MCD-LEVEL-2 Q&As

MuleSoft Certified Developer - Level 2 (Mule 4)

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

Free Download Real Questions & Answers **PDF** and **VCE** file from:

https://www.geekcert.com/mcd-level-2.html

100% Passing Guarantee
100% Money Back Assurance

Following Questions and Answers are all new published by Mulesoft Official Exam Center

- **Instant Download** After Purchase
- **100% Money Back** Guarantee
- **365 Days** Free Update
- **800,000+** Satisfied Customers

QUESTION 1

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?

A. Configure retrying until a MULERETRY_EXHAUSTED error is raised or the API responds back with a successful response.

B. In Until Successful configuration, set the retry count to 1 for error type HTTP: UNAUTHORIZED.

C. 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.

D. 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.

Correct Answer: C

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

QUESTION 2

A Mule application need to invoice an API hosted by an external system to initiate a process. The external API takes anywhere between one minute and 24 hours to compute its process.

Which implementation should be used to get response data from the external API after it completes processing?

A. Use an HTTP Connector to invoke the API and wait for a response

B. Use a Scheduler to check for a response every minute

C. Use an HTTP Connector inside Async scope to invoice the API and wait for a response D. Expose an HTTP callback API in Mule and register it with the external system

Correct Answer: D

To get response data from the external API after it completes processing, the developer should expose an HTTP callback API in Mule and register it with the external system. This way, the external API can invoke the callback API with the response data when it is ready, instead of making the Mule application wait for a long time or poll for a response repeatedly. References:https://docs.mulesoft.com/mule-runtime/4.3/http-listener-ref#callback

QUESTION 3

A heathcare customer wants to use hospital system data, which includes code that was developed using legacy tools

and methods. The customer has created reusable Java libraries in order to read the data from the system.

What is the most effective way to develop an API retrieve the data from the hospital system?

A. Refer to JAR files in the code

B. Include the libraries writes deploying the code into the runtime

C. Create the Java code in your project and invoice the data from the code

D. Install libraries in a local repository and refer to it in the pm.xml file

Correct Answer: D

To develop an API that retrieves data from a hospital system using reusable Java libraries, the developer should install libraries in a local repository and refer to it in the pom.xml file. This way, the developer can use Maven to manage dependencies and invoke Java code from Muleapplications using Java Module operations. References: https://docs.mulesoft.com/mule-runtime/4.3/java-module-reference#add-the-java-module-to-your-projecthttps://docs.mulesoft.com/mule-runtime/4.3/java-module-reference#invoke-java-code

QUESTION 4

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>
```

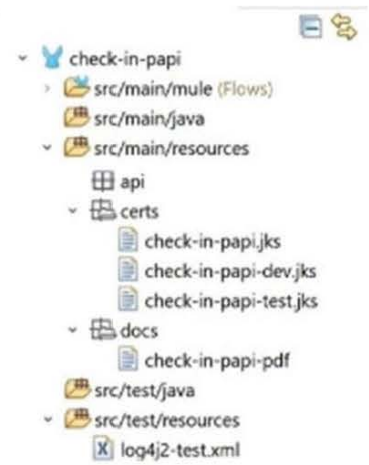
A Mule application pom.xml configures the Maven Resources plugin to exclude parsing binary files in the project\\'s src/main/resources/certs directory.

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

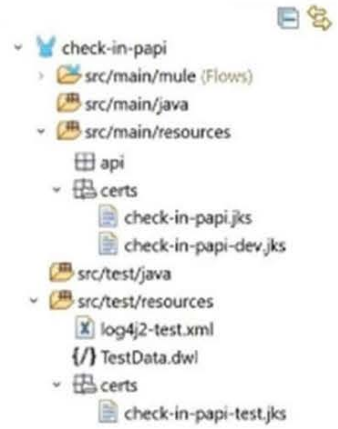
B.


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 5

A developer is working on a project that requires encrypting all data before sending it to a backend application. To accomplish this, the developer will use PGP encryption in the Mule 4 Cryptography module.

What is required to encrypt the data before sending it to the backend application?

A. The application needs to configure HTTPS TLS context information to encrypt the data

B. The application needs to both the private and public keys to encrypt the data

C. The application needs the public key from the backend service to encrypt the data

D. The application needs the private key from the backend service to encrypt the data

Correct Answer: C

To encrypt the data before sending it to the backend application using PGP encryption, the application needs the public key from the backend service. PGP encryption uses a public-key cryptography system, which means that each party has a pair of keys: a public key and a private key. The public key is used to encrypt data, and the private key is used to decrypt data. Therefore, to encrypt data for a specific recipient (the backend service), the application needs to use the recipient\\'s public key. The recipient can then use its own private key to decrypt the data. References:https://docs.mulesoft.com/mule-runtime/4.3/cryptography-pgp