value that is not used in API Manager. This way, the developer can use different configuration properties for different environments and avoid triggering API autodiscovery when running tests locally. API autodiscovery is a mechanism that associates an API implementation with its corresponding API specification and policies in API Manager based on its API instance ID. By setting this ID to a value that does not exist in API Manager, the developer can prevent API autodiscovery from finding and applying any policies to the local test.

References: https://docs.mulesoft.com/api-manager/2.x/api-auto-discovery-new-concept#configuring-api-autodiscovery
https://docs.mulesoft.com/mule-runtime/4.3/configuring-properties