# Get all the Information About MuleSoft MCD-Level-2 Exam 2023 Practice Test Questions [Q26-Q47

**Get all the Information About MuleSoft MCD-Level-2 Exam 2023 Practice Test Questions Check Real MuleSoft MCD-Level-2 Exam Question for Free (2023) NEW QUESTION 26**

Mule application A is deployed to CloudHub and is using Object Store v2. Mute application B is also deployed to CloudHub.

Which approach can Mule application B use to remove values from Mule application A&#8217;S Object Store?
* Object Store v2 REST API
* CloudHub Connector
* Object Store Connector
* CloudHub REST API
Explanation

To remove values from Mule application A&#8217;s Object Store v2, Mule application B can use Object Store v2 REST API. This API allows performing operations on Object Store v2 resources using HTTP methods, such as GET, POST, PUT, and DELETE. Mule application B can use the DELETE method to remove values from Mule application A&#8217;s Object Store v2 by specifying the object store ID and the key of the value to delete.

References: https://docs.mulesoft.com/object-store/osv2-apis

**NEW QUESTION 27**

A Mule implementation uses a HTTP Request within an Unit Successful scope to connect to an API.

How should a permanent error response like HTTP:UNAUTHORIZED be handle inside Until Successful to reduce latency?
* Configure retrying until a MULERETRY_EXHAUSTED error is raised or the API responds back with a successful response.
* In Until Successful configuration, set the retry count to 1 for error type HTTP: UNAUTHORIZED.
* Put the HTTP Request inside a try scope in Unit Successful.

In the error handler, use On Error Continue to catch permanent errors like HTTP UNAUTHORIZED.
* Put the HTTP Request inside a try scope in Unit Successful.

In the error handler, use On Error Propagate to catch permanent errors like HTTP UNAUTHORIZED.
Explanation

To handle a permanent error response like HTTP:UNAUTHORIZED inside Until Successful, the developer should put the HTTP Request inside a try scope in Unit Successful, and use On Error Continue to catch permanent errors like HTTP UNAUTHORIZED in the error handler. This way, the developer can avoid retrying requests that will always fail due to a permanent error, and reduce latency. On Error Continue allows the flow to continue processing after handling the error. References:

https://docs.mulesoft.com/mule-runtime/4.3/until-successful-scope

https://docs.mulesoft.com/mule-runtime/4.3/on-error-continue-concept

**NEW QUESTION 28**

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?
* Download the common application from Naxus and copy it to the src/main/resources folder in the API
* Copy the common application&#8217;s source XML file and out it in a new flow file in the src/main/mule folder
* Add a Maven dependency in the PCM file with multiple-plugin as <classifier>
* Add a Maven dependency in the POM file with jar as <classifier>
Explanation

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 <classifier>. 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

**NEW QUESTION 29**

A Flight Management System publishes gate change notification events whenever a flight&#8217;s arrival gate changes. Other systems, including Baggage Handler System. Inflight Catering System and Passenger Notifications System, must each asynchronously receive the same gate change notification to process the event according.

Which configuration is required in Anypoint MQ to achieve this publish/subscribe model?
* Publish each client subscribe directly to the exchange.

Have each client subscribe directly to the queue.
* Publish the gate change notification to an Anypoint MC queue

Have each client subscribe directly to the queue
* Publish the gate change notification to an Anypoint MQ queue.

Create different anypoint MQ exchange meant for each of the other subscribing systems Bind the queue with each of the exchanges
* Publish the gate change notification to an Anypoint MQ exchanhe.

Create different Anypoint MQ queues meant for each of the other subscribing systems.

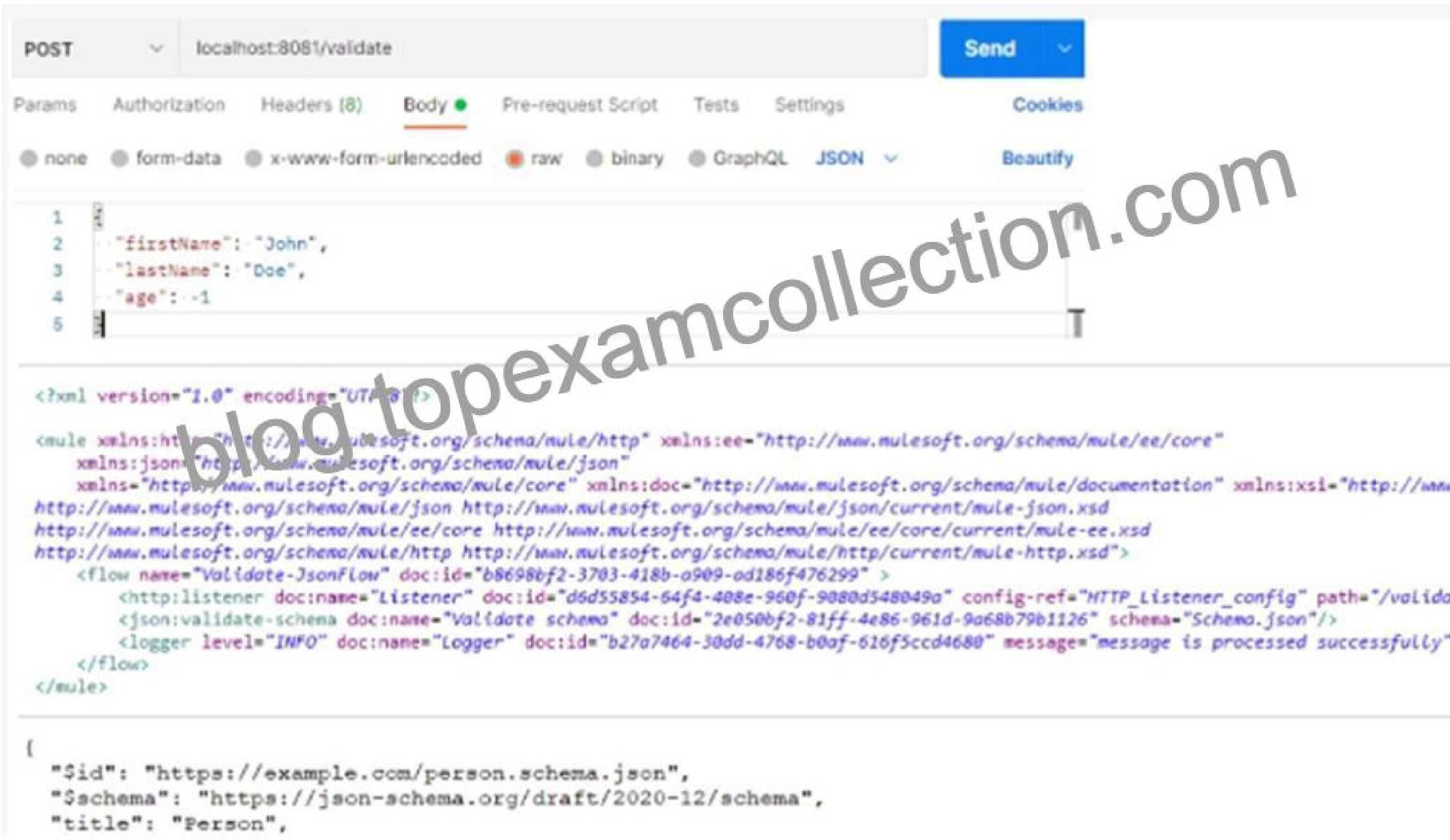
Bind the exchange with each of the queues.
Explanation

