



# MuleSoft

## Exam Questions MCPA-Level-1

MuleSoft Certified Platform Architect - Level 1

## About ExamBible

### *Your Partner of IT Exam*

## Found in 1998

ExamBible is a company specialized on providing high quality IT exam practice study materials, especially Cisco CCNA, CCDA, CCNP, CCIE, Checkpoint CCSE, CompTIA A+, Network+ certification practice exams and so on. We guarantee that the candidates will not only pass any IT exam at the first attempt but also get profound understanding about the certificates they have got. There are so many alike companies in this industry, however, ExamBible has its unique advantages that other companies could not achieve.

## Our Advances

### \* 99.9% Uptime

All examinations will be up to date.

### \* 24/7 Quality Support

We will provide service round the clock.

### \* 100% Pass Rate

Our guarantee that you will pass the exam.

### \* Unique Gurantee

If you do not pass the exam at the first time, we will not only arrange FULL REFUND for you, but also provide you another exam of your claim, ABSOLUTELY FREE!

### NEW QUESTION 1

How are an API implementation, API client, and API consumer combined to invoke and process an API?

- A. The API consumer creates an API implementation, which receives API invocations from an API such that they are processed for an API client
- B. The API client creates an API consumer, which receives API invocations from an API such that they are processed for an API implementation
- C. The API consumer creates an API client, which sends API invocations to an API such that they are processed by an API implementation
- D. The API client creates an API consumer, which sends API invocations to an API such that they are processed by an API implementation

**Answer: C**

#### Explanation:

Correct Answer

The API consumer creates an API client, which sends API invocations to an API such that they are processed by an API implementation

\*\*\*\*\* Terminology:

>> API Client - It is a piece of code or program that is written to invoke an API

>> API Consumer - An owner/entity who owns the API Client. API Consumers write API clients.

>> API - The provider of the API functionality. Typically an API Instance on API Manager where they are managed and operated.

>> API Implementation - The actual piece of code written by API provider where the functionality of the API is implemented. Typically, these are Mule Applications running on Runtime Manager.

### NEW QUESTION 2

In which layer of API-led connectivity, does the business logic orchestration reside?

- A. System Layer
- B. Experience Layer
- C. Process Layer

**Answer: C**

#### Explanation:

Correct Answer

Process Layer

\*\*\*\*\*

>> Experience layer is dedicated for enrichment of end user experience. This layer is to meet the needs of different API clients/ consumers.

>> System layer is dedicated to APIs which are modular in nature and implement/ expose various individual functionalities of backend systems

>> Process layer is the place where simple or complex business orchestration logic is written by invoking one or many System layer modular APIs  
So, Process Layer is the right answer.

### NEW QUESTION 3

True or False. We should always make sure that the APIs being designed and developed are self-servable even if it needs more man-day effort and resources.

- A. FALSE
- B. TRUE

**Answer: B**

#### Explanation:

Correct Answer

TRUE

\*\*\*\*\*

>> As per MuleSoft proposed IT Operating Model, designing APIs and making sure that they are discoverable and self-servable is VERY VERY IMPORTANT and decides the success of an API and its application network.

### NEW QUESTION 4

What best describes the Fully Qualified Domain Names (FQDNs), also known as DNS entries, created when a Mule application is deployed to the CloudHub Shared Worker Cloud?

- A. A fixed number of FQDNs are created, IRRESPECTIVE of the environment and VPC design
- B. The FQDNs are determined by the application name chosen, IRRESPECTIVE of the region
- C. The FQDNs are determined by the application name, but can be modified by an administrator after deployment
- D. The FQDNs are determined by both the application name and the Anypoint Platform organization

**Answer: B**

#### Explanation:

Correct Answer

The FQDNs are determined by the application name chosen, IRRESPECTIVE of the region

\*\*\*\*\*

>> When deploying applications to Shared Worker Cloud, the FQDN are always determined by application name chosen.

>> It does NOT matter what region the app is being deployed to.

>> Although it is fact and true that the generated FQDN will have the region included in it (Ex: exp-salesorder-api.au-s1.cloudhub.io), it does NOT mean that the same name can be used when deploying to another CloudHub region.

>> Application name should be universally unique irrespective of Region and Organization and solely determines the FQDN for Shared Load Balancers.

#### NEW QUESTION 5

What best explains the use of auto-discovery in API implementations?

- A. It makes API Manager aware of API implementations and hence enables it to enforce policies
- B. It enables Anypoint Studio to discover API definitions configured in Anypoint Platform
- C. It enables Anypoint Exchange to discover assets and makes them available for reuse
- D. It enables Anypoint Analytics to gain insight into the usage of APIs

**Answer:** A

#### Explanation:

Correct Answer

It makes API Manager aware of API implementations and hence enables it to enforce policies.

\*\*\*\*\*

>> API Autodiscovery is a mechanism that manages an API from API Manager by pairing the deployed application to an API created on the platform.

