100% Money Back Guarantee

Vendor: Mulesoft

Exam Code:MCD-LEVEL-2

Exam Name: MuleSoft Certified Developer - Level 2 (Mule 4)

Version: Demo

QUESTION 1

Which statement is true when working with correlation IDS?

A. The HTTP Listener regenerates correlation IDs regardless of the HTTP request

B. The Anypoint MQ Connector automatically propagates correlation IDS

C. The HTTP Listener generates correlation IDS unless a correlation ID is received in the HTTP request

D. The VM Connector does not automatically propagate correction IDs

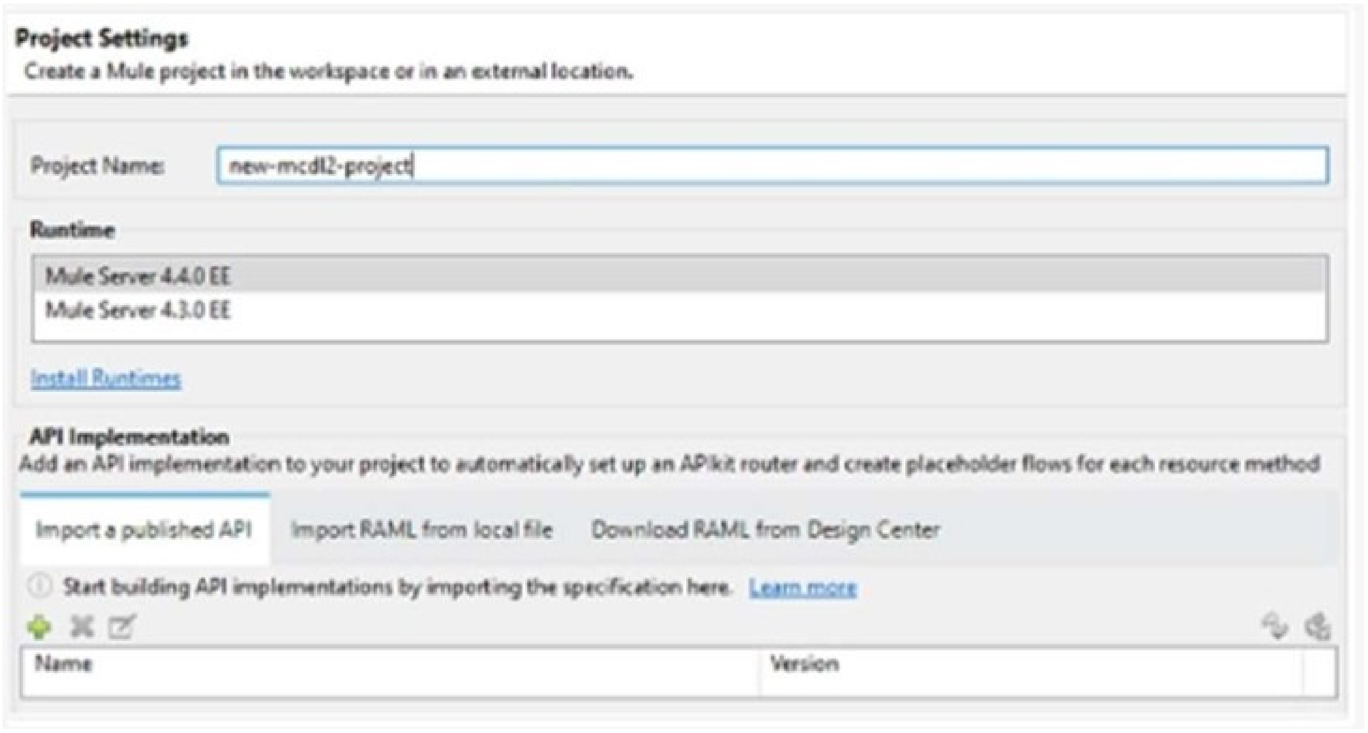
Correct Answer: C

When working with correlation IDs, the HTTP Listener generates correlation IDs unless a correlation ID is received in the HTTP request. In that case, it propagates the received correlation ID throughout the flow execution. Correlation IDs are used to track events across different flows or applications. References:https://docs.mulesoft.com/mule-runtime/4.3/about-mule-message#message-attributes

---

QUESTION 2

Refer to the exhibit.