To achieve a publish/subscribe model using Anypoint MQ, where each system receives the same gate change notification event, the developer should publish the gate change notification to an Anypoint MQ exchange, create different Anypoint MQ queues meant for each of the other subscribing systems, and bind the exchange with each of the queues. An exchange is a message routing agent that can send messages to different queues based on predefined criteria. By binding an exchange with multiple queues, each queue receives a copy of every message sent to that exchange. Therefore, each system can subscribe to its own queue and receive every gate change notification event. References:

https://docs.mulesoft.com/anypoint-mq/3.x/anypoint-mq-exchanges

**NEW QUESTION 30**

Refer to the exhibit.



```
{
  "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
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="/valida
        <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?

* The Mule flow will execute successfully with status code 200, and the response will be the JSON sent in request
* The Mule flow will execute successfully with status code 204
* The Mule flow will throw the exception &#8216;JSON:SCHEMA_NOT_HONOURED
* The Mule flow will execute successfully with status code 200m and a response will display the message

&#8221; Age in years which must equal to or greater than zero.&#8221;
Explanation

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

&#8216;JSON:SCHEMA_NOT_HONOURED&#8217;. 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

&#8216;JSON:SCHEMA_NOT_HONOURED&#8217;. References:

https://docs.mulesoft.com/json-module/1.1/json-validate-schema

**NEW QUESTION 31**

A company has been using CI/CD. Its developers use Maven to handle build and deployment activities.

What is the correct sequence of activities that takes place during the Maven build and deployment?
* Initialize, validate, compute, test, package, verify, install, deploy
* Validate, initialize, compile, package, test, install, verify, verify, deploy
* Validate, initialize, compile, test package, verify, install, deploy
* Validation, initialize, compile, test, package, install verify, deploy

Explanation

The correct sequence of activities that takes place during the Maven build and deployment is validate, initialize, compile, test package, verify, install, deploy. These are Maven lifecycle phases that define a sequence of goals to execute during a build process. Each phase represents a stage in the build lifecycle and can have zero or more goals bound to it. References:

https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html

**NEW QUESTION 32**

A healthcare portal needs to validate the token that it sends to a Mule API. The developer plans to implement a custom policy using the HTTP Policy Transform Extension to match the token received in the header from the heathcare portal.

Which files does the developer need to create in order to package the custom policy?
* Deployable ZIP file, YAML configuration file
* JSON properties file, YAML configuration file
* JSON properties file, XML template file
* XML template file, YAML configuration file

Explanation

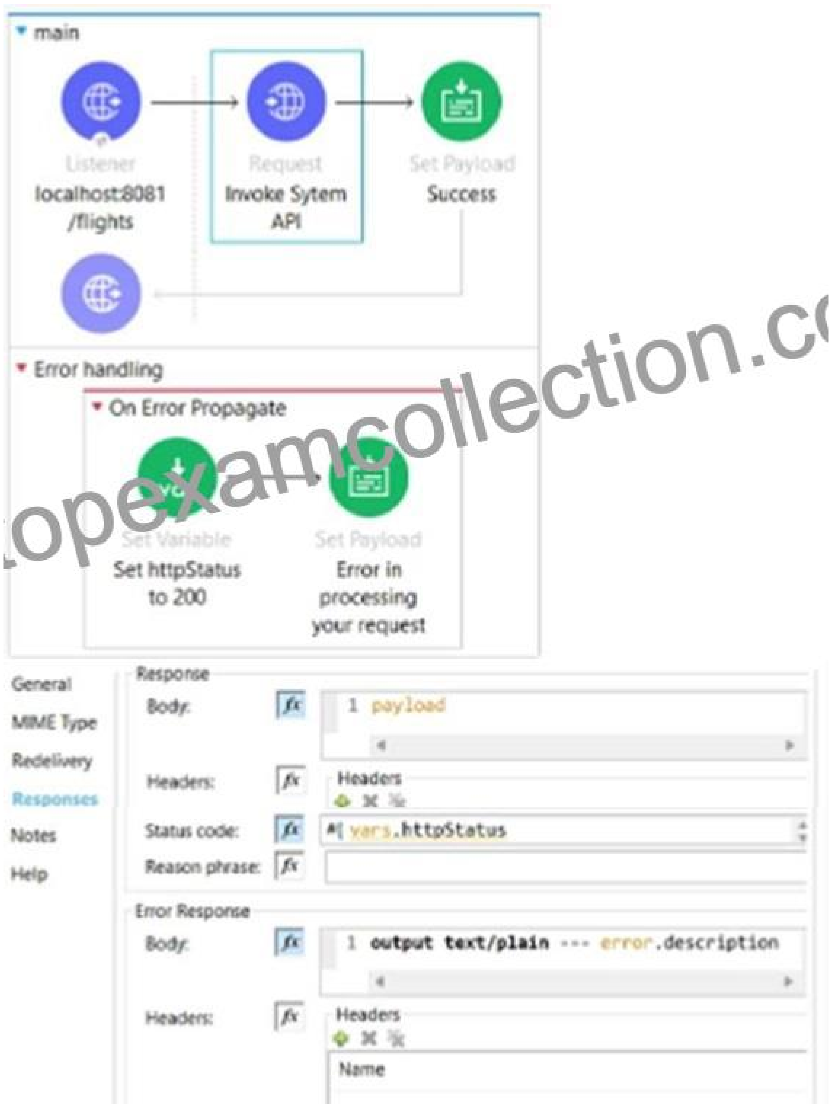
To package a custom policy using the HTTP Policy Transform Extension, the developer needs to create an XML template file and a YAML configuration file. The XML template file defines the policy logic using Mule components and placeholders for user-defined properties. The YAML configuration file defines the metadata of the policy, such as its name, description, category, parameters, and dependencies. References:

https://docs.mulesoft.com/api-manager/2.x/http-policy-transform#packaging-the-policy

**NEW QUESTION 33**

The HTTP Request operation raises an HTTP CONNECTIVITY error.

Which HTTP status code and body are returned to the web client?