>> API Management includes tracking, enforcing policies if you apply any, and reporting API analytics.

>> Critical to the Autodiscovery process is identifying the API by providing the API name and version. References:

<https://docs.mulesoft.com/api-manager/2.x/api-auto-discovery-new-concept> <https://docs.mulesoft.com/api-manager/1.x/api-auto-discovery>

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

#### NEW QUESTION 6

What is most likely NOT a characteristic of an integration test for a REST API implementation?

- A. The test needs all source and/or target systems configured and accessible
- B. The test runs immediately after the Mule application has been compiled and packaged
- C. The test is triggered by an external HTTP request
- D. The test prepares a known request payload and validates the response payload

**Answer:** B

#### Explanation:

Correct Answer

The test runs immediately after the Mule application has been compiled and packaged

\*\*\*\*\*

>> Integration tests are the last layer of tests we need to add to be fully covered.

>> These tests actually run against Mule running with your full configuration in place and are tested from external source as they work in PROD.

>> These tests exercise the application as a whole with actual transports enabled. So, external systems are affected when these tests run.

So, these tests do NOT run immediately after the Mule application has been compiled and packaged.

FYI... Unit Tests are the one that run immediately after the Mule application has been compiled and packaged.

#### NEW QUESTION 7

Say, there is a legacy CRM system called CRM-Z which is offering below functions:

- \* 1. Customer creation
- \* 2. Amend details of an existing customer
- \* 3. Retrieve details of a customer
- \* 4. Suspend a customer

- A. Implement a system API named customerManagement which has all the functionalities wrapped in it as various operations/resources
- B. Implement different system APIs named createCustomer, amendCustomer, retrieveCustomer and suspendCustomer as they are modular and have separation of concerns
- C. Implement different system APIs named createCustomerInCRMZ, amendCustomerInCRMZ, retrieveCustomerFromCRMZ and suspendCustomerInCRMZ as they are modular and have separation of concerns

**Answer:** B

#### Explanation:

Correct Answer

Implement different system APIs named createCustomer, amendCustomer, retrieveCustomer and suspendCustomer as they are modular and have separation of concerns

\*\*\*\*\*

>> It is quite normal to have a single API and different Verb + Resource combinations. However, this fits well for an Experience API or a Process API but not a best architecture style for System APIs. So, option with just one customerManagement API is not the best choice here.

>> The option with APIs in createCustomerInCRMZ format is next close choice w.r.t modularization and less maintenance but the naming of APIs is directly coupled with the legacy system. A better foreseen approach would be to name your APIs by abstracting the backend system names as it allows seamless replacement/migration of any backend system anytime. So, this is not the correct choice too.

>> createCustomer, amendCustomer, retrieveCustomer and suspendCustomer is the right approach and is the best fit compared to other options as they are both modular and same time got the names decoupled from backend system and it has covered all requirements a System API needs.

#### NEW QUESTION 8

Refer to the exhibit.

Three business processes need to be implemented, and the implementations need to communicate with several different SaaS applications.

These processes are owned by separate (siloe) LOBs and are mainly independent of each other, but do share a few business entities. Each LOB has one development team and their own budget

In this organizational context, what is the most effective approach to choose the API data models for the APIs that will implement these business processes with minimal redundancy of the data models?

- A) Build several Bounded Context Data Models that align with coherent parts of the business processes and the definitions of associated business entities

B) Build distinct data models for each API to follow established micro-services and Agile API-centric practices

C) Build all API data models using XML schema to drive consistency and reuse across the organization

D) Build one centralized Canonical Data Model (Enterprise Data Model) that unifies all the data types from all three business processes, ensuring the data model is consistent and non-redundant

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

**Answer:** A

**Explanation:**

Correct Answer

Build several Bounded Context Data Models that align with coherent parts of the business processes and the definitions of associated business entities.

\*\*\*\*\*

>> The options w.r.t building API data models using XML schema/ Agile API-centric practices are irrelevant to the scenario given in the question. So these two are INVALID.

>> Building EDM (Enterprise Data Model) is not feasible or right fit for this scenario as the teams and LOBs work in silo and they all have different initiatives, budget etc.. Building EDM needs intensive coordination among all the team which evidently seems not possible in this scenario.

So, the right fit for this scenario is to build several Bounded Context Data Models that align with coherent parts of the business processes and the definitions of associated business entities.

#### NEW QUESTION 9

What are the major benefits of MuleSoft proposed IT Operating Model?

- A. \* 1. Decrease the IT delivery gap\* 2. Meet various business demands without increasing the IT capacity\* 3. Focus on creation of reusable assets first
- B. Upon finishing creation of all the possible assets then inform the LOBs in the organization to start using them
- C. \* 1. Decrease the IT delivery gap\* 2. Meet various business demands by increasing the IT capacity and forming various IT departments\* 3. Make consumption of assets at the rate of production
- D. \* 1. Decrease the IT delivery gap\* 2. Meet various business demands without increasing the IT capacity\* 3. Make consumption of assets at the rate of

production

**Answer:** C

**Explanation:**

Correct Answer

- \* 1. Decrease the IT delivery gap
- \* 2. Meet various business demands without increasing the IT capacity
- \* 3. Make consumption of assets at the rate of production.

