Question 1

A developer has created the first version of an API designed for business partners to work commodity prices.

What should developer do to allow more than one major version of the same API to be exposed by the implementation?

Options:

A- In Design Center, open the RAML and modify each operation to include the major version number

B- In Anypoint Studio, generate scaffolding from the RAML, and the modify the <http:listerner> in the generated flows to include a parameter to replace the version number

C- In Design Center, open the RAML and modify baseUn to include a variable that indicates the version number

D- In Anypoint Studio, generate scaffolding from the RAML, and then modify the flow names generated by APIKit to include a variable with the major version number

Answer:

C

Explanation:

To allow more than one major version of the same API to be exposed by the implementation, the developer should modify the baseUri property in the RAML file to include a variable that indicates the version number. The baseUri property defines the base URL of the API and can include variables that are replaced with actual values when mocking or deploying the API. By using a variable for the version number, the developer can expose different versions of the API using different base URLs and avoid conflicts or confusion. Reference: https://docs.mulesoft.com/api-designer/design-modify-raml-specs#baseuri https://docs.mulesoft.com/api-manager/2.x/api-versioning

Question 2

Question Type: MultipleChoice

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?

Options:

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

Answer:

B

Explanation:

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. Reference: 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

Question 3

Question Type: **MultipleChoice**

Refer to the exhibit.

What is the result if "Insecure" selected as part of the HTTP Listener configuration?

Global Element Properties

HTTP Listener config

Configuration element for a HttpListener

General   Notes   Help

Name: HTTP_Listener_config

Connection

General   TLS   Advanced

TLS Configuration   Edit inline   ⬍

Trust Store Configuration

Path: [   ]

Password: [   ]   ◯ Show password

Type: [⌄]

Algorithm: [   ]

☑ ⚠ Insecure

⑦   Test Connection...   Cancel   OK

Options:

A- The HTTP Listener will trust any certificate presented by the HTTP client

B- The HTTP Lister will accept any unauthenticated request

C- The HTTP listener will only accept HTTP requests

D- Mutual TLS authentication will be enabled between this HTTP Listener and an HTTP client

Answer:

C

Explanation:

Based on the exhibit below, if 'Insecure' is selected as part of the HTTP Listener configuration, the HTTP listener will only accept HTTP requests. This means that no TLS context will be configured for this listener and no encryption or authentication will be applied to incoming requests. The protocol attribute of this listener will be set to HTTP instead of HTTPS. Reference: https://docs.mulesoft.com/http-connector/1.6/http-listener-ref#insecure

Question 4

A Mule application for processing orders must log the order ID for every log message output.

What is a best practice to enrich every log message with the order ID?

Options:

A- Use flow variables within every logger processor to log the order ID

B- Set a flow variable and edit the log4/2.xml file to output the variable as part of the message pattern

C- Create a custom XML SDK component to wrap the logger processor and automatically add the order ID within the connector

D- Use the Tracing module to set logging variables with a Mapped Diagnostic Context

Answer:

D

Explanation:

To enrich every log message with the order ID, the developer should use the Tracing module to set logging variables with a Mapped Diagnostic Context (MDC). The Tracing module allows adding custom key-value pairs to log messages using MDC variables. The developer can use Set Logging Variables operation to set the order ID as an MDC variable and then use it in any logger processor within the same thread or event. Reference: https://docs.mulesoft.com/tracing-module/1.0/tracing-module-reference#set-logging-variables

Question 5

A Mule application contain two policies Policy A and Policy A has order1, and Policy B has order 2. Policy A Policy B, and a flow are defined by he configuration below.

```
<http-policy:proxy name="policy-A">
  <http-policy:source>
    <A1/>
    <http-policy:execute-next/>
    <A2/>
  </http-policy:source>
</http-policy:proxy>

<http-policy:proxy name="policy-B">
  <http-policy:source>
    <B1/>
    <http-policy:execute-next/>
    <B2/>
  </http-policy:source>
</http-policy:proxy>

<flow name="flow">
  <http:listener/>
  <F1/>
</flow>
```

When a HTTP request arrives at the Mule application's endpoint, what will be the execution order?

Options:

A- A1, B1, F1, B2, A2

B- B1, A1, F1, A2, B2

C- F1, A1, B1, B2, A2

D- F1, B1, A1, A2, B2

Answer:

A

Explanation:

Based on the configuration below, when a HTTP request arrives at the Mule application's endpoint, the execution order will be A1, B1, F1, B2, A2. This is because policies are executed before and after the API implementation flow according to their order attribute. Policy A has order 1, which means it is executed first before Policy B, which has order 2. The flow is executed after both policies are executed before the flow. Then, Policy B is executed after the flow before Policy A is executed after the flow. Reference: https://docs.mulesoft.com/api-manager/2.x/policies-policy-order

Question 6

Question Type: MultipleChoice

Refer to the exhibit.

What required changes can be made to give a partial successful response in case the United Airlines API returns with a timeout?
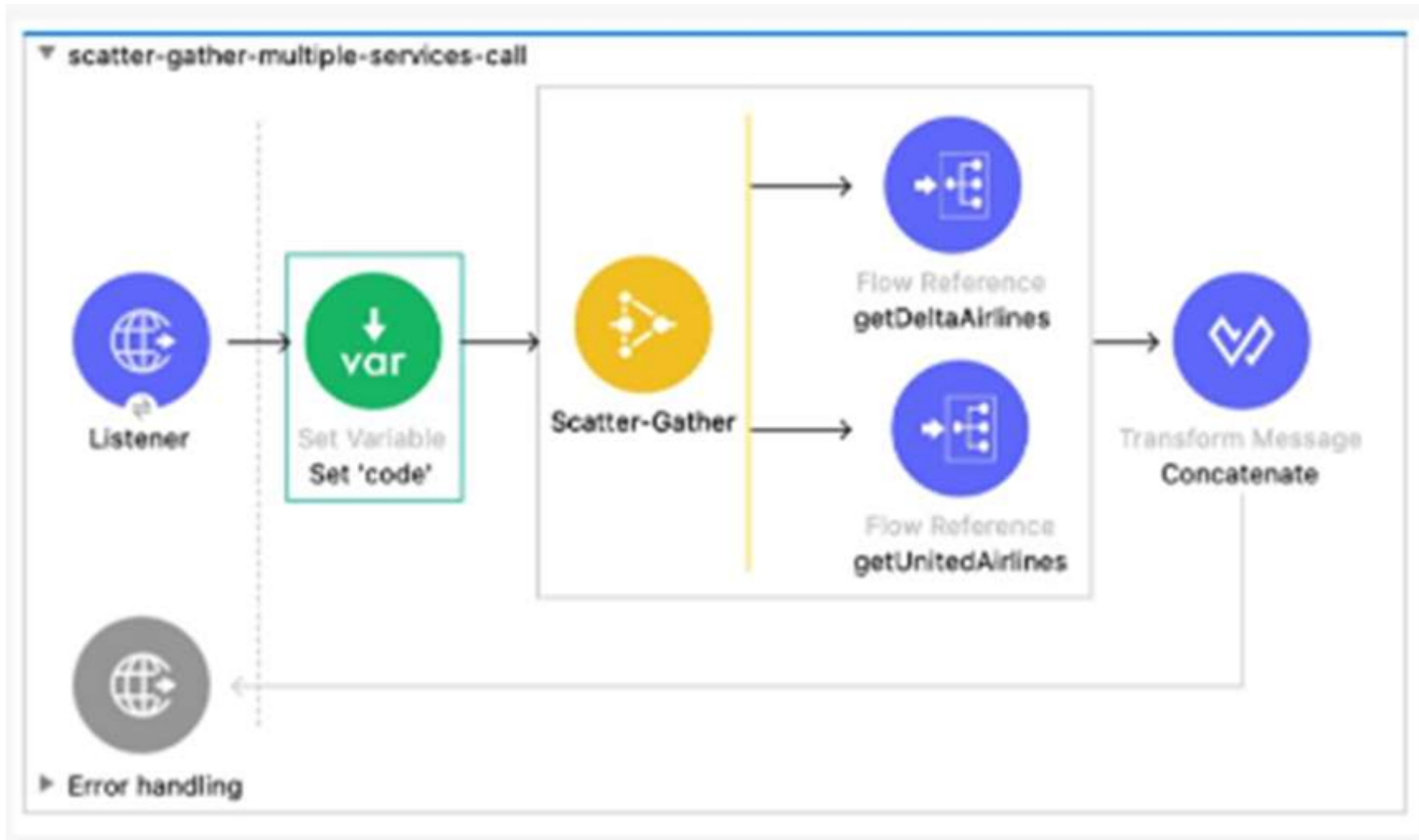
Options:

A- Add a Scatter-gather component inside a Try scope.

Set the payload to a default value 'Error' inside the error handler using the On Error Propagate scope.

B- Add Flow Reference components inside a Try scope.

Set the payload to a default value" insider the error handler using the ON Error Continue scope

C- Add Flow Reference components inside a Try scope

Set the payload to a default value " inside the error handler using the On Error Propagate scope

D- Add a Scatter-Gather component inside a Try scope.

Set the payload to a default value 'Error" inside the error handler using the On Error Continue scope.

Answer:

D

Explanation:

To give a partial successful response in case the United Airlines API returns with a timeout, the developer should add a Scatter-Gather component inside a Try scope, and set the payload to a default value 'Error' inside the error handler using the On Error Continue scope. A Scatter-Gather component allows sending multiple requests concurrently and aggregating the responses into an array. A Try scope allows handling errors that occur within it using an error handler. An On Error Continue scope allows continuing the flow execution after handling an error. Therefore, by using these components, the developer can send requests to both APIs in parallel, handle any timeout

errors from United Airlines API, and return a partial response with a default value for that API. Reference:
https://docs.mulesoft.com/mule-runtime/4.3/scatter-gather-concept https://docs.mulesoft.com/mule-runtime/4.3/try-scope-concept
https://docs.mulesoft.com/mule-runtime/4.3/on-error-continue-concept

Question 7

Multiple individual Mute application need to use the Mule Maven plugin to deploy to CloudHub.

The plugin configuration should .. reused where necessary and anything project, specific should be property-based.

Where should the Mule Maven details be configured?

Options:

A- A parent pom.xml

B- Settings, xml

C- Pom, xml

D- A Bill of Materials (BOM) parent pm

Answer:

A

Explanation:

To reuse Mule Maven plugin configuration across multiple individual Mule applications, the developer should use a parent pom.xml file. A parent pom.xml file defines common configuration for one or more child projects that inherit from it. The developer can specify common properties and dependencies for all child projects in the parent pom.xml file, such as Mule Maven plugin configuration, and then reference them in each child project's pom.xml file using placeholders. Reference: https://docs.mulesoft.com/mule-runtime/4.3/mmp-concept#parent-pom https://maven.apache.org/guides/introduction/introduction-to-the-pom.html#Project_Inheritance

Question 8

Question Type: **MultipleChoice**

Two APIs are deployed to a two-node on-prem cluster. Due to a requirements change, the two APIs must communicate to exchange data asynchronously.

Options:

A- If the two APIs use the same domain, the VM Connector can be leveraged

B- The VM Connector is used to inter-application communication, so it is not possible to use the VM Connector

C- Instead of using the VM Connector use <flow-ref>directly

D- It is not possible to use the VM Connector since the APIs are running in a cluster mode and each mode has it own set of VM Queues

Answer:

A

Explanation:

To communicate asynchronously between two APIs deployed to a two-node on-prem cluster, the developer can use the VM Connector if the two APIs use the same domain. The VM Connector allows passing messages between different Mule applications within a single Mule runtime instance or across different instances using shared memory or persistent storage. If two APIs are deployed under the same domain, they can share resources such as VM queues and communicate asynchronously using VM Connector operations. Reference: https://docs.mulesoft.com/mule-runtime/4.3/vm-connector https://docs.mulesoft.com/mule-runtime/4.3/shared-resources

Question 9

Question Type: **MultipleChoice**

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?

Options:

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

Answer:

B

Explanation:

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

Question 10

Question Type: **MultipleChoice**

Refer to the exhibit.

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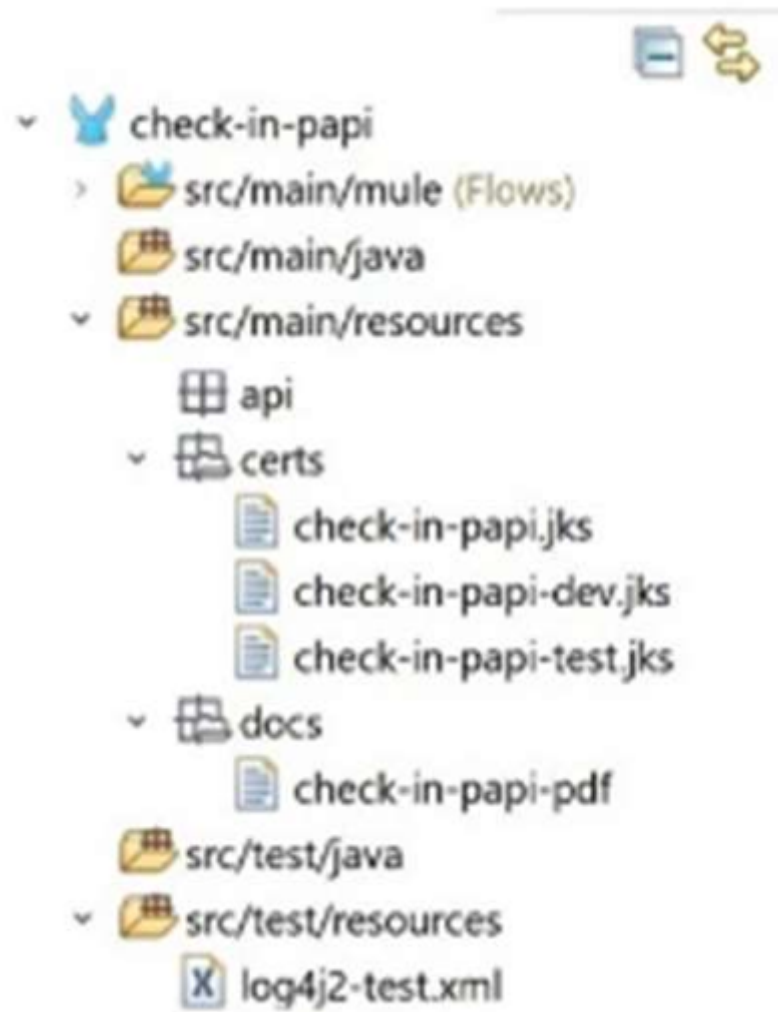
A Mule application pom.xml configures the Maven Resources plugin to exclude parsing binary files in the project's src/main/resources/certs directory.

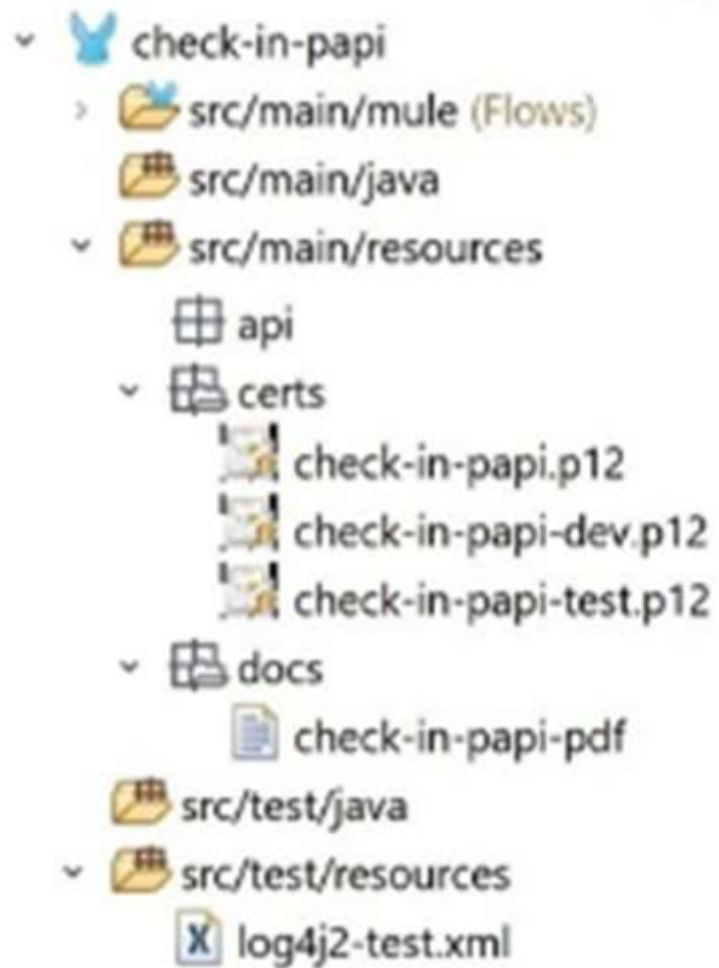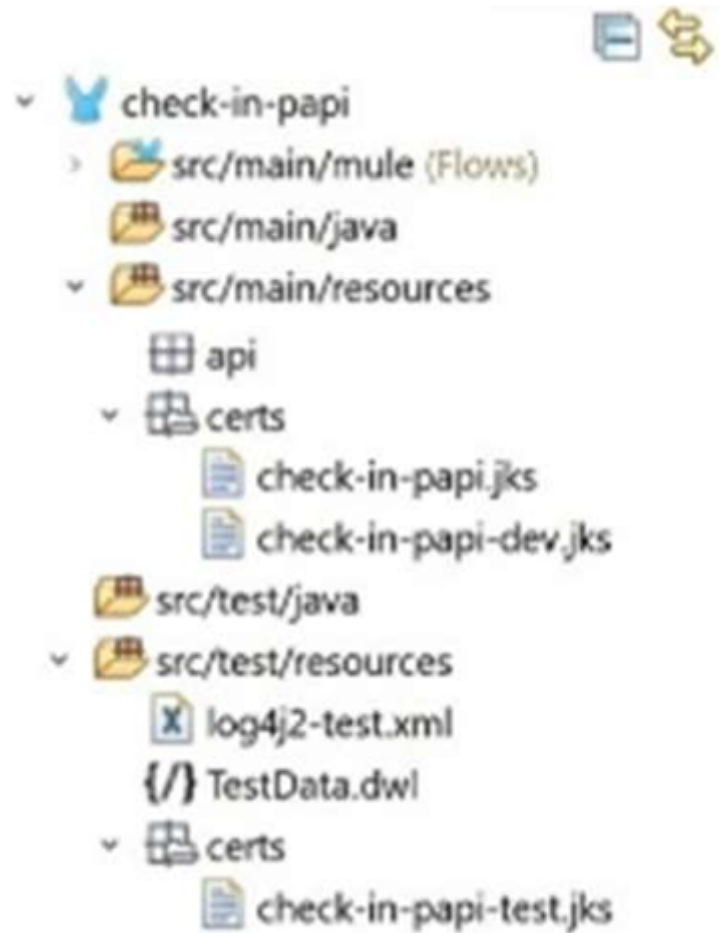Which configuration of this plugin achieves a successful build?

A)

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

Options:

A- Option A

B- Option B

C- Option C

D- Option D

Answer:

C

Explanation:

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. Reference: https://maven.apache.org/plugins/maven-resources-plugin/examples/filter.html https://maven.apache.org/plugins/maven-resources-plugin/examples/include-exclude.html