



"Please note that these files may not be up to date. However, the questions will help you understand the exam format and typical question patterns."

www.atmicnetworks.com

Warning: Keep connected with our support team
for latest updates

Question: 1

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>.

▼ vehicles-flow



► Error handling

Configuration

Protocol:

HTTPS

Host:

`$(vehicle.api.host)`

Port:

`$(vehicle.api.port)`

Use persistent connections

Max connections: -1

Connection idle timeout: 30000

Stream response

Response buffer size: 1024

TLS Configuration None

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

Answer: B

Explanation:

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. Reference: <https://docs.mulesoft.com/mule-runtime/4.3/tls-configuration#tls-default>

Question: 2

When a client and server are exchanging messages during the mTLS handshake, what is being agreed on during the cipher suite exchange?

- A. A protocol
- B. The TLS version
- C. An encryption algorithm
- D. The Public key format

Answer: C

Explanation:

A cipher suite is a set of cryptographic algorithms that are used to secure the communication between a client and a server. A cipher suite consists of four components: a key exchange algorithm, an authentication algorithm, an encryption algorithm, and a message authentication code (MAC) algorithm. During the cipher suite exchange, the client and the server agree on which encryption algorithm to use for encrypting and decrypting the data. Reference: <https://docs.mulesoft.com/mule-runtime/4.3/tls-configuration#cipher-suites>

Question: 3

A custom policy needs to be developed to intercept all outbound HTTP requests made by Mule applications. Which XML element must be used to intercept outbound HTTP requests?

- A. It is not possible to intercept outgoing HTTP requests, only inbound requests
- B. http-policy:source
- C. http-policy:operation
- D. http-policy:processor

Answer: B

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

Question: 4

An API has been built to enable scheduling email provider. The front-end system does very little data entry validation, and problems have started to appear in the email that go to patients. A validate- customer” flow is added validate the data.

What is the expected behavior of the ‘validate-customer” flow?

```
<flow name="validate-customer"
  validation:01|
  validation:is-email value="${payload.customer.emailAddress} message="invalid email address">
    <error-mapping key="VALIDATION:INVALID_EMAIL" targetType="SCHEDULE:INVALID_EMAIL_ADDRESS"/>
  </validation:is-email>
  validation:matches-regex value="${a(payload.schedule.appointmentDate regex="^\d{2}-\d{2}-\d{2}$" message="Invalid appointment date")}"
    <error-mapping sourceType="VALIDATION:MISMATCH" targetType="SCHEDULE:INVALID_APPOINTMENT_DATE"/>
  </validation:matches-regex>
  validation:is-not-null value="${payload.customer.name}" message="Invalid customer name" error-mapping
    key="VALIDATION:WILL" targetType="SCHEDULE:INVALID_CUSTOMER_NAME"/>
  </validation:is-not-null>
</validation:all>
</flow>
```

- A. If only the email address is invalid a VALIDATION.INVALID_EMAIL error is raised
- B. If the email address is invalid, processing continues to see if the appointment data and customer name are also invalid
- C. If the appointment date and customer name are invalid, a SCHEDULE:INVALID_APPOINTMENT_DATE error is raised
- D. If all of the values are invalid the last validation error is raised: SCHEDULE:INVALID_CUSTOMER_NAME

Answer: A

Explanation:

The validate-customer flow uses an until-successful scope to validate each field of the customer data. The until-successful scope executes its processors until they succeed or exhausts the maximum number of retries. If any processor fails, it raises an error and stops executing the remaining processors. Therefore, if only the email address is invalid, a VALIDATION.INVALID_EMAIL error is raised and the validation of appointment date and customer name is skipped. Reference: <https://docs.mulesoft.com/mule-runtime/4.3/until-successful-scope>

Question: 5

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

Answer: D

Explanation:

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.

Reference: <https://docs.mulesoft.com/mule-runtime/4.3/about-mule-message#message-attributes>

Question: 6

Which statement is true about using mutual TLS to secure an application?

- A. Mutual TLS requires a hardware security module to be used
- B. Mutual TLS authenticates the identity of the server before the identity of the client
- C. Mutual TLS ensures only authorized end users are allowed to access an endpoint
- D. Mutual TLS increases the encryption strength versus server-side TLS alone

Answer: B

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'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's certificate. Reference: <https://docs.mulesoft.com/mule-runtime/4.3/tls-configuration#mutual-authentication>

Question: 7

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

Answer: A

Explanation:

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.

Reference: <https://docs.mulesoft.com/mule-sdk/1.1/xml-sdk#operations>

Question: 8

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

- A. Write-through invalidation
- B. Write-behind invalidation
- C. Time to live
- D. Notification-based invalidation

Answer: C

Explanation:

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. Reference:

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

Question: 9

What is the MuleSoft recommended method to encrypt sensitive property data?

- A. The encryption key and sensitive data should be different for each environment
- B. The encryption key should be identical for all environments
- C. The encryption key should be identical for all environments and the sensitive data should be different for each environment
- D. The encryption key should be different for each environment and the sensitive data should be the same for all environments

Answer: A

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

Question: 10

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 healthcare portal.

Which files does the developer need to create in order to package the custom policy?

- A. Deployable ZIP file, YAML configuration file
- B. JSON properties file, YAML configuration file
- C. JSON properties file, XML template file
- D. XML template file, YAML configuration file

Answer: D

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

Question: 11

Refer to the exhibit.

What action must be performed to log all the errors raised by the VM Connector?



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <Configuration>
3
4 <Appenders>
5 <RollingFile name="file"
13 </Appenders>
14
15 <Loggers>
16 </Loggers>
17
18 </Configuration>
```

- A. Add `<AsyncLogger name='org.mule.extension.vm' level=ERROR'>` inside the Logger tag
- B. Add `<AsyncLogger name='org.mule.extension.vm' level=ERROR' />` inside the Appenders tag
- C. Configure `<Logger level='ERROR' />` inside the VM Connector configuration
- D. Nothing, as error-level events are automatically logged

Answer: B

Explanation:

To log all the errors raised by the VM Connector, the developer needs to add an async logger with the name 'org.mule.extension.vm' and the level 'ERROR' inside the appenders tag of the log4j2.xml file. This will enable logging all error-level events generated by the VM Connector to the console appender. Reference: <https://docs.mulesoft.com/mule-runtime/4.3/logging-in-mule#configuring-custom-logging-settings>

Question: 12

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 hit, 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 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

Answer: B

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.

Reference: <https://docs.mulesoft.com/api-manager/2.x/http-caching-policy#policy-ordering>

Question: 13

A system API that communicates to an underlying MySQL database is deploying to CloudHub. The DevOps team requires a readiness endpoint to monitor all system APIs.

Which strategy should be used to implement this endpoint?

- A. Create a dedicated endpoint that responds with the API status and reachability of the underlying systems
- B. Create a dedicated endpoint that responds with the API status and health of the server
- C. Use an existing resource endpoint of the API
- D. Create a dedicated endpoint that responds with the API status only

Answer: A

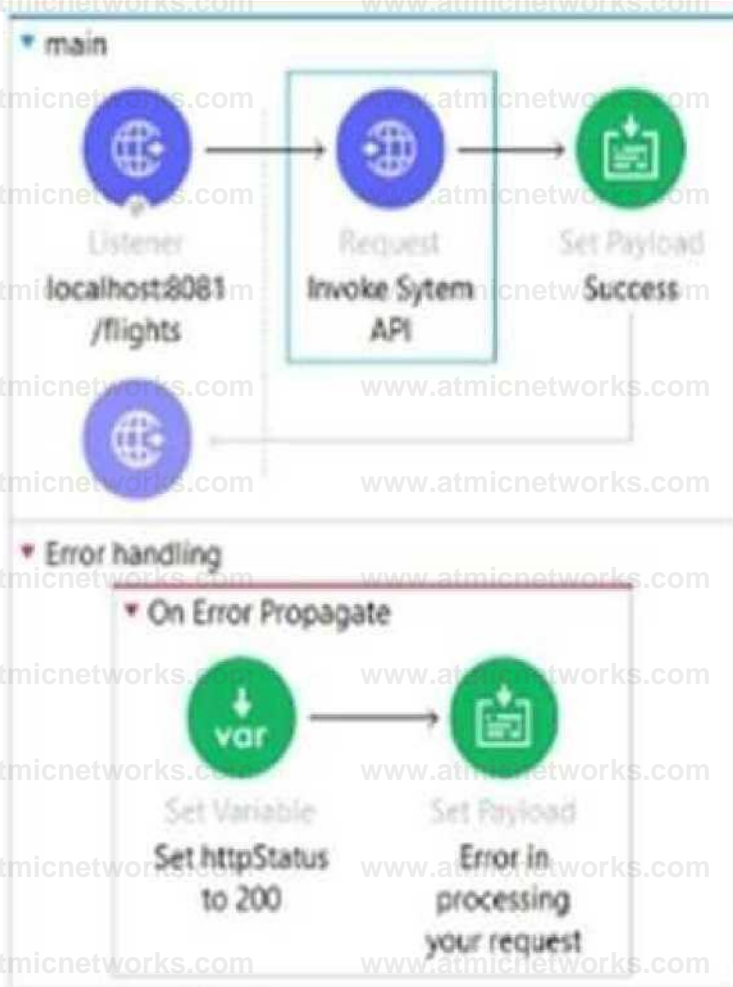
Explanation:

To implement a readiness endpoint to monitor all system APIs, the developer should create a dedicated endpoint that responds with the API status and reachability of the underlying systems. This way, the DevOps team can check if the system API is ready to receive requests and if it can communicate with its backend systems without errors.

Reference: <https://docs.mulesoft.com/mule-runtime/4.3/deployment-strategies#readiness-probes>

Question: 14

The HTTP Request operation raises an HTTP CONNECTIVITY error. Which HTTP status code and body are returned to the web client?



General	Response	
MIME Type	Body:	<code>1 payload</code>
Redelivery	Headers:	Headers
Responses	Status code:	<code>200</code>
Notes	Reason phrase:	
Help	Error Response	
	Body:	<code>1 output text/plain --- error.description</code>
	Headers:	Headers
		Name

A. HTTP Status Code:200.

Body 'Error in processing your request

B. HTTP Status Code:500.

Body 'The HTTP CONNECTIVITY Error description

C. HTTP Status Code:500.

Body 'Error in processing your request D. HTTP Status Code:500.

Body 'Error in processing your request

Answer: C

Explanation:

When the HTTP Request operation raises an HTTP CONNECTIVITY error, it triggers an on-error-continue handler that sets a payload with 'Error in processing your request'. 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 'Error in processing your request'.

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

Question: 15

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

How can this 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}`

Answer: B

Explanation:

**

`secure::tiskeystore,keyPassowrd`

ShortExplanationofCorrectAnswerOnly:Toreferenceasecureproper

tyvaluewithintheapplication,thedeveloperneedstousesthesyntax{secure::}. In this case, the property name is tiskeystore,keyPassword, so the correct syntax is `${secure::tiskeystore,keyPassowrd}`.

Reference: <https://docs.mulesoft.com/mule-runtime/4.3/secure-configuration-properties#referencing-secure-properties>

Question: 16

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?

- A. Chain together the test suites and test cases for Flow-1 and Flow-2
- B. Use "Set Event to pass the input that is needed, and keep the test cases for Flow-1 and Flow-2 independent
- C. Use "Before Test Case" To collect data from Flow-1 test cases before running Flow-2 test cases
- D. Use 'After Test Case' to produce the data needed from Flow-1 test cases to pass to Flow-2 test CASES

Answer: B

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.

Reference: <https://docs.mulesoft.com/munit/2.3/munit-test-flow>

Question: 17

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

Answer: D

Explanation:

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. Reference: <https://docs.mulesoft.com/mule-runtime/4.3/http-listener-ref#callback>

Question: 18

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="cs" entryTtl="T" entryTtlUnit="SfCOMOS"
expirationInterval="M30M" expirationIntervalUnit="SfCOWDS"/>
```

```
<How naffie>N0aln-flowM>
```

```
<set-payload value="originalPayload"/>
```

```
<os:store objectStore="os" key="testKey"/>
```

```
<os:value! [COATA("#testPayload"j)]></os:value>
```

```
</os:store*
```

```
<os:retrieve objectStore="os" key="a/'testKey'/"*
```

```
<os:default-value#[nullPayload]></os:default-value*
```

```
</os:retrieve*
```

```
</flow*
```

- A. nullPayload
- B. originalPayload
- C. OS:KEY_NOT_FOUND
- D. testPayload

Answer: A

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. Reference:

<https://docs.mulesoft.com/object-store/osv2-faq#how-does-the-time-to-live-work>

Question: 19

Which plugin or dependency is required to unit test modules created with XML SDK?

- A. XMLUnit
- B. Junit
- C. MUnit Extensions Maven plugin
- D. MUnit Maven plugin

Answer: C

Explanation:

To unit test modules created with XML SDK, the developer needs to use the MUnit Extensions Maven plugin. This plugin allows testing XML SDK modules using MUnit by adding a dependency to the module under test and using a custom processor tag to invoke it. Reference: <https://docs.mulesoft.com/mule-sdk/1.1/xml-sdk#testing>

Question: 20

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

Answer: C

Explanation:

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. Reference: <https://docs.mulesoft.com/mule-runtime/4.3/about-mule-message#message-attributes>

Question: 21

Refer to the exhibit.

What is the result of the Mule Maven Plugin configuration of the value of property its,keystorePassword in CloudHub 2.0?

<securePropeties>

<its keyStore password >\${0s keyStore password}'^ keyStore password
</securePropeties>

- A. CloudHub encrypts the value
- B. The Mule server encrypts the value
- C. Anypoint Studio secures the value
- D. Runtime Manager masks the value

Answer: D

Explanation:

The result of the Mule Maven Plugin configuration of the value of property its,keystorePassword in CloudHub 2.0 is that Runtime Manager masks the value. This means that Runtime Manager hides or obscures the value from anyone who views it in Runtime Manager or Anypoint Platform. Reference: <https://docs.mulesoft.com/runtime-manager/runtime-manager-agent-for-mule4#properties-tab>

Question: 22

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.

Answer: C

Explanation:

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. Reference: <https://docs.mulesoft.com/mule-runtime/4.3/error-handling#global-error-handler>

Question: 23

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

Answer: D

Explanation:

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

Question: 24

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

Answer: C

Explanation:

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 ScatterGather router as a Saga orchestrator allows invoking multiple HTTP APIs in parallel and rolling back any failed requests in sequence. Reference: <https://docs.mulesoft.com/mule-runtime/4.3/scatter-gather-concept>

Question: 25

Which properties are mandatory on the HTTP Connector configuration in order to use the OAuth 2.0 Authorization

Code grant type for authentication?

- A. External callback URL, access token URL, client ID response access token
- B. Token URL, authorization URL, client ID, client secret local callback URL
- C. External callback URL, access token URL, client ID, response refresh token
- D. External callback URL, access token URL, local authorization URL, authorization URL, client ID, client secret

Answer: D

Explanation:

To use the OAuth 2.0 Authorization Code grant type for authentication, the HTTP Connector configuration requires the following properties: token URL, authorization URL, client ID, client secret, and local callback URL. The token URL is the endpoint of the authorization server that provides access tokens. The authorization URL is the endpoint of the authorization server that initiates the user consent flow. The client ID and client secret are the credentials of the Mule application registered with the authorization server. The local callback URL is the endpoint of the Mule

application that receives the authorization code from the authorization server. Reference:

<https://docs.mulesoft.com/http-connector/1.6/http-authentication#oauth-2-0>

Question: 26

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']

Answer: A

Explanation:

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

Question: 27

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

Answer: C

Explanation:

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

Question: 28

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

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

Answer: B

Explanation:

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). Reference: <https://docs.oracle.com/en/java/javase/11/tools/keytool.html#GUID-5990A2E4-78E3-47B7-AE75-6D1826259549>

Question: 29

A new Mule project has been created in Anypoint Studio with the default settings.

Which file inside the Mule project must be modified before using Maven to successfully deploy the application?

- A. Settings.xml
- B. Config.yaml
- C. Pom.xml
- D. Mule.artifact.json

Answer: C

Explanation:

To use Maven to successfully deploy the application, the developer needs to modify the pom.xml file inside the Mule project. The pom.xml file contains the configuration and dependencies of the Mule application, as well as the

deployment information such as the CloudHub region, environment, and worker size. The developer needs to specify these properties in the pom.xml file before using Maven to deploy the application. Reference:

<https://docs.mulesoft.com/mule-runtime/4.3/mmp-concept#application-deployment-descriptor>

Question: 30

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?

- A. Access Anypoint Monitoring's built-in dashboard. Select a resource. Locate the information under the Connectors tab.

Answer: C

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 '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'. Reference: <https://docs.mulesoft.com/json-module/1.1/json-validate-schema>

Question: 32

When registering a client application with an existing API instance or API Group instance, what is required to manually approve or reject request access?

- A. To configure the SLA tier for the application and have the role of Organization Administrator, API Manager Environment Administrator, or the Manage Contacts permission
- B. To configure the SLA tier for the application and have the Exchange Administrator permission
- C. To configure the SLA tier for the application
- D. To only have Exchange Administrator permission

Answer: D

Explanation:

To manually approve or reject request access when registering a client application with an existing API instance or API Group instance, it is required to configure the SLA tier for the application and have one of the following roles or permissions: Organization Administrator, API Manager Environment Administrator, or Manage Contracts permission. These roles or permissions allow managing client applications and contracts in API Manager. Reference: <https://docs.mulesoft.com/api-manager/2.x/client-applications#managing-client-applications-and-contracts>

Question: 33

A Mule application deployed to a standardalone Mule runtime uses VM queues to publish messages to be consumed asynchronously by another flow.

In the case of a system failure, what will happen to in-flight messages in the VM queues that have been consumed?

- A. For any type of queue, the message will be processed after the system comes online
- B. For persistent queues, the message will be processed after the system comes online
- C. For transient queues, the message will be processed after the system comes online
- D. For any type of queue, the message will be lost

Answer: B

Explanation:

In case of a system failure, in-flight messages in persistent VM queues that have been consumed will be processed after the system comes online. This is because persistent VM queues store messages on disk and guarantee delivery even if there is a system crash or restart. Therefore, any in-flight messages that have been consumed but not processed will be recovered from disk and processed when the system is back online. Reference:

<https://docs.mulesoft.com/mule-runtime/4.3/vm-connector#persistent-queues>

Question: 34

Refer to the exhibit.

traits:

```
client** id-required: headers;
clients id:
  type: it ring
Client_secret: type:
  string
```

protocols:

```
- HTTPS
- HUP
```

```
Customers: !include datatypcs/Custoaers-request.raml
```

/customers:

```
H: (client-id-required)
```

body:

```
application/json:
```

```
type: Customers
```

```
example: 'include examples/Customers-request.json responses:
```

```
description: Successfull body:
```

```
application/json:
```

```
example: 'include examples/Customers-response.json 4M:
```

```
description: Bad request
```

```
401:
```

```
application/json:
```

```
example: 'include examples/postErrorCode403.Json 4M:
```

```
description: Resource not found
```

```
body:
```

```
application/json:
```

```
example:
```

```
'include examples/postErrorCode4e4.json 5M:
```

```
description: Internal server error body:
```

```
application/json:
```

```
example:
```

```
'include examples/postErrorCodeSM.json 5M:
```

```
description: Not implemented
```

```
body:
```

```
application/json:
```

```
example:
```

```
'include examples/postErrorcodeSOI.Json 5M:
```

A developer generates the base scaffolding for an API in Anypoint Studio.

Which HTTP status code is returned while testing using the API Kit console if no values are entered in client-secret?

- A. HTTP status code:200
- B. HTTP status code:403
- C. HTTP status code:400
- D. HTTP status code:500

Answer: B

Explanation:

Based on the code snippet and schema.json file below, when testing using the API Kit console if no values are entered in client-secret, HTTP status code 403 (FORBIDDEN) is returned. This is because client-secret is defined as a required header parameter in schema.json file, which means that it must be present in every request. If no values are entered in client-secret, then it is equivalent to omitting this header parameter, which violates the schema and causes APIKit Router to return HTTP status code 403. Reference: <https://docs.mulesoft.com/apikit/4.x/apikit-4->

[headers](#)

Question: 35

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?

- A. Initialize, validate, compute, test, package, verify, install, deploy
- B. Validate, initialize, compile, package, test, install, verify, verify, deploy
- C. Validate, initialize, compile, test package, verify, install, deploy
- D. Validation, initialize, compile, test, package, install verify, deploy

Answer: C

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

Question: 36

A Mule application deployed to multiple Cloudhub 2.0 replicas needs to temporarily persist large files over 10MB between flow executions, and routinely needs to query whether the file data exists on separate executions. How can this be achieved?

- A. Store the contents of the file on separate storage, and store the key and location of the file Object using Object Store v2
- B. Use an in-memory Object Store
- C. Store the key and full contents of the file in an Object Store
- D. Store the key and full contents of the file, caching the filename and location between requests

Answer: A

Explanation:

To temporarily persist large files over 10MB between flow executions, and routinely query whether the file data exists on separate executions, the developer should store the contents of the file on separate storage, and store the key and location of the file object using Object Store v2. This way, the

developer can avoid storing large files in memory or exceeding the size limit of Object Store v2 (10MB per object). The developer can also use Object Store v2 operations to query, retrieve, or delete the file object by its key.

Reference: <https://docs.mulesoft.com/object-store/osv2-faq#can-i-store-files-in-object-store-v2>

Question: 37

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

Answer: C

Explanation:

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.

Reference: <https://docs.mulesoft.com/mule-runtime/4.3/cryptography-pgp>

Question: 38

A healthcare 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

Answer: D

Explanation:

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 Mule applications using Java Module operations. Reference: [https://docs.mulesoft.com/mule-runtime/4.3/java-](https://docs.mulesoft.com/mule-runtime/4.3/java-module-reference#add-the-java-module-to-your-project)

[module-reference#add-the-java-module-to-your-project](https://docs.mulesoft.com/mule-runtime/4.3/java-module-reference#invoke-java-code) [https://docs.mulesoft.com/mule- runtime/4.3/java-
module-reference#invoke-java-code](https://docs.mulesoft.com/mule-runtime/4.3/java-module-reference#invoke-java-code)

Question: 39

Refer to the exhibit.

Project Setting*

Put a Mule project in the workspace or in a local location.

Project Morne »*« ■ me dL'project

Rent*?

M4efer.tr 440(1)

M4efer.tr 4.10(1)

API Impiemeatotnn

Add ic AP t^pl^nfr/M cc to yout ptow<1 to >4 w*?K«ly wf up *« APkt rout#* #rd c^Mto p<M#*oM** lto«H tot tM* ^r

Import a published API

Import RAML from local

51*4 KWdri? API npUmcnUtiocily mpoftc^thi ip«cl«t«cho»o

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

Answer: C

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. Reference:

<https://docs.mulesoft.com/apikit/4.x/apikit-4-new-project-task>

Question: 40

Which pattern can a web API use to notify its client of state changes as soon as they occur?

- A. HTTP Webhook
- B. Shared database trigger
- C. Schedule Event Publisher
- D. ETL data load

Answer: A

Explanation:

A web API can use HTTP Webhook to notify its client of state changes as soon as they occur. A webhook is an HTTP callback that allows an API to send real-time notifications to another system or application when an event happens. The client registers a URL with the API where it wants to receive notifications, and then the API sends an HTTP request to that URL with information about the event. Reference:

<https://docs.mulesoft.com/connectors/webhook/webhook-connector>

Question: 41

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?

A)

```
<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>
```

B)

```
<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>
```

C)

```
<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>
```

D)

```
<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>
```

E)

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

- A. Option A
- B. Option B
- C. Option C
- D. Option D
- E. Option E

Answer: E

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). Reference: <https://docs.mulesoft.com/apikit/4.x/apikit-4-generate-from-rest-api-task#add-the-api-dependency-to-the-pom-file>

Question: 42

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.

Answer: C

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. Reference: <https://docs.mulesoft.com/mule-runtime/4.3/until-successful-scope><https://docs.mulesoft.com/mule-runtime/4.3/on-error-continue-concept>

Question: 43

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?

- A. From one public location with each API in its own schedule
- B. From one private location with all 10 APIs in a single schedule
- C. From one public location with all 10 APIs in a single schedule
- D. From 10 public locations with each API in its own schedule

Answer: C

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

Question: 44

An order processing system is composed of multiple Mule application responsible for warehouse, sales and shipping. Each application communication using Anypoint MQ. Each message must be correlated against the original order ID for observability and tracing.

How should a developer propagate the order ID as the correlation ID across each message?

- A. Use the underlying HTTP request of Anypoint MQ to set the 'X-CORRELATION_ID' header to the order ID
- B. Set a custom Anypoint MQ user property to propagate the order ID and set the correlation ID in the receiving applications.
- C. Use the default correlation ID, Anypoint MQ will automatically propagate it.
- D. Wrap all Anypoint MQ Publish operations within a With CorrelationID scope from the Tracing module, setting the correlation ID to the order ID

Answer: D

Explanation:

To propagate the order ID as the correlation ID across each message using Anypoint MQ, the developer should wrap all Anypoint MQ Publish operations within a With CorrelationID scope from the Tracing module, setting the correlation ID to the order ID. The With CorrelationID scope allows setting a custom correlation ID for any event that occurs within it. The Tracing module also enables distributed tracing across different Mule applications and services using Anypoint Monitoring.

Reference: <https://docs.mulesoft.com/tracing-module/1.0/tracing-module-reference#with-correlation-id-scope>
<https://docs.mulesoft.com/tracing-module/1.0/tracing-module-concepts>

Question: 45

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 Nexus and copy it to the src/main/resources folder in the API
- B. Copy the common application's source XML file and put it in a new flow file in the src/main/mule folder
- C. Add a Maven dependency in the POM file with multiple-plugin as <classifier>
- D. Add a Maven dependency in the POM file with jar as <classifier>

Answer: D

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.

Reference: <https://docs.mulesoft.com/mule-runtime/4.3/mmp-concept#add-a-maven-dependency-to-the-pom-file>

Question: 46

Refer to the exhibit.

```
<flow name="implewntation" >
  <raise-error doc:rares"totse error" type: 'APP:CUSTOH_ElifW' />
</flow>
<nunit:test name="start-up test description="Test that Ail« opp starts up" expectedErrorType='APP.-O/S W_fWO'">
  <<un it executions
    (flow-ref doc:nane)"txple.nentotion" nanes"iaples>entation7s
  </munit:executions
  <nunit:validation s
    count-tools:assert-that doc :nair>"Assert that" expressions"eftruej* it*" "[MunitTools: :equalTo(false)]"/> </munit:validations
</nunit:test>
```

The flow name is "implewntation" 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

Answer: A

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 'Hello World', but the actual value returned by the implementation flow is 'Hello Mule'.

Therefore, the assertion fails and an error is thrown. Reference:

<https://docs.mulesoft.com/munit/2.3/assert-equals-processor>

Question: 47

A Flight Management System publishes gate change notification events whenever a flight'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?

- A. Publish each client subscribe directly to the exchange.
Have each client subscribe directly to the queue.
- B. Publish the gate change notification to an Anypoint MC queue
Have each client subscribe directly to the queue
- C. 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
- D. Publish the gate change notification to an Anypoint MQ exchange.
Create different Anypoint MQ queues meant for each of the other subscribing systems.
Bind the exchange with each of the queues.

Answer: D

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.

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

Question: 48

A Mule application uses API autodiscovery to access and enforce policies for a RESTful implementation.

- A. Nothing because flowRef is an optional attribute which can be passed runtime
- B. The name of the flow that has APIKit Console to receive all incoming RESTful operation requests.
- C. Any of the APIKit generate implement flows
- D. The name of the flow that has HTTP listener to receive all incoming RESTful operation requests

Answer: D

Explanation:

To use API autodiscovery to access and enforce policies for a RESTful implementation, flowRef must be set to the name of the flow that has HTTP listener to receive all incoming RESTful operation requests. This way, API autodiscovery can identify the API implementation and associate it with the corresponding API specification and policies in API Manager. The flow that has HTTP listener is usually the main flow that contains the APIKit Router.

Reference: <https://docs.mulesoft.com/api-manager/2.x/api-auto-discovery-new-concept#flowref>

Question: 49

A mule application exposes an 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?

- A. Create a health check endpoint that listens on a separate port and uses a separate HTTP Listener configuration from the API
- B. Configure the application to send health data to an external system
- C. Create a health check endpoint that reuses the same port number and HTTP Listener configuration as the API itself
- D. Monitor the Payment API directly sending real customer payment data

Answer: A

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

Question: 50

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

Which approach can Mule application B use to remove values from Mule application A's Object Store?

- A. Object Store v2 REST API
- B. CloudHub Connector
- C. Object Store Connector
- D. CloudHub REST API

Answer: A

Explanation:

To remove values from Mule application A'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's Object Store v2 by specifying the object store ID and the key of the value to delete. Reference:

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

Question: 51

Refer to the exhibit.

```

40 <build>
41   <resources>
42     <resource>
43       <directory>src/main/resources</directory>
44       <filtering>true</filtering>
45     </resource>
46   </resources>
47   <testResources>
48     <testResource>
49       <directory>src/test/resources</directory>
50       <filtering>true</filtering>
51     </testResource>
52     <testResource>
53       <directory>src/test/funnon</directory>
54       <filtering>true</filtering>
55       <targetPath>funnon</targetPath>
56     </testResource>
57   </testResources>
58   <pluginManagement>
59     <plugins>
60       <plugin>
61         <groupId>org.apache.maven.plugins</groupId>
62         <artifactId>maven-resources-plugin</artifactId>
63         <<configuration> .
64           <nonFilteredFileExtensions>
65             <nonFilteredFileExtension>p12</nonFilteredFileExtension>
66             <nonFilteredFileExtension>cert</nonFilteredFileExtension>
67             <nonFilteredFileExtension>pem</nonFilteredFileExtension>
68           </nonFilteredFileExtensions>
69         </configuration>
70       </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

L? src/mam/mule (Flows)

® sr c/mam/java

• ® src/mam/resources

ffiap*

* £j certs

.-4 check-in-pap p12

4 check in pap dev pi2

^(check in papi test p*2 . *

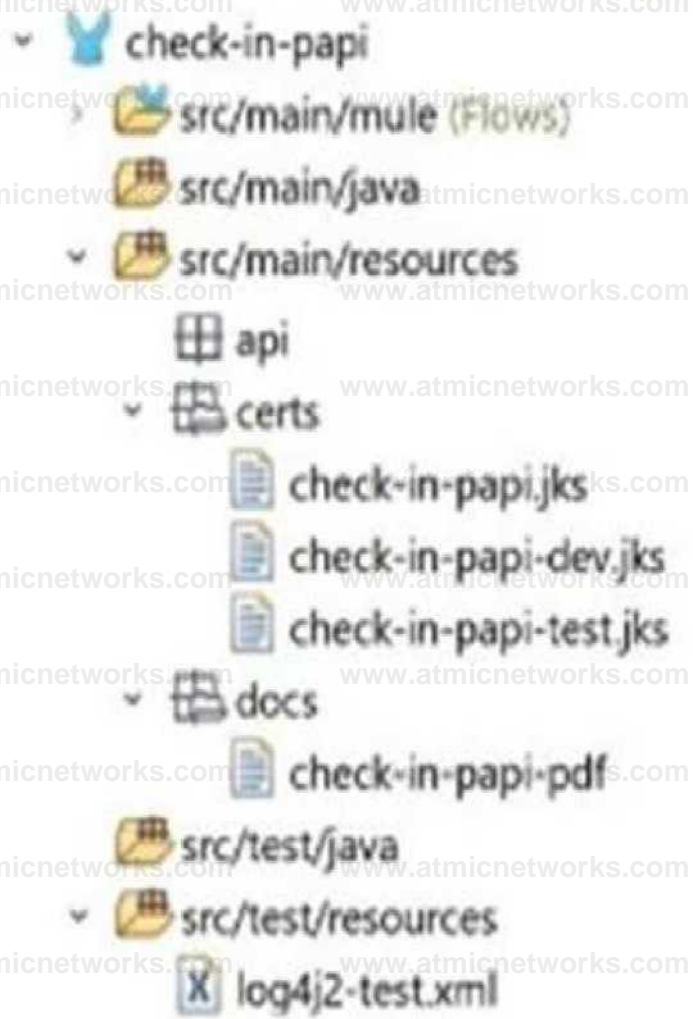
wc/tesl/java

* vc/teit/ttKurces

X log4j2 test xml

{/) T«tCata <hvl

B)



c)

*** check-in-papi**

Ssfc/mam/mule (Flows) 0 src/mam/java

*** L? src/mam/resources**

LU apt

*** Hr, cens**

Jl checkm-papi p12

,^(check m papi-dev p12

.Jx checkmpapi-test pi2

*** @docs**

| check in papi-pdf 0 src/test/java

*** 0 src/test/rcsources**

X log4j2 test xml

D)

checknc-papi

.J^ vc/nuin/mulc Tic

, * Jrc/nwin/iivi

. ■ vc/m^in/fesoucces

Hi JR



CheCK'in

check'in-pAp-devjks

¹ vc/te\$Vjava

¹ vc/Xest/rwources

X loq4L2-test xml

{/} TestDaU dwl

CfnS

check in pjpi test jk*»

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>

Question: 52

A Mule application includes a subflow containing a Scatter.Gather scope. Within each log of the Scatter.Gather, 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 idempotent 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

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 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.

Reference: <https://docs.mulesoft.com/mule-runtime/4.3/scatter-gather-concept>
<https://docs.mulesoft.com/mule-runtime/4.3/error-handling>

Question: 53

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.

- 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 its 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: 54

Multiple individual Mule applications need to use the Mule Maven plugin to deploy to CloudHub. The plugin configuration should be reused where necessary and anything project-specific should be property-based. Where should the Mule Maven details be configured?

- A. A parent pom.xml
- B. Settings, xml

- C. Pom, xml
- D. A Bill of Materials (BOM) parent pm

Answer: A

Explanation:

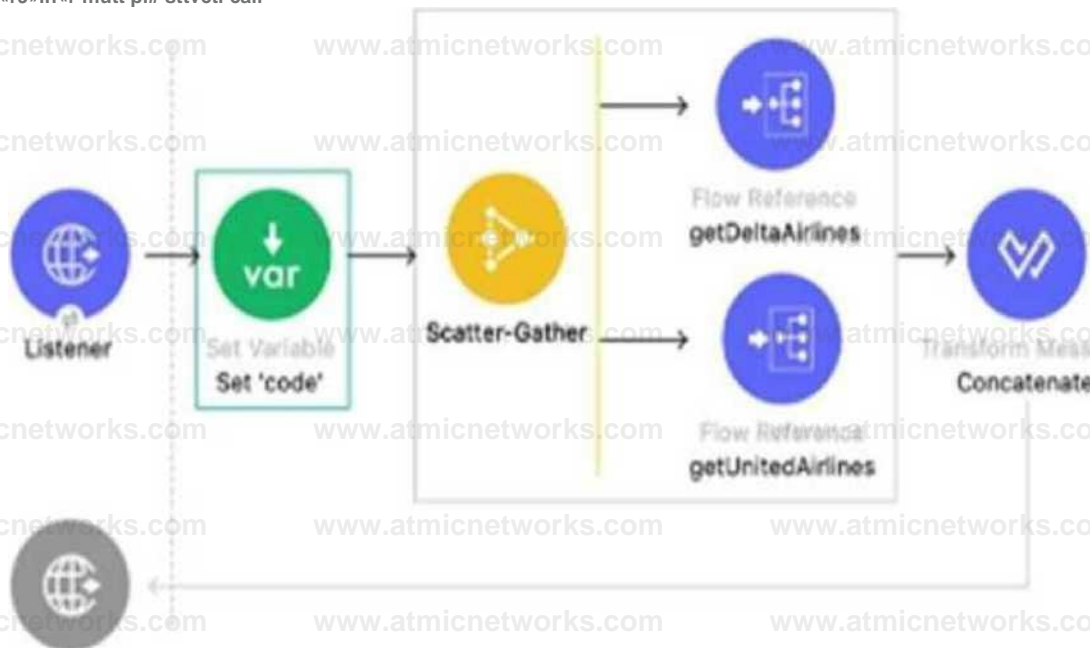
To reuse Mule Maven plugin configuration across multiple individual Mule applications, the developer should use a parent pom.xml file. A parent pom.xml file defines common configuration for one or more child projects that inherit from it. The developer can specify common properties and dependencies for 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: 55

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?

\$C4tt«r9»lh«r mutt pl# sttvcti call



► Error handling

- 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 " inside 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: 56

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?

- 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: 57

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?

- 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: 58

Refer to the exhibit.

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

Global Element Properties

HTTP UiteMf config Colouration attract for * Mttpllttefw

General Hotei Help

Hma . HTTPJUMeeer.COOnly

Connercoo

General RS Mranced

RS Cont gvratio Wt rene

Tout Store Cerf 9.x at cn

Path: Peftworp:

Show peeinord

Type:

Algothm

t insecure



- 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: 59

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

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: 60

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?

- 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 then modify the <http:listener> 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>