```
payload
```

Status code: `#[ vars.httpStatus`

```
output text/plain --- error.description
```

* HTTP Status Code:200.

Body &#8216;Error in processing your request
* HTTP Status Code:500.

Body &#8216;The HTTP CONNECTIVITY Error description
* HTTP Status Code:500.

Body &#8216;Error in processing your request
* HTTP Status Code:500.

Body &#8216;Error in processing your request
Explanation

When the HTTP Request operation raises an HTTP CONNECTIVITY error, it triggers an on-error-continue handler that sets a payload with &#8216;Error in processing your request&#8217;. Since no status code is explicitly set in this handler, it defaults to 500 (INTERNAL SERVER ERROR). Therefore, the web client receives an HTTP response with status code 500 and body &#8216;Error in processing your request&#8217;. References:

https://docs.mulesoft.com/mule-runtime/4.3/error-handling#on-error-continue

**NEW QUESTION 34**

A custom policy needs to be developed to intercept all cutbound HTTP requests made by Mule applications.

Which XML element must be used to intercept outbound HTTP requests?
* It is not possible to intercept outgoing HTTP requests, only inbound requests
* http-policy:source
* htt-policy:operation
* http-policy:processor

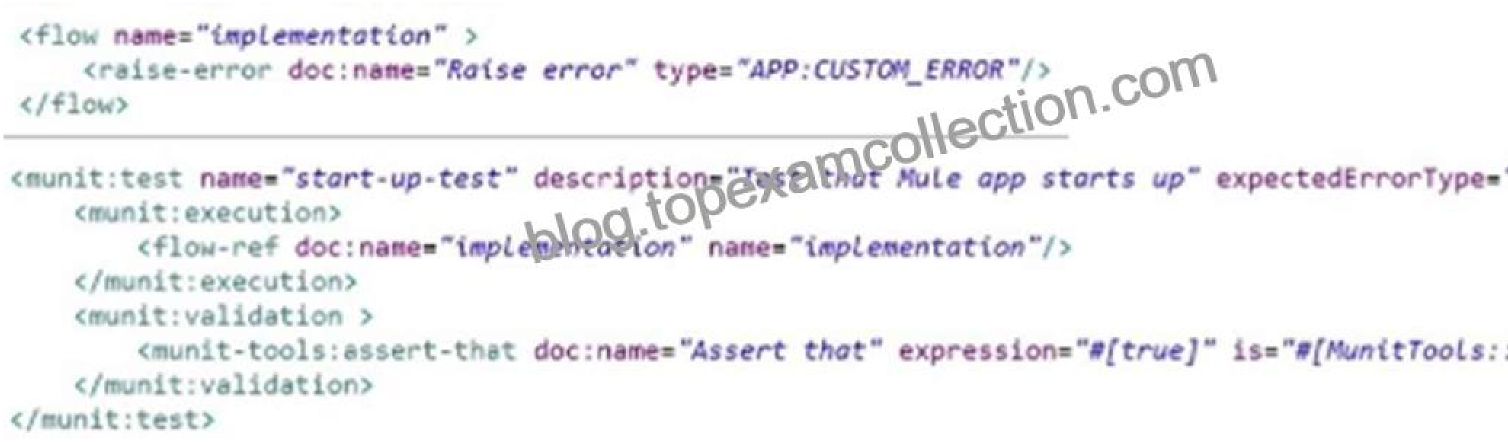
Explanation

The http-policy:processor element is used to intercept outbound HTTP requests made by Mule applications. It allows customizing the request before it is sent to the target API and modifying the response after it is received from the target API. References:

https://docs.mulesoft.com/api-manager/2.x/policy-mule4-custom-policy#policy-xml-file

**NEW QUESTION 35**

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="
    <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::
    </munit:validation>
</munit:test>
```

The flow name is &#8221;implementation&#8221; with code for the MUnit test case.