Project Settings
Create a Mule project in the workspace or in an external location.

Project Name: new-mcdl2-project

Runtime
Mule Server 4.4.0 EE
Mule Server 4.3.0 EE

Install Runtimes

API Implementation
Add an API implementation to your project to automatically set up an APIkit router and create placeholder flows for each resource method

Import a published API
Import RAML from local file
Download RAML from Design Center
Start building API implementations by importing the specification here. Learn more
Name
Version

When creating a new project, which API implementation allows for selecting the correct API version and scaffolding the flows from the API specification?

A. Import a published API

B. Generate a local RAML from anypoint Studio

C. Download RAML from Design Center\\'

D. Import RAML from local file

Correct Answer: A

To create a new project that selects the correct API version and scaffolds the flows from the API specification, the developer should import a published API. This option allows importing an API specification that has been published to Anypoint Exchange or Design Center, and selecting a specific version of that API specification. The developer can also choose to scaffold flows based on that API specification. References:https://docs.mulesoft.com/apikit/4.x/apikit-4-new-project-task

---

QUESTION 3

Which type of cache invalidation does the Cache scope support without having to write any additional code?

A. Write-through invalidation

B. White-behind invalidation

C. Time to live

D. Notification-based invalidation

Correct Answer: C

The Cache scope supports time to live (TTL) as a cache invalidation strategy without having to write any additional code. TTL specifies how long the cached response is valid before it expires and needs to be refreshed. The Cache scope also supports custom invalidation strategies using MEL or DataWeave expressions.

References:https://docs.mulesoft.com/mule-runtime/4.3/cache-scope#cache_invalidation

---

QUESTION 4

When implementing a synchronous API where the event source is an HTTP Listener, a developer needs to return the same correlation ID back to the caller in the HTTP response header.

How can this be achieved?

A. Enable the auto-generate CorrelationID option when scaffolding the flow

B. Enable the CorrelationID checkbox in the HTTP Listener configuration

C. Configure a custom correlation policy

D. NO action is needed as the correlation ID is returned to the caller in the response header by default

Correct Answer: D

When implementing a synchronous API where the event source is an HTTP Listener, Mule automatically propagates some message attributes between flows via outbound and inbound properties. One of these attributes is correlation ID,

which is returned to the caller in the response header by default as MULE_CORRELATION_ID.

References: https://docs.mulesoft.com/mule-runtime/4.3/about-mule-message#message-attributes

---

QUESTION 5

A Mule API receives a JSON payload and updates the target system with the payload. The developer uses JSON schemas to ensure the data is valid.

How can the data be validation before posting to the target system?