\*\*\*\*\*

#### NEW QUESTION 10

How can the application of a rate limiting API policy be accurately reflected in the RAML definition of an API?

- A. By refining the resource definitions by adding a description of the rate limiting policy behavior
- B. By refining the request definitions by adding a remaining Requests query parameter with description, type, and example
- C. By refining the response definitions by adding the out-of-the-box Anypoint Platform rate-limit-enforcement securityScheme with description, type, and example
- D. By refining the response definitions by adding the x-ratelimit-\* response headers with description, type, and example

**Answer:** D

**Explanation:**

Correct Answer

By refining the response definitions by adding the x-ratelimit-\* response headers with description, type, and example

\*\*\*\*\*

References:

<https://docs.mulesoft.com/api-manager/2.x/rate-limiting-and-throttling#response-headers> <https://docs.mulesoft.com/api-manager/2.x/rate-limiting-and-throttling-sla-based-policies#response-headers>

#### NEW QUESTION 10

In an organization, the InfoSec team is investigating Anypoint Platform related data traffic.

From where does most of the data available to Anypoint Platform for monitoring and alerting originate?

- A. From the Mule runtime or the API implementation, depending on the deployment model
- B. From various components of Anypoint Platform, such as the Shared Load Balancer, VPC, and Mule runtimes
- C. From the Mule runtime or the API Manager, depending on the type of data
- D. From the Mule runtime irrespective of the deployment model

**Answer:** D

**Explanation:**

Correct Answer

From the Mule runtime irrespective of the deployment model

\*\*\*\*\*

>> Monitoring and Alerting metrics are always originated from Mule Runtimes irrespective of the deployment model.

>> It may seem that some metrics (Runtime Manager) are originated from Mule Runtime and some are (API Invocations/ API Analytics) from API Manager.

However, this is realistically NOT TRUE. The reason is, API manager is just a management tool for API instances but all policies upon applying on APIs eventually gets executed on Mule Runtimes only (Either Embedded or API Proxy).

>> Similarly all API Implementations also run on Mule Runtimes.

So, most of the day required for monitoring and alerts are originated from Mule Runtimes only irrespective of whether the deployment model is MuleSoft-hosted or Customer-hosted or Hybrid.

#### NEW QUESTION 15

A Mule application exposes an HTTPS endpoint and is deployed to the CloudHub Shared Worker Cloud. All traffic to that Mule application must stay inside the AWS VPC.

To what TCP port do API invocations to that Mule application need to be sent?

- A. 443
- B. 8081
- C. 8091
- D. 8082

**Answer:** D

**Explanation:**

Correct Answer 8082

\*\*\*\*\*

>> 8091 and 8092 ports are to be used when keeping your HTTP and HTTPS app private to the LOCAL VPC respectively.

>> Above TWO ports are not for Shared AWS VPC/ Shared Worker Cloud.

>> 8081 is to be used when exposing your HTTP endpoint app to the internet through Shared LB

>> 8082 is to be used when exposing your HTTPS endpoint app to the internet through Shared LB So, API invocations should be sent to port 8082 when calling this HTTPS based app.

References:

<https://docs.mulesoft.com/runtime-manager/cloudhub-networking-guide> <https://help.mulesoft.com/s/article/Configure-Cloudhub-Application-to-Send-a-HTTPS-Request-Directly-to-An>

<https://help.mulesoft.com/s/question/0D52T00004mXXULSA4/multiple-http-listeners-on-cloudhub-one-with-p>

### NEW QUESTION 18

What are 4 important Platform Capabilities offered by Anypoint Platform?

- A. API Versioning, API Runtime Execution and Hosting, API Invocation, API Consumer Engagement
- B. API Design and Development, API Runtime Execution and Hosting, API Versioning, API Deprecation
- C. API Design and Development, API Runtime Execution and Hosting, API Operations and Management, API Consumer Engagement
- D. API Design and Development, API Deprecation, API Versioning, API Consumer Engagement

**Answer:** C

#### Explanation:

Correct Answer

API Design and Development, API Runtime Execution and Hosting, API Operations and Management, API Consumer Engagement

\*\*\*\*\*

- >> API Design and Development - Anypoint Studio, Anypoint Design Center, Anypoint Connectors
- >> API Runtime Execution and Hosting - Mule Runtimes, CloudHub, Runtime Services
- >> API Operations and Management - Anypoint API Manager, Anypoint Exchange
- >> API Consumer Management - API Contracts, Public Portals, Anypoint Exchange, API Notebooks

### NEW QUESTION 22

When using CloudHub with the Shared Load Balancer, what is managed EXCLUSIVELY by the API implementation (the Mule application) and NOT by Anypoint Platform?