When the MUnit test case is executed, what is the expected result?
* The test case fails with an assertion error
* The test throws an error and does not start
* The test case fails with an unexpected error type
* The test case passes

Explanation

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 &#8216;Hello World&#8217;, but the actual value returned by the implementation flow is

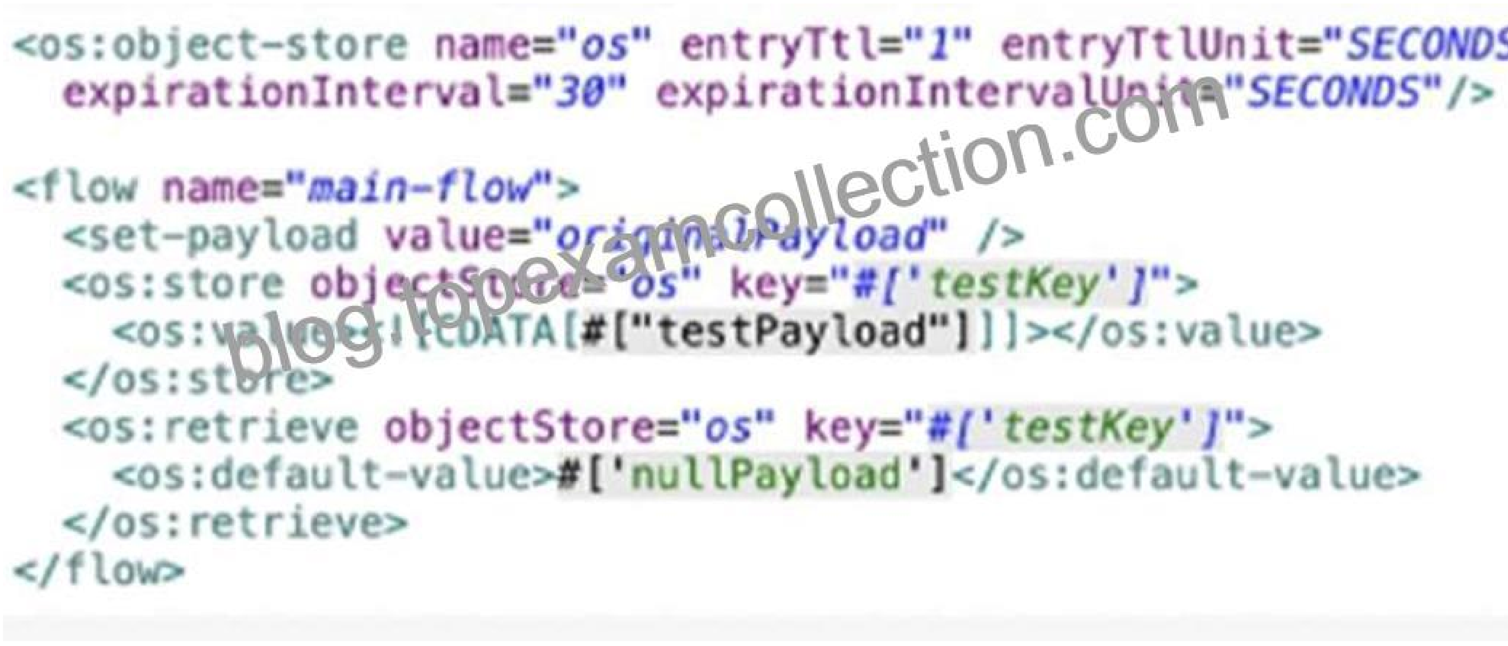
&#8216;Hello Mule&#8217;. Therefore, the assertion fails and an error is thrown. References:
https://docs.mulesoft.com/munit/2.3/assert-equals-processor