A. Use a DataWeave 2.09 transform operation, and at the log of the DataWeave script, add: %dw 2.0 Import.json-moduls

B. Using the DataWeave if Else condition test the values of the payload against the examples included in the schema

C. Apply the JSON Schema policy in API Manager and reference the correct schema in the policy configuration

D. Add the JSON module dependency and add the validate-schema operation in the flow, configured to reference the schema

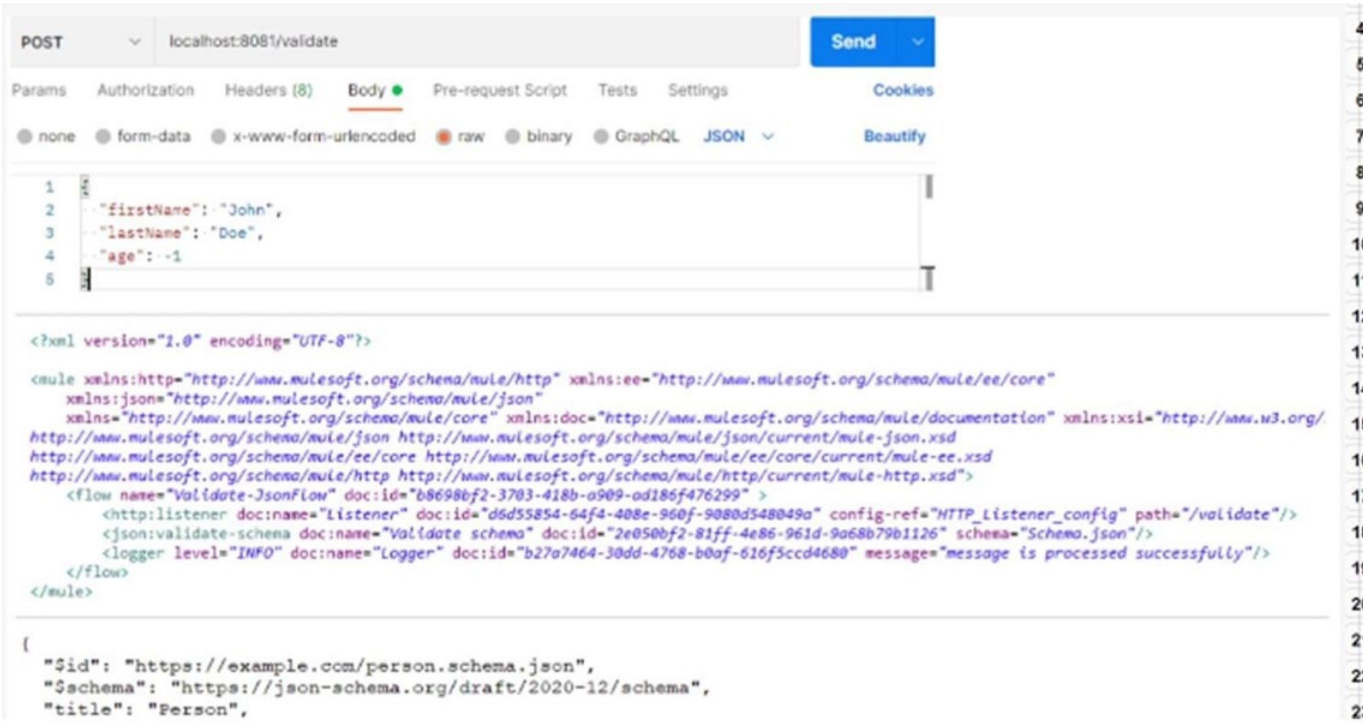
Correct Answer: D

To validate the data before posting to the target system, the developer should add the JSON module dependency and add the validate-schema operation in the flow, configured to reference the schema. The JSON module provides a validate-schema operation that validates a JSON payload against a JSON schema and throws an error if the payload is invalid. References:https://docs.mulesoft.com/json-module/1.1/json-validate-schema

---

QUESTION 6

Refer to the exhibit.



localhost:8081/validate

```
{
  "firstName": "John",
  "lastName": "Doe",
  "age": -1
}
```

```
<?xml version="1.0" encoding="UTF-8"?>

<mule xmlns:http="http://www.mulesoft.org/schema/mule/http" xmlns:ee="http://www.mulesoft.org/schema/mule/ee/core"
    xmlns:json="http://www.mulesoft.org/schema/mule/json"
    xmlns="http://www.mulesoft.org/schema/mule/core" xmlns:doc="http://www.mulesoft.org/schema/mule/documentation" xmlns:xsi="http://www.w3.org/
http://www.mulesoft.org/schema/mule/json http://www.mulesoft.org/schema/mule/json/current/mule-json.xsd
http://www.mulesoft.org/schema/mule/ee/core http://www.mulesoft.org/schema/mule/ee/core/current/mule-ee.xsd
http://www.mulesoft.org/schema/mule/http http://www.mulesoft.org/schema/mule/http/current/mule-http.xsd">
    <flow name="Validate-JsonFlow" doc:id="b8698bf2-3703-418b-a909-ad186f476299" >
        <http:listener doc:name="Listener" doc:id="d6d55854-64f4-408e-960f-9080d548049a" config-ref="HTTP_Listener_config" path="/validate"/>
        <json:validate-schema doc:name="Validate schema" doc:id="2e050bf2-81ff-4e86-961d-9a68b79b1126" schema="Schema.json"/>
        <logger level="INFO" doc:name="Logger" doc:id="b27a7464-30dd-4768-b0af-616f5ccd4680" message="message is processed successfully"/>
    </flow>
</mule>
```

```
{
  "$id": "https://example.com/person.schema.json",
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "title": "Person",
```

Based on the code snippet, schema,json file, and payload below, what is the outcome of the given code snippet when a request is sent with the payload?

A. The Mule flow will execute successfully with status code 200, and the response will be the JSON sent in request

B. The Mule flow will execute successfully with status code 204

C. The Mule flow will throw the exception `JSON:SCHEMA_NOT_HONOURED

D. The Mule flow will execute successfully with status code 200m andaresponse will display the message `\\' Age in years which must equal to or greater than zero.\\'\\'

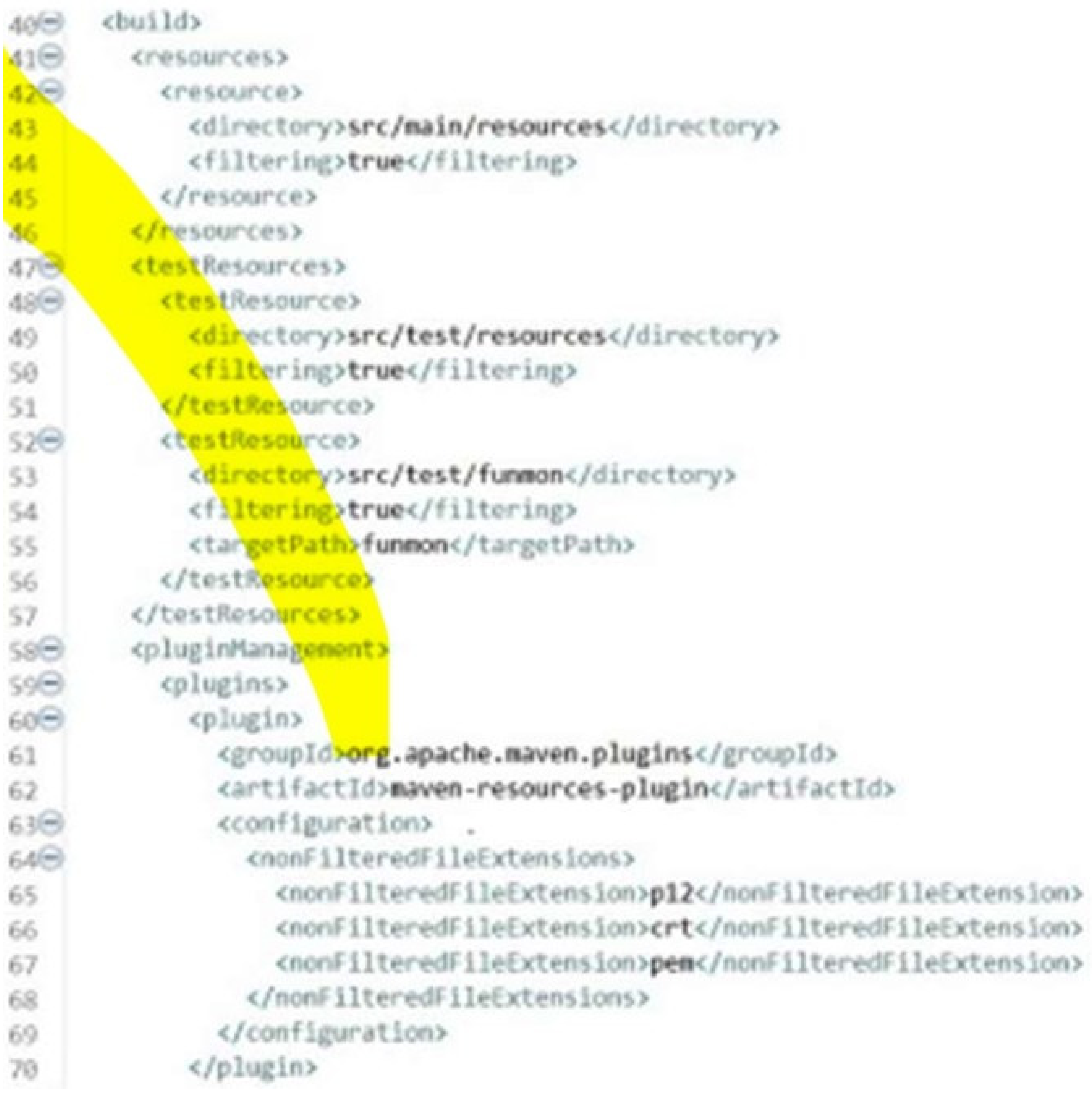
Correct Answer: C

Based on the code snippet, schema.json file, and payload below, the outcome of the given code snippet when a request is sent with the payload is that the Mule flow will throw the exception `JSON:SCHEMA_NOT_HONOURED\\'. This is because the payload does not conform to the schema.json file, which specifies that age must be a number greater than or equal to zero. The payload has age as a string with a negative value, which violates the schema. Therefore, the validate-schema operation throws an error with type `JSON:SCHEMA_NOT_HONOURED\\'.
References:https://docs.mulesoft.com/json-module/1.1/json-validate-schema

---

QUESTION 7

Refer to the exhibit.

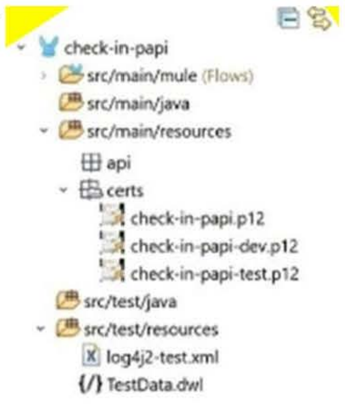


```
  <build>
    <resources>
      <resource>
        <directory>src/main/resources</directory>
        <filtering>true</filtering>
      </resource>
    </resources>
    <testResources>
      <testResource>
        <directory>src/test/resources</directory>
        <filtering>true</filtering>
      </testResource>
      <testResource>
        <directory>src/test/funmon</directory>
        <filtering>true</filtering>
        <targetPath>funmon</targetPath>
      </testResource>
    </testResources>
    <pluginManagement>
      <plugins>
        <plugin>
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-resources-plugin</artifactId>
          <configuration>
            <nonFilteredFileExtensions>
              <nonFilteredFileExtension>p12</nonFilteredFileExtension>
              <nonFilteredFileExtension>crt</nonFilteredFileExtension>
              <nonFilteredFileExtension>pem</nonFilteredFileExtension>
            </nonFilteredFileExtensions>
          </configuration>
        </plugin>
```

A Mule application pom.xml configures the Maven Resources plugin to exclude parsing binary files in the project\\'s src/main/resources/certs directory.

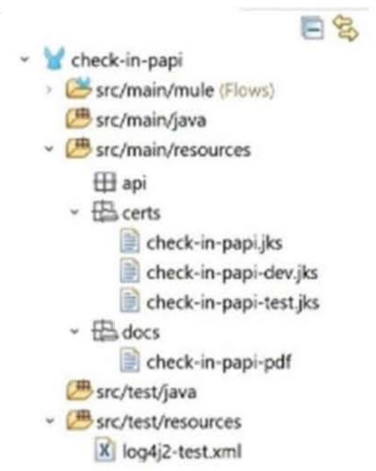
Which configuration of this plugin achieves a successful build?

A.



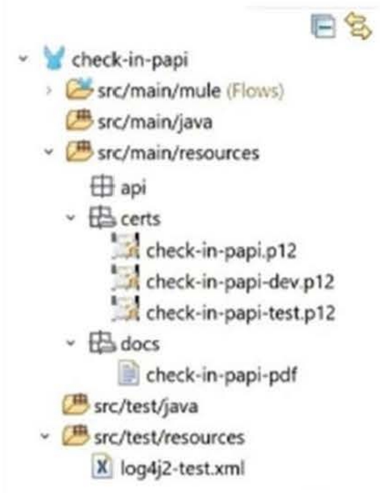
```
check-in-papi
  src/main/mule (Flows)
  src/main/java
  src/main/resources
    api
    certs
      check-in-papi.p12
      check-in-papi-dev.p12
      check-in-papi-test.p12
  src/test/java
  src/test/resources
    log4j2-test.xml
    TestData.dwl
```

B.