- A. The assignment of each HTTP request to a particular CloudHub worker
- B. The logging configuration that enables log entries to be visible in Runtime Manager
- C. The SSL certificates used by the API implementation to expose HTTPS endpoints
- D. The number of DNS entries allocated to the API implementation

**Answer:** C

#### Explanation:

Correct Answer

The SSL certificates used by the API implementation to expose HTTPS endpoints

\*\*\*\*\*

- >> The assignment of each HTTP request to a particular CloudHub worker is taken care by Anypoint Platform itself. We need not manage it explicitly in the API implementation and in fact we CANNOT manage it in the API implementation.
  - >> The logging configuration that enables log entries to be visible in Runtime Manager is ALWAYS managed in the API implementation and NOT just for SLB. So this is not something we do EXCLUSIVELY when using SLB.
  - >> We DO NOT manage the number of DNS entries allocated to the API implementation inside the code. Anypoint Platform takes care of this.
- It is the SSL certificates used by the API implementation to expose HTTPS endpoints that is to be managed EXCLUSIVELY by the API implementation. Anypoint Platform does NOT do this when using SLBs.

### NEW QUESTION 23

A set of tests must be performed prior to deploying API implementations to a staging environment. Due to data security and access restrictions, untested APIs cannot be granted access to the backend systems, so instead mocked data must be used for these tests. The amount of available mocked data and its contents is sufficient to entirely test the API implementations with no active connections to the backend systems. What type of tests should be used to incorporate this mocked data?

- A. Integration tests
- B. Performance tests
- C. Functional tests (Blackbox)
- D. Unit tests (Whitebox)

**Answer:** D

#### Explanation:

Correct Answer

Unit tests (Whitebox)

\*\*\*\*\*

### NEW QUESTION 25

When could the API data model of a System API reasonably mimic the data model exposed by the corresponding backend system, with minimal improvements over the backend system's data model?

- A. When there is an existing Enterprise Data Model widely used across the organization
- B. When the System API can be assigned to a bounded context with a corresponding data model
- C. When a pragmatic approach with only limited isolation from the backend system is deemed appropriate
- D. When the corresponding backend system is expected to be replaced in the near future

**Answer:** C

#### Explanation:

#### Correct Answer

When a pragmatic approach with only limited isolation from the backend system is deemed appropriate.

\*\*\*\*\* General guidance w.r.t choosing Data Models:

>> If an Enterprise Data Model is in use then the API data model of System APIs should make use of data types from that Enterprise Data Model and the corresponding API implementation should translate between these data types from the Enterprise Data Model and the native data model of the backend system.

>> If no Enterprise Data Model is in use then each System API should be assigned to a Bounded Context, the API data model of System APIs should make use of data types from the corresponding Bounded Context Data Model and the corresponding API implementation should translate between these data types from the Bounded Context Data Model and the native data model of the backend system. In this scenario, the data types in the Bounded Context Data Model are defined purely in terms of their business characteristics and are typically not related to the native data model of the backend system. In other words, the translation effort may be significant.

>> If no Enterprise Data Model is in use, and the definition of a clean Bounded Context Data Model is considered too much effort, then the API data model of System APIs should make use of data types that approximately mirror those from the backend system, same semantics and naming as backend system, lightly sanitized, expose all fields needed for the given System API's functionality, but not significantly more and making good use of REST conventions.

The latter approach, i.e., exposing in System APIs an API data model that basically mirrors that of the backend system, does not provide satisfactory isolation from backend systems through the System API tier on its own. In particular, it will typically not be possible to "swap out" a backend system without significantly changing all System APIs in front of that backend system and therefore the API implementations of all Process APIs that depend on those System APIs! This is so because it is not desirable to prolong the life of a previous backend system's data model in the form of the API data model of System APIs that now front a new backend system. The API data models of System APIs following this approach must therefore change when the backend system is replaced.

On the other hand:

>> It is a very pragmatic approach that adds comparatively little overhead over accessing the backend system directly

>> Isolates API clients from intricacies of the backend system outside the data model (protocol, authentication, connection pooling, network address, ...)

>> Allows the usual API policies to be applied to System APIs

>> Makes the API data model for interacting with the backend system explicit and visible, by exposing it in the RAML definitions of the System APIs

>> Further isolation from the backend system data model does occur in the API implementations of the Process API tier

#### NEW QUESTION 26

Question 10: Skipped

An API implementation returns three X-RateLimit-\* HTTP response headers to a requesting API client. What type of information do these response headers indicate to the API client?

- A. The error codes that result from throttling
- B. A correlation ID that should be sent in the next request
- C. The HTTP response size
- D. The remaining capacity allowed by the API implementation

**Answer:** D

#### Explanation:

#### Correct Answer

The remaining capacity allowed by the API implementation.

\*\*\*\*\*

>> Reference:

<https://docs.mulesoft.com/api-manager/2.x/rate-limiting-and-throttling-sla-based-policies#response-headers>

#### NEW QUESTION 27

What correctly characterizes unit tests of Mule applications?

- A. They test the validity of input and output of source and target systems
- B. They must be run in a unit testing environment with dedicated Mule runtimes for the environment
- C. They must be triggered by an external client tool or event source
- D. They are typically written using MUnit to run in an embedded Mule runtime that does not require external connectivity

**Answer:** D

**Explanation:**

Correct Answer

They are typically written using MUnit to run in an embedded Mule runtime that does not require external connectivity.

\*\*\*\*\*

Below TWO are characteristics of Integration Tests but NOT unit tests:

>> They test the validity of input and output of source and target systems.

>> They must be triggered by an external client tool or event source.

It is NOT TRUE that Unit Tests must be run in a unit testing environment with dedicated Mule runtimes for the environment.

MuleSoft offers MUnit for writing Unit Tests and they run in an embedded Mule Runtime without needing any separate/ dedicated Runtimes to execute them. They also do NOT need any external connectivity as MUnit supports mocking via stubs.

<https://dzone.com/articles/munit-framework>

**NEW QUESTION 31**

What API policy would LEAST likely be applied to a Process API?

- A. Custom circuit breaker
- B. Client ID enforcement
- C. Rate limiting
- D. JSON threat protection

**Answer:** D

**Explanation:**

Correct Answer

JSON threat protection

\*\*\*\*\*

Fact: Technically, there are no restrictions on what policy can be applied in what layer. Any policy can be applied on any layer API. However, context should also be considered properly before blindly applying the policies on APIs.

That is why, this question asked for a policy that would LEAST likely be applied to a Process API. From the given options:

>> All policies except "JSON threat protection" can be applied without hesitation to the APIs in Process tier.

>> JSON threat protection policy ideally fits for experience APIs to prevent suspicious JSON payload coming from external API clients. This covers more of a security aspect by trying to avoid possibly malicious and harmful JSON payloads from external clients calling experience APIs.

As external API clients are NEVER allowed to call Process APIs directly and also these kind of malicious and harmful JSON payloads are always stopped at experience API layer only using this policy, it is LEAST LIKELY that this same policy is again applied on Process Layer API.

**NEW QUESTION 35**

An organization wants MuleSoft-hosted runtime plane features (such as HTTP load balancing, zero downtime, and horizontal and vertical scaling) in its Azure environment. What runtime plane minimizes the organization's effort to achieve these features?

- A. Anypoint Runtime Fabric
- B. Anypoint Platform for Pivotal Cloud Foundry
- C. CloudHub
- D. A hybrid combination of customer-hosted and MuleSoft-hosted Mule runtimes

**Answer:** A

**Explanation:**

Correct Answer

Anypoint Runtime Fabric

\*\*\*\*\*

>> When a customer is already having an Azure environment, It is not at all an ideal approach to go with hybrid model having some Mule Runtimes hosted on Azure and some on MuleSoft. This is unnecessary and useless.

>> CloudHub is a Mulesoft-hosted Runtime plane and is on AWS. We cannot customize to point CloudHub to customer's Azure environment.

>> Anypoint Platform for Pivotal Cloud Foundry is specifically for infrastructure provided by Pivotal Cloud Foundry

>> Anypoint Runtime Fabric is right answer as it is a container service that automates the deployment and orchestration of Mule applications and API gateways.

Runtime Fabric runs within a customer-managed infrastructure on AWS, Azure, virtual machines (VMs), and bare-metal servers.

-Some of the capabilities of Anypoint Runtime Fabric include:

-Isolation between applications by running a separate Mule runtime per application.

-Ability to run multiple versions of Mule runtime on the same set of resources.

-Scaling applications across multiple replicas.

-Automated application fail-over.

-Application management with Anypoint Runtime Manager.

**NEW QUESTION 38**

What is a key performance indicator (KPI) that measures the success of a typical C4E that is immediately apparent in responses from the Anypoint Platform APIs?

- A. The number of production outage incidents reported in the last 24 hours
- B. The number of API implementations that have a publicly accessible HTTP endpoint and are being managed by Anypoint Platform
- C. The fraction of API implementations deployed manually relative to those deployed using a CI/CD tool
- D. The number of API specifications in RAML or OAS format published to Anypoint Exchange

**Answer:** D

**Explanation:**

Correct Answer

The number of API specifications in RAML or OAS format published to Anypoint Exchange

\*\*\*\*\*

>> The success of C4E always depends on their contribution to the number of reusable assets that they have helped to build and publish to Anypoint Exchange.

>> It is NOT due to any factors w.r.t # of outages, Manual vs CI/CD deployments or Publicly accessible HTTP endpoints

>> Anypoint Platform APIs helps us to quickly run and get the number of published RAML/OAS assets to Anypoint Exchange. This clearly depicts how successful a C4E team is based on number of returned assets in the response.