**NEW QUESTION 36**

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&#8217;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>
```

* nullPayload
* originalPayload
* OS:KEY_NOT_FOUND
* testPayload

Explanation

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

**NEW QUESTION 37**

A company deploys 10 public APIs to CloudHub. Each API has its individual health endpoint defined. The platform operation team wants to configure API Functional Monitoring to monitor the health of the APIs periodically while minimizing operational overhead and cost.

How should API Functional Monitoring be configured?
* From one public location with each API in its own schedule
* From one private location with all 10 APIs in a single schedule
* From one public location with all 10 APIs in a single schedule
* From 10 public locations with each API in its own schedule
Explanation

To configure API Functional Monitoring to monitor the health of 10 public APIs periodically while minimizing operational overhead and cost, the developer should use one public location with all 10 APIs in a single schedule. A public location is a worker that runs in a CloudHub shared environment, which is cheaper and easier to maintain than a private location. A single schedule allows running all 10 APIs tests at the same time and frequency, which reduces complexity and resource consumption. References:

https://docs.mulesoft.com/functional-monitoring/fm-create-monitor#create-a-monitor

**NEW QUESTION 38**

What is the MuleSoft recommended method to encrypt sensitive property data?
* The encryption key and sensitive data should be different for each environment
* The encryption key should be identical for all environments
* The encryption key should be identical for all environments and the sensitive data should be different for each environment
* The encryption key should be different for each environment and the sensitive data should be the same for all environments
Explanation

The MuleSoft recommended method to encrypt sensitive property data is to use the Secure Properties Tool that comes with Anypoint Studio. This tool allows encrypting properties files with a secret key and then decrypting them at runtime using the same key. The encryption key and sensitive data should be different for each environment to ensure security and avoid accidental exposure of sensitive data. References:

https://docs.mulesoft.com/mule-runtime/4.3/secure-configuration-properties

**NEW QUESTION 39**

A company with MuleSoft Titanium develops a Salesforce System API using MuleSoft out-of-the-box Salesforce Connector and deploys the API to CloudHub.

Which steps provide the average number of requests and average response time of the Salesforce Connector?
* Access Anypoint Monitoring&#8217;s built-in dashboard. Select a resource.

Locate the information under the Connectors tab.
* Access Anypoint Monitoring&#8217;s built-in dashboard

Seclect a resource.

Create a custom dashboard to retrieve the information.
* Access Anypoint Monitoring built-in dashboard.

Select a resource.

Locate the information under Log Manager < Raw Data.
* Change the API Implementation to capture the information in the log.

Retrieve the information from the log file.
Explanation

To get the average number of requests and average response time of the Salesforce Connector, the developer should access Anypoint Monitoring&#8217;s built-in dashboard, select a resource (such as an application or an API), and locate the information under the Connectors tab. The Connectors tab shows metrics for each connector used by the resource, such as average requests per minute, average response time, and failures. References:

https://docs.mulesoft.com/monitoring/built-in-dashboard-reference

**NEW QUESTION 40**

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?
* In case of a cache hit, both the OAuth and HTTP Caching policies are evaluated; then the cached response is returned to the caller
* In case of a cache it, only the HTTP Caching policy is evaluating; then the cached response is returned to the caller
* 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
* In case of a cache miss, both the OAuth and HTTP Caching policies are evaluated; then the API retrieves the data from the API implementation, and the policy does not store the data in Object Store
Explanation

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

**NEW QUESTION 41**

In a Mule project, Flow-1 contains a flow-ref to Flow-2 depends on data from Flow-1 to execute successfully.

Which action ensures the test suites and test cases written for Flow-1 and Flow-2 will execute successfully?
* Use &#8216;After Test Case&#8217; to produce the data needed from Flow-1 test cases to pass to Flow-2 test cases
* Use &#8221;Before Test Case&#8221; To collect data from Flow-1 test cases before running Flow-2 test cases
* Chain together the test suites and test cases for Flow-1 and Flow-2
* Use &#8221;Set Event to pass the input that is needed, and keep the test cases for Flow-1 and Flow-2 independent
Explanation

To ensure the test suites and test cases written for Flow-1 and Flow-2 will execute successfully, the developer should use a Set Event processor to pass the input that is needed by Flow-2, and keep the test cases for Flow-1 and Flow-2 independent. This way, the developer can isolate the testing of each flow and avoid coupling them together. References:
https://docs.mulesoft.com/munit/2.3/munit-test-flow

**NEW QUESTION 42**

A Mule application defines as SSL/TLS keystore properly &#8217;tis,keystore.keyPassword&#8221; as secure.

How can this property be referenced to access its value within the application?
* #{secure::tiskeystore,keyPassowrd}
* ${secure::tiskeystore,keyPassowrd}
* ${secure::tiskeystore,keyPassowrd}
* p{secure::tiskeystore,keyPassowrd}
Explanation

secure::tiskeystore,keyPassowrdShortExplanationofCorrectAnswerOnly:Toreferenceasecurepropertyvaluewithin 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

**NEW QUESTION 43**

Which statement is true about using mutual TLS to secure an application?
* Mutual TLS requires a hardware security module to be used
* Mutual TLS authenticates the identity of the server before the identity of the client
* Mutual TLS ensures only authorized end users are allowed to access an endpoint
* Mutual TLS increases the encryption strength versus server-side TLS alone
Explanation

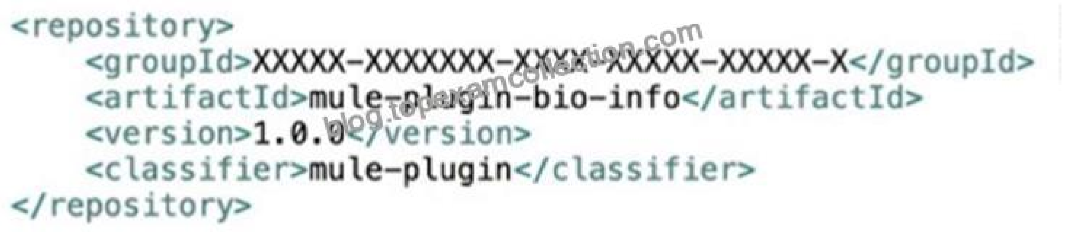
Mutual TLS (mTLS) is an extension of TLS that requires both parties (client and server) to present their certificates to each other during the handshake process. This way, both parties can verify each other&#8217;s identity and establish a secure connection. The authentication of the server happens before the authentication of the client, as the server sends its certificate first and then requests the client&#8217;s certificate. References:

https://docs.mulesoft.com/mule-runtime/4.3/tls-configuration#mutual-authentication

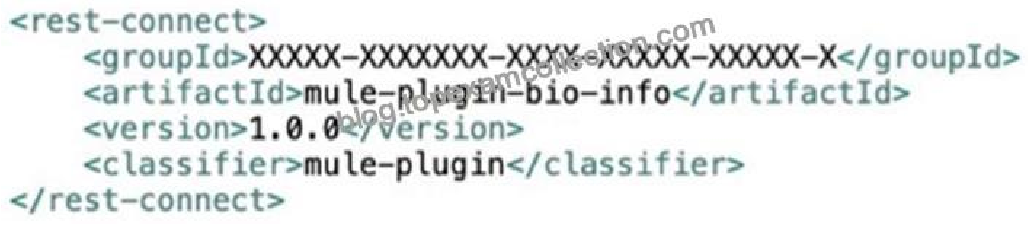
**NEW QUESTION 44**

Refer to the exhibits.

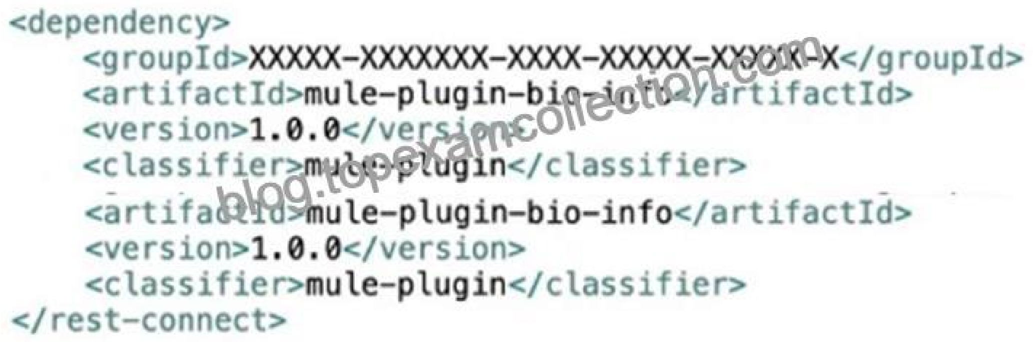
Bioinfo System API is implemented and published to Anypoint Exchange. A developer wants to invoke this API using its REST Connector.

What should be added to the POM?
*


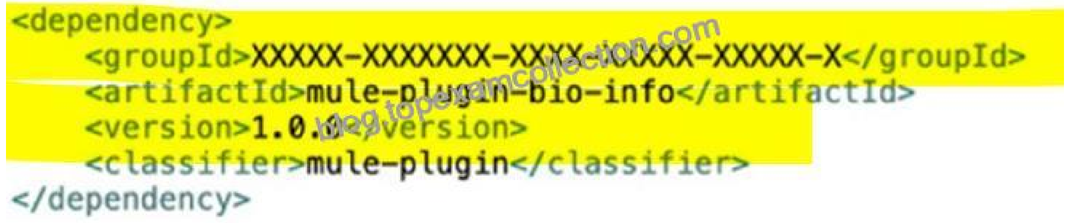
```
<repository>
    <groupId>XXXXX-XXXXXXX-XXXX-XXXXX-XXXXX-X</groupId>
    <artifactId>mule-plugin-bio-info</artifactId>
    <version>1.0.0</version>
    <classifier>mule-plugin</classifier>
</repository>
```

* 

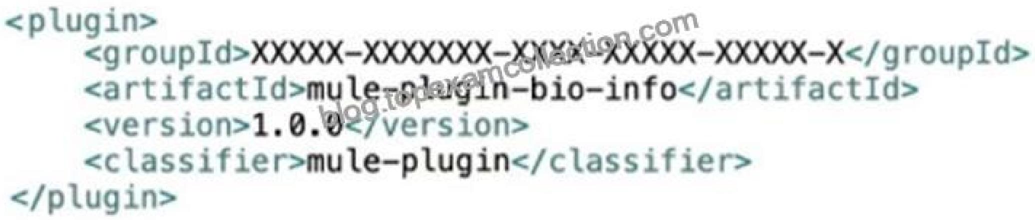
```
<rest-connect>
    <groupId>XXXXX-XXXXXXX-XXXX-XXXXX-XXXXX-X</groupId>
    <artifactId>mule-plugin-bio-info</artifactId>
    <version>1.0.0</version>
    <classifier>mule-plugin</classifier>
</rest-connect>
```

* 

```
<dependency>
    <groupId>XXXXX-XXXXXXX-XXXX-XXXXX-XXXXX-X</groupId>
    <artifactId>mule-plugin-bio-info</artifactId>
    <version>1.0.0</version>
    <classifier>mule-plugin</classifier>
    <artifactId>mule-plugin-bio-info</artifactId>
    <version>1.0.0</version>
    <classifier>mule-plugin</classifier>
</rest-connect>
```

* 

```
<dependency>
    <groupId>XXXXX-XXXXXXX-XXXX-XXXXX-XXXXX-X</groupId>
    <artifactId>mule-plugin-bio-info</artifactId>
    <version>1.0.0</version>
    <classifier>mule-plugin</classifier>
</dependency>
```

* 

```
<plugin>
    <groupId>XXXXX-XXXXXXX-XXXX-XXXXX-XXXXX-X</groupId>
    <artifactId>mule-plugin-bio-info</artifactId>
    <version>1.0.0</version>
    <classifier>mule-plugin</classifier>
</plugin>
```

Explanation

To invoke Bioinfo System API using its REST Connector, option E should be added to the pom.xml file. This option adds a dependency for Bioinfo System API REST Connector with its group ID, artifact ID, version, classifier, and type. The classifier specifies that it is a REST Connector (raml-client), and the type specifies that it is a Mule plugin (mule-plugin). References:

https://docs.mulesoft.com/apikit/4.x/apikit-4-generate-from-rest-api-task#add-the-api-dependency-to-the-pom-fil

**NEW QUESTION 45**

A mule application exposes and API for creating payments. An Operations team wants to ensure that the Payment API is up and running at all times in production.

Which approach should be used to test that the payment API is working in production?
* Create a health check endpoint that listens on a separate port and uses a separate HTTP Listener configuration from the API
* Configure the application to send health data to an external system
* Create a health check endpoint that reuses the same port number and HTTP Listener configuration as the API itself
* Monitor the Payment API directly sending real customer payment data

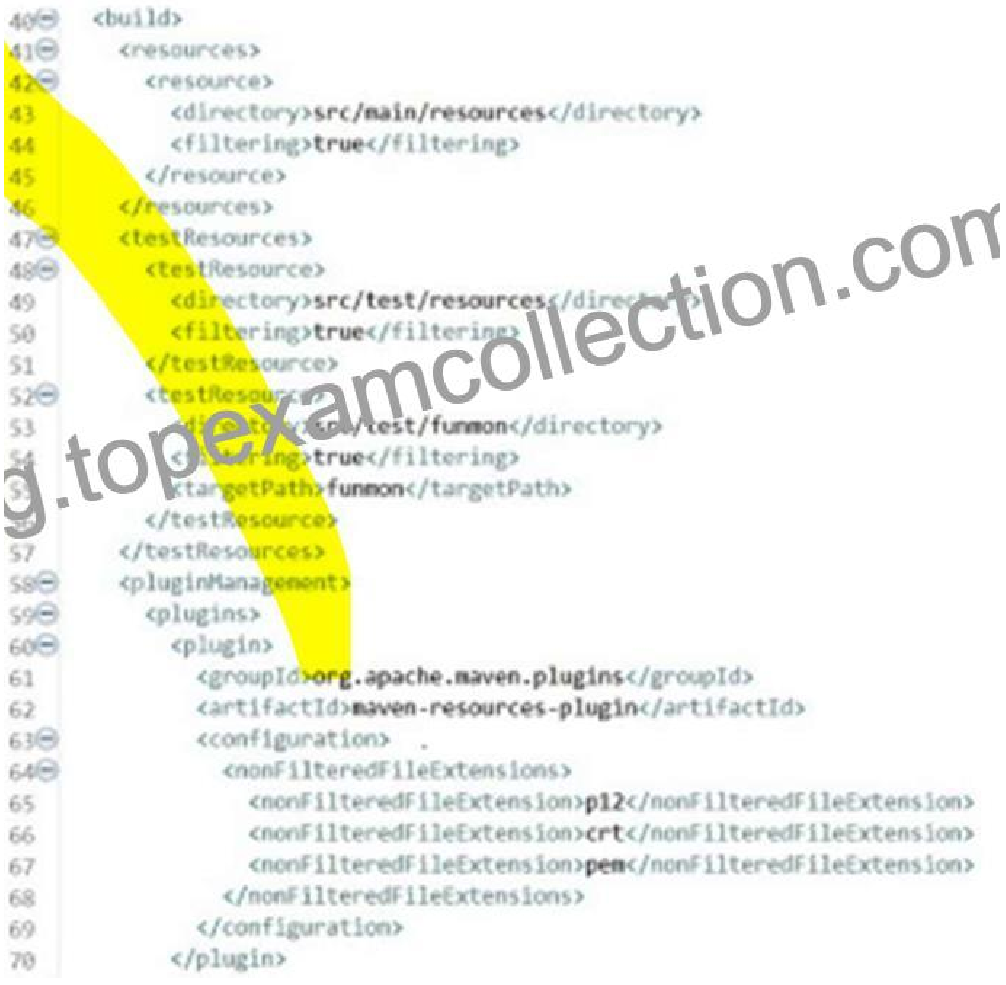
Explanation

To test that the payment API is working in production, the developer should create a health check endpoint that listens on a separate port and uses a separate HTTP Listener configuration from the API. This way, the developer can isolate the health check endpoint from the API traffic and avoid affecting the performance or availability of the API. The health check endpoint should return a simple response that indicates the status of the API, such as OK or ERROR. References:

https://docs.mulesoft.com/api-functional-monitoring/afm-create-monitor#create-a-monitor

**NEW QUESTION 46**

Refer to the exhibit.

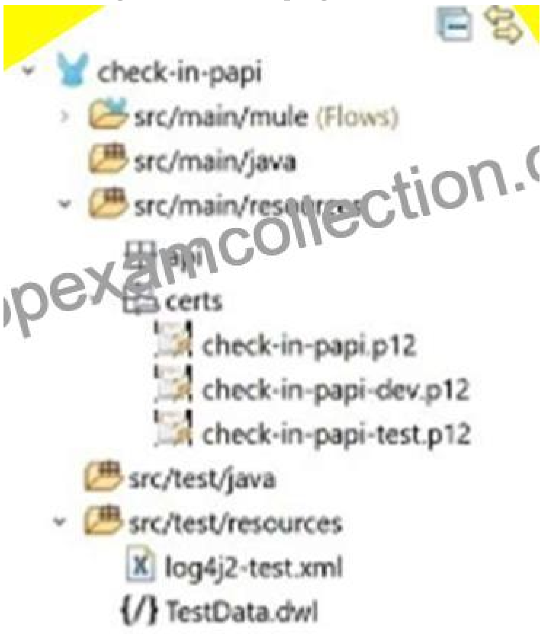


```
<build>
  <resources>
    <resource>
      <directory>src/main/resources</directory>
      <filtering>true</filtering>
    </resource>
  </resources>
  <testResources>
    <testResource>
      <directory>src/test/resources</directory>
      <filtering>true</filtering>
    </testResource>
    <testResource>
      <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&#8217;s src/main/resources/certs directory.

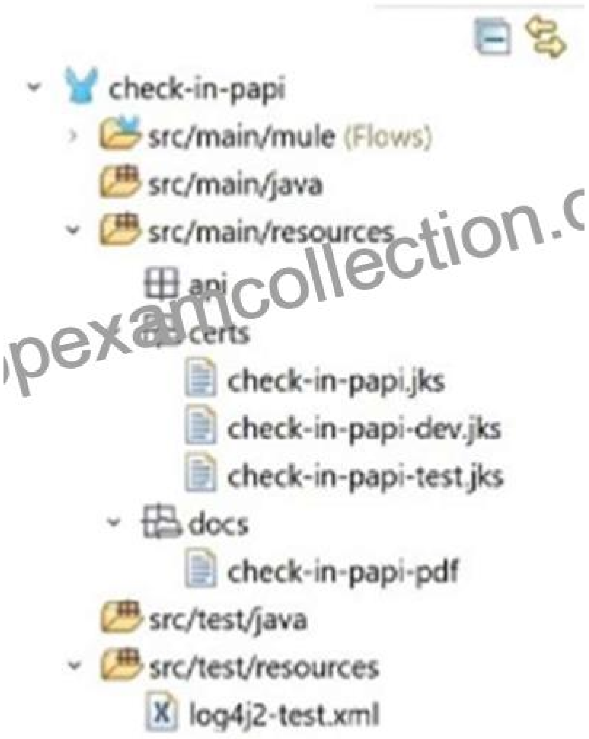
Which configuration of this plugin achieves a successful build?

*



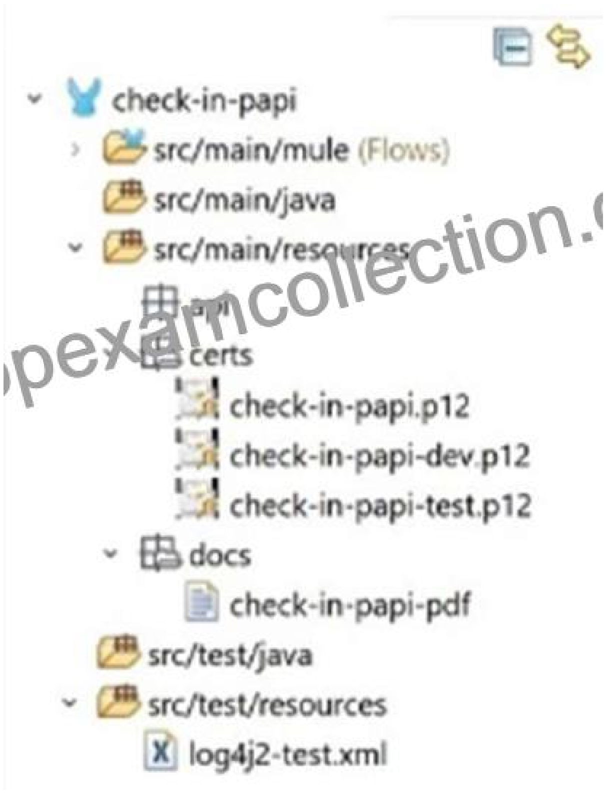
```
check-in-papi
  src/main/mule (Flows)
  src/main/java
  src/main/resources
    api
    certs
      check-in-papi.p12
      check-in-papi-dev.p12
      check-in-papi-test.p12
  src/test/java
  src/test/resources
    log4j2-test.xml
    TestData.dwl
```

*



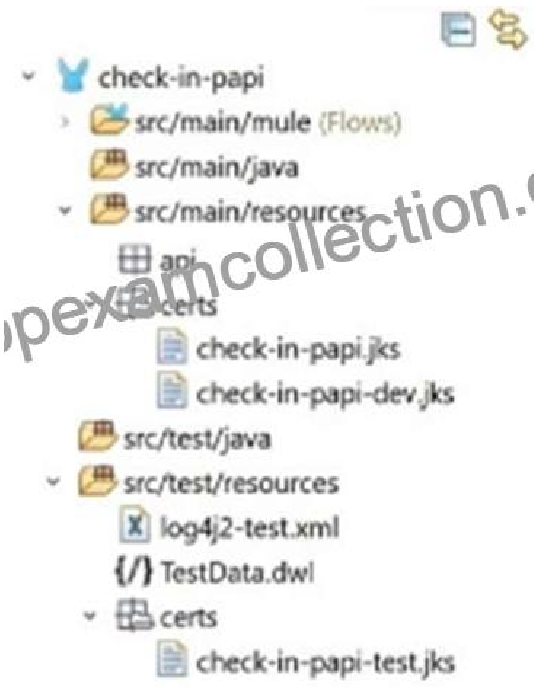
```
check-in-papi
  src/main/mule (Flows)
  src/main/java
  src/main/resources
    api
    certs
      check-in-papi.jks
      check-in-papi-dev.jks
      check-in-papi-test.jks
    docs
      check-in-papi-pdf
  src/test/java
  src/test/resources
    log4j2-test.xml
```

*



```
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
```

*



```
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
```

Explanation

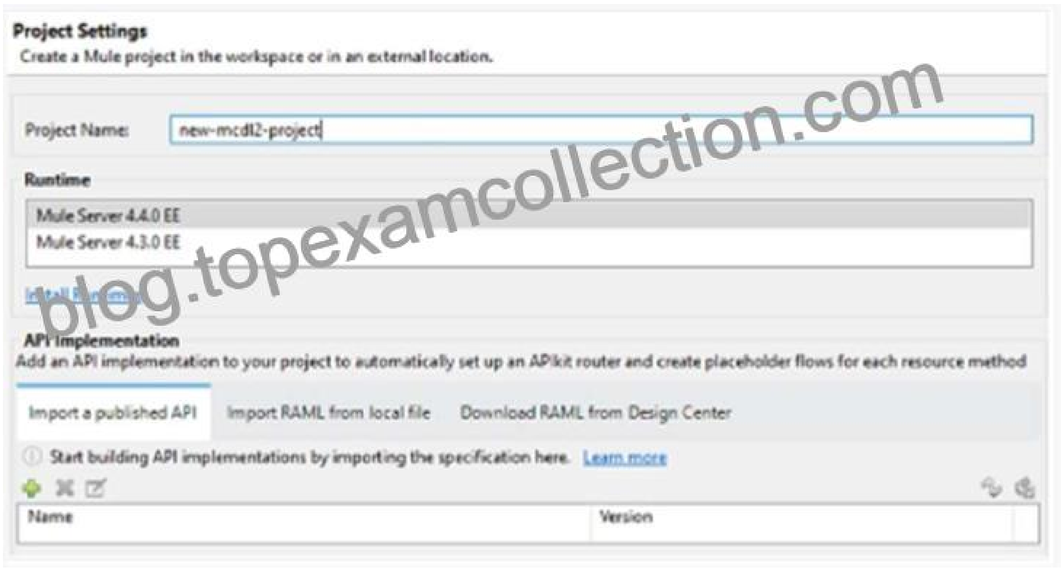
To configure the Maven Resources plugin to exclude parsing binary files in the project&#8217;s src/main/resources/certs directory, option C should be used. This option specifies that any files with .cer or .jks extensions under the certs directory should be excluded from filtering. Filtering is a process 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

**NEW QUESTION 47**

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?
* Import a published API
* Generate a local RAML from anypoint Studio
* Download RAML from Design Center&#8217;
* Import RAML from local file
Explanation

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