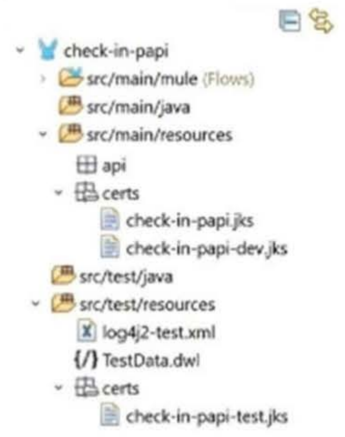
```
check-in-papi
  src/main/mule (Flows)
  src/main/java
  src/main/resources
    api
    certs
      check-in-papi.jks
      check-in-papi-dev.jks
      check-in-papi-test.jks
    docs
      check-in-papi-pdf
  src/test/java
  src/test/resources
    log4j2-test.xml
```

C.



```
check-in-papi
  src/main/mule (Flows)
  src/main/java
  src/main/resources
    api
    certs
      check-in-papi.p12
      check-in-papi-dev.p12
      check-in-papi-test.p12
    docs
      check-in-papi-pdf
  src/test/java
  src/test/resources
    log4j2-test.xml
```

D.



```
check-in-papi
  src/main/mule (Flows)
  src/main/java
  src/main/resources
    api
    certs
      check-in-papi.jks
      check-in-papi-dev.jks
  src/test/java
  src/test/resources
    log4j2-test.xml
    TestData.dwl
    certs
      check-in-papi-test.jks
```

A. Option A

B. Option B

C. Option C

D. Option D

Correct Answer: C

To configure the Maven Resources plugin to exclude parsing binary files in the project\\'s src/main/resources/certs directory, option C should be used. This option specifies that any files with .cer or .jks extensions under the certs directory should be excluded from filtering. Filtering is a process of replacing placeholders with actual values in resource files during the build process. Binary files should not be filtered because they may become corrupted or unusable. References: https://maven.apache.org/plugins/maven-resources-plugin/examples/filter.html https://maven.apache.org/plugins/maven-resources-plugin/examples/include-exclude.html

---

QUESTION 8

A Mule application defines as SSL/TLS keystore properly `tis,keystore.keyPassword\\'\\' as secure.

How canthis property be referenced to access its value within the application?

A. #{secure::tiskeystore,keyPassowrd}

B. ${secure::tiskeystore,keyPassowrd}

C. ${secure::tiskeystore,keyPassowrd}

D. p{secure::tiskeystore,keyPassowrd}

Correct Answer: B

secure::tiskeystore,keyPassowrdShortExplanationofCorrectAnswerOnly:Toreferenceasecur epropertyvaluewithintheapplication,thedeveloperneedstousethesyntax{secure::}. In this case, the property name is tiskeystore,keyPassword, so the correct syntax is ${secure::tiskeystore,keyPassowrd}. References: https://docs.mulesoft.com/mule-runtime/4.3/secure-configuration-properties#referencing-secure-properties

---

QUESTION 9

Which statement is true when using XML SDK for creating custom message processors?

A. Properties are fields defined by an end user of the XML SDK component and serve as a global configuration for the entire Mule project in which they are used

B. An XML SDK provides both inbound and outbound operations

C. Operations can be reused in recursive calls

D. All operations are public

Correct Answer: D

When using XML SDK for creating custom message processors, all operations are public by default and can be used by any Mule application that imports them. There is no way to make an operation private or protected in XML SDK. References:https://docs.mulesoft.com/mule-sdk/1.1/xml-sdk#operations

---

QUESTION 10

Which command is used to convert a JKS keystore to PKCS12?

A. Keytool-importkeystore -srckeystore keystorep12-srcstoretype PKCS12 -destkeystore keystore.jks -deststoretype JKS

B. Keytool-importkeystore -srckeystore keystore p12-srcstoretype JKS -destkeystore keystore.p12 -deststoretype PKCS12

C. Keytool-importkeystore -srckeystore keystore jks-srcstoretype JKS -destkeystore keystore.p13 -deststoretype PKCS12

D. Keytool-importkeystore -srckeystore keystore jks-srcstoretype PKCS12 -destkeystore keystore.p12 -deststoretype JKS

Correct Answer: B

To convert a JKS keystore to PKCS12, the developer needs to use the keytool-importkeystore command with the following options:-srckeystore keystore.jks-srcstoretype JKS-destkeystore keystore.p12-deststoretype PKCS12. This command imports all entries from a source JKS keystore (keystore.jks) into a destination PKCS12 keystore (keystore.p12). References:https://docs.oracle.com/en/java/javase/11/tools/keytool.html#GUID-5990A2E4-78E3-47B7-AE75-6D1826259549

QUESTION 11

Which pattern should be used to invoke multiple HTTP APIs in parallel and roll back failed requests in sequence?

A. A database as a transactional outbox and an Until Successful router to retry any requests

B. A Parallel for Each scope with each HTTP request wrapped in a Try scope

C. Scatter-Gather as central Saga orchestrator for all API request with compensating actions for failing routes

D. VM queues as a reliability pattern with error handlers to roll back any requests

Correct Answer: C

To invoke multiple HTTP APIs in parallel and roll back failed requests in sequence, the developer should use a Scatter-Gather router as a central Saga orchestrator for all API requests with compensating actions for failing routes. A Scatter-Gather router executes multiple routes concurrently and aggregates the results. A Saga orchestrator coordinates a series of actions across different services and handles failures by executing compensating actions. Therefore, using a Scatter-Gather router as a Saga orchestrator allows invoking multiple HTTP APIs in parallel and rolling back any failed requests in sequence. References: https://docs.mulesoft.com/mule-runtime/4.3/scatter-gather-concept https://docs.mulesoft.com/mule-runtime/4.3/saga

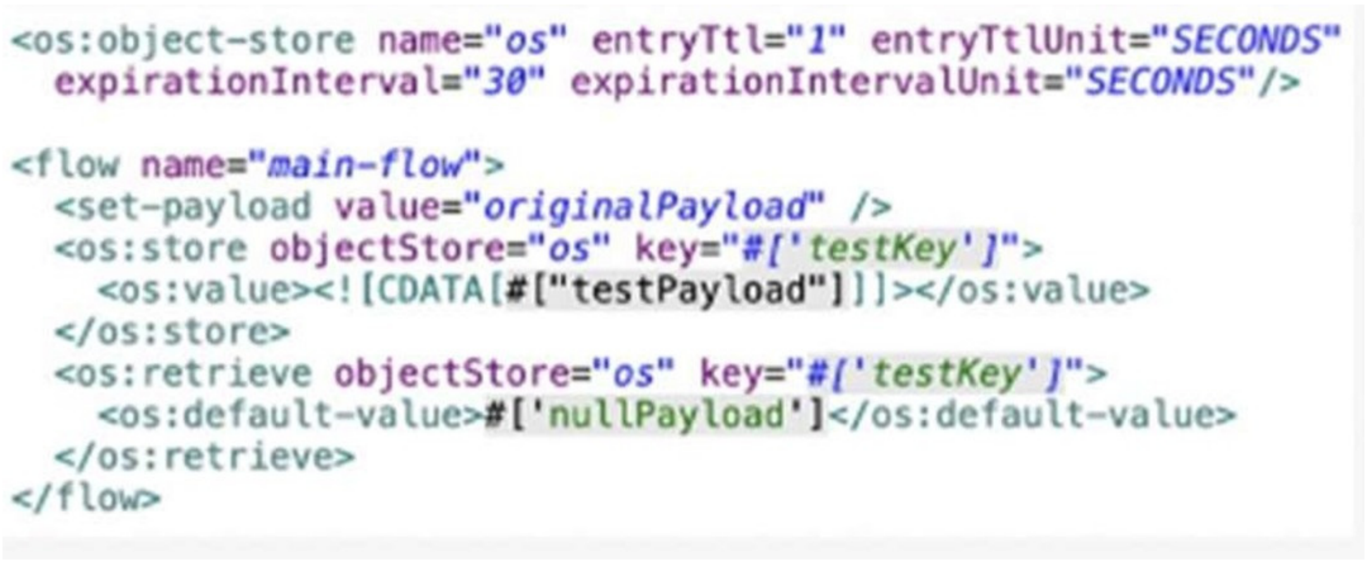
---

QUESTION 12

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