**NEW QUESTION 43**

Refer to the exhibit. An organization needs to enable access to their customer data from both a mobile app and a web application, which each need access to common fields as well as certain unique fields.

The data is available partially in a database and partially in a 3rd-party CRM system.

What APIs should be created to best fit these design requirements?

A) A Process API that contains the data required by both the web and mobile apps, allowing these applications to invoke it directly and access the data they need thereby providing the flexibility to add more fields in the future without needing API changes

B) One set of APIs (Experience API, Process API, and System API) for the web app, and another set for the mobile app

C) Separate Experience APIs for the mobile and web app, but a common Process API that invokes separate System APIs created for the database and CRM system

D) A common Experience API used by both the web and mobile apps, but separate Process APIs for the web and mobile apps that interact with the database and the CRM System

A. Option A

B. Option B

C. Option C

D. Option D

**Answer:** C

**Explanation:**

Correct Answer

Separate Experience APIs for the mobile and web app, but a common Process API that invokes separate System APIs created for the database and CRM system

\*\*\*\*\* As per MuleSoft's API-led connectivity:

>> Experience APIs should be built as per each consumer needs and their experience.

>> Process APIs should contain all the orchestration logic to achieve the business functionality.

>> System APIs should be built for each backend system to unlock their data.

**NEW QUESTION 46**

What is a key requirement when using an external Identity Provider for Client Management in Anypoint Platform?

A. Single sign-on is required to sign in to Anypoint Platform

B. The application network must include System APIs that interact with the Identity Provider

C. To invoke OAuth 2.0-protected APIs managed by Anypoint Platform, API clients must submit access tokens issued by that same Identity Provider

D. APIs managed by Anypoint Platform must be protected by SAML 2.0 policies

**Answer:** C

**Explanation:**

<https://www.folkstalk.com/2019/11/mulesoft-integration-and-platform.html>

Correct Answer

To invoke OAuth 2.0-protected APIs managed by Anypoint Platform, API clients must submit access tokens issued by that same Identity Provider

\*\*\*\*\*

>> It is NOT necessary that single sign-on is required to sign in to Anypoint Platform because we are using an external Identity Provider for Client Management

>> It is NOT necessary that all APIs managed by Anypoint Platform must be protected by SAML 2.0 policies because we are using an external Identity Provider for Client Management

>> Not TRUE that the application network must include System APIs that interact with the Identity Provider because we are using an external Identity Provider for Client Management

Only TRUE statement in the given options is - "To invoke OAuth 2.0-protected APIs managed by Anypoint Platform, API clients must submit access tokens issued by that same Identity Provider"

References:

<https://docs.mulesoft.com/api-manager/2.x/external-oauth-2.0-token-validation-policy> <https://blogs.mulesoft.com/dev/api-dev/api-security-ways-to-authenticate-and-authorize/>

**NEW QUESTION 47**

Once an API Implementation is ready and the API is registered on API Manager, who should request the access to the API on Anypoint Exchange?

- A. None
- B. Both
- C. API Client
- D. API Consumer

**Answer: D**

**Explanation:**

Correct Answer  
API Consumer

\*\*\*\*\*

>> API clients are piece of code or programs that use the client credentials of API consumer but does not directly interact with Anypoint Exchange to get the access

>> API consumer is the one who should get registered and request access to API and then API client needs to use those client credentials to hit the APIs  
So, API consumer is the one who needs to request access on the API from Anypoint Exchange

**NEW QUESTION 52**

What should be ensured before sharing an API through a public Anypoint Exchange portal?

- A. The visibility level of the API instances of that API that need to be publicly accessible should be set to public visibility
- B. The users needing access to the API should be added to the appropriate role in Anypoint Platform
- C. The API should be functional with at least an initial implementation deployed and accessible for users to interact with
- D. The API should be secured using one of the supported authentication/authorization mechanisms to ensure that data is not compromised

**Answer: A**

**Explanation:**

Correct Answer

The visibility level of the API instances of that API that need to be publicly accessible should be set to public visibility.

\*\*\*\*\*

**NEW QUESTION 57**

Refer to the exhibit.

A RAML definition has been proposed for a new Promotions Process API, and has been published to Anypoint Exchange.

The Marketing Department, who will be an important consumer of the Promotions API, has important requirements and expectations that must be met.

What is the most effective way to use Anypoint Platform features to involve the Marketing Department in this early API design phase?

- A) Ask the Marketing Department to interact with a mocking implementation of the API using the automatically generated API Console
- B) Organize a design workshop with the DBAs of the Marketing Department in which the database schema of the Marketing IT systems is translated into RAML
- C) Use Anypoint Studio to Implement the API as a Mule application, then deploy that API implementation to CloudHub and ask the Marketing Department to interact with it
- D) Export an integration test suite from API designer and have the Marketing Department execute the tests In that suite to ensure they pass

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

**Answer:** A

**Explanation:**

Correct Answer

Ask the Marketing Department to interact with a mocking implementation of the API using the automatically generated API Console.

\*\*\*\*\* As per MuleSoft's IT Operating Model:

>> API consumers need NOT wait until the full API implementation is ready.

>> NO technical test-suites needs to be shared with end users to interact with APIs.

>> Anypoint Platform offers a mocking capability on all the published API specifications to Anypoint Exchange which also will be rich in documentation covering all details of API functionalities and working nature.

>> No needs of arranging days of workshops with end users for feedback.

API consumers can use Anypoint Exchange features on the platform and interact with the API using its mocking feature. The feedback can be shared quickly on the same to incorporate any changes.

**NEW QUESTION 59**

A System API is designed to retrieve data from a backend system that has scalability challenges. What API policy can best safeguard the backend system?

- A. IPwhitelist
- B. SLA-based rate limiting
- C. Auth 2 token enforcement
- D. Client ID enforcement

**Answer:** B

**Explanation:**

Correct Answer

SLA-based rate limiting

\*\*\*\*\*

>> Client Id enforcement policy is a "Compliance" related NFR and does not help in maintaining the "Quality of Service (QoS)". It CANNOT and NOT meant for protecting the backend systems from scalability challenges.

>> IP Whitelisting and OAuth 2.0 token enforcement are "Security" related NFRs and again does not help in maintaining the "Quality of Service (QoS)". They CANNOT and are NOT meant for protecting the backend systems from scalability challenges.

Rate Limiting, Rate Limiting-SLA, Throttling, Spike Control are the policies that are "Quality of Service (QOS)" related NFRs and are meant to help in protecting the backend systems from getting overloaded.

<https://dzone.com/articles/how-to-secure-apis>

**NEW QUESTION 63**

An organization has several APIs that accept JSON data over HTTP POST. The APIs are all publicly available and are associated with several mobile applications and web applications.

The organization does NOT want to use any authentication or compliance policies for these APIs, but at the same time, is worried that some bad actor could send payloads that could somehow compromise the applications or servers running the API implementations.

What out-of-the-box Anypoint Platform policy can address exposure to this threat?

- A. Shut out bad actors by using HTTPS mutual authentication for all API invocations
- B. Apply an IP blacklist policy to all APIs; the blacklist will Include all bad actors
- C. Apply a Header injection and removal policy that detects the malicious data before it is used
- D. Apply a JSON threat protection policy to all APIs to detect potential threat vectors

**Answer:** D

**Explanation:**

Correct Answer

Apply a JSON threat protection policy to all APIs to detect potential threat vectors

\*\*\*\*\*

>> Usually, if the APIs are designed and developed for specific consumers (known consumers/customers) then we would IP Whitelist the same to ensure that traffic only comes from them.

>> However, as this scenario states that the APIs are publicly available and being used by so many mobile and web applications, it is NOT possible to identify and blacklist all possible bad actors.

>> So, JSON threat protection policy is the best chance to prevent any bad JSON payloads from such bad actors.

**NEW QUESTION 64**

What is the main change to the IT operating model that MuleSoft recommends to organizations to improve innovation and clock speed?

- A. Drive consumption as much as production of assets; this enables developers to discover and reuse assets from other projects and encourages standardization
- B. Expose assets using a Master Data Management (MDM) system; this standardizes projects and enables developers to quickly discover and reuse assets from other projects
- C. Implement SOA for reusable APIs to focus on production over consumption; this standardizes on XML and WSDL formats to speed up decision making
- D. Create a lean and agile organization that makes many small decisions everyday; this speeds up decision making and enables each line of business to take ownership of its projects

**Answer:** A

**Explanation:**

Correct Answer

Drive consumption as much as production of assets; this enables developers to discover and reuse assets from other projects and encourages standardization  
\*\*\*\*\*

>> The main motto of the new IT Operating Model that MuleSoft recommends and made popular is to change the way that they are delivered from a production model to a production + consumption model, which is done through an API strategy called API-led connectivity.

>> The assets built should also be discoverable and self-serveable for reusability across LOBs and organization.

>> MuleSoft's IT operating model does not talk about SDLC model (Agile/ Lean etc) or MDM at all. So, options suggesting these are not valid.

References:

<https://blogs.mulesoft.com/biz/connectivity/what-is-a-center-for-enablement-c4e/> <https://www.mulesoft.com/resources/api/secret-to-managing-it-projects>

#### NEW QUESTION 69

A company has started to create an application network and is now planning to implement a Center for Enablement (C4E) organizational model. What key factor would lead the company to decide upon a federated rather than a centralized C4E?

- A. When there are a large number of existing common assets shared by development teams
- B. When various teams responsible for creating APIs are new to integration and hence need extensive training
- C. When development is already organized into several independent initiatives or groups
- D. When the majority of the applications in the application network are cloud based

**Answer: C**

#### Explanation:

Correct Answer

When development is already organized into several independent initiatives or groups  
\*\*\*\*\*

>> It would require lot of process effort in an organization to have a single C4E team coordinating with multiple already organized development teams which are into several independent initiatives. A single C4E works well with different teams having at least a common initiative. So, in this scenario, federated C4E works well instead of centralized C4E.

#### NEW QUESTION 70

A company wants to move its Mule API implementations into production as quickly as possible. To protect access to all Mule application data and metadata, the company requires that all Mule applications be deployed to the company's customer-hosted infrastructure within the corporate firewall. What combination of runtime plane and control plane options meets these project lifecycle goals?

- A. Manually provisioned customer-hosted runtime plane and customer-hosted control plane
- B. MuleSoft-hosted runtime plane and customer-hosted control plane
- C. Manually provisioned customer-hosted runtime plane and MuleSoft-hosted control plane
- D. iPaaS provisioned customer-hosted runtime plane and MuleSoft-hosted control plane

**Answer: A**

#### Explanation:

Correct Answer

Manually provisioned customer-hosted runtime plane and customer-hosted control plane  
\*\*\*\*\*

There are two key factors that are to be taken into consideration from the scenario given in the question.

>> Company requires both data and metadata to be resided within the corporate firewall

>> Company would like to go with customer-hosted infrastructure.

Any deployment model that is to deal with the cloud directly or indirectly (Mulesoft-hosted or Customer's own cloud like Azure, AWS) will have to share atleast the metadata.

Application data can be controlled inside firewall by having Mule Runtimes on customer hosted runtime plane. But if we go with Mulsoft-hosted/ Cloud-based control plane, the control plane required atleast some minimum level of metadata to be sent outside the corporate firewall.

As the customer requirement is pretty clear about the data and metadata both to be within the corporate firewall, even though customer wants to move to production as quickly as possible, unfortunately due to the nature of their security requirements, they have no other option but to go with manually provisioned customer-hosted runtime plane and customer-hosted control plane.

#### NEW QUESTION 73

An organization makes a strategic decision to move towards an IT operating model that emphasizes consumption of reusable IT assets using modern APIs (as defined by MuleSoft).

What best describes each modern API in relation to this new IT operating model?

- A. Each modern API has its own software development lifecycle, which reduces the need for documentation and automation
- B. Each modern API must be treated like a product and designed for a particular target audience (for instance, mobile app developers)
- C. Each modern API must be easy to consume, so should avoid complex authentication mechanisms such as SAML or JWT D
- D. Each modern API must be REST and HTTP based

**Answer: B**

#### Explanation:

Correct Answers

\* 1. Each modern API must be treated like a product and designed for a particular target audience (for instance mobile app developers)  
\*\*\*\*\*

Bottom of Form Top of Form

#### NEW QUESTION 76

The implementation of a Process API must change.

What is a valid approach that minimizes the impact of this change on API clients?

- A. Update the RAML definition of the current Process API and notify API client developers by sending them links to the updated RAML definition
- B. Postpone changes until API consumers acknowledge they are ready to migrate to a new Process API or API version
- C. Implement required changes to the Process API implementation so that whenever possible, the Process API's RAML definition remains unchanged
- D. Implement the Process API changes in a new API implementation, and have the old API implementation return an HTTP status code 301 - Moved Permanently to inform API clients they should be calling the new API implementation

**Answer:** C

**Explanation:**

Correct Answer

Implement required changes to the Process API implementation so that, whenever possible, the Process API's RAML definition remains unchanged.

\*\*\*\*\* Key requirement in the question is:

>> Approach that minimizes the impact of this change on API clients Based on above:

>> Updating the RAML definition would possibly impact the API clients if the changes require any thing mandatory from client side. So, one should try to avoid doing that until really necessary.

>> Implementing the changes as a completely different API and then redirectly the clients with 3xx status code is really upsetting design and heavily impacts the API clients.

>> Organisations and IT cannot simply postpone the changes required until all API consumers acknowledge they are ready to migrate to a new Process API or API version. This is unrealistic and not possible.

The best way to handle the changes always is to implement required changes to the API implementations so that, whenever possible, the API's RAML definition remains unchanged.

**NEW QUESTION 78**

What is true about where an API policy is defined in Anypoint Platform and how it is then applied to API instances?

- A. The API policy Is defined In Runtime Manager as part of the API deployment to a Mule runtime, and then ONLY applied to the specific API Instance
- B. The API policy Is defined In API Manager for a specific API Instance, and then ONLY applied to the specific API instance
- C. The API policy Is defined in API Manager and then automatically applied to ALL API instances
- D. The API policy is defined in API Manager, and then applied to ALL API instances in the specified environment

**Answer:** B

**Explanation:**

Correct Answer

The API policy is defined in API Manager for a specific API instance, and then ONLY applied to the specific API instance.

\*\*\*\*\*

>> Once our API specifications are ready and published to Exchange, we need to visit API Manager and register an API instance for each API.

>> API Manager is the place where management of API aspects takes place like addressing NFRs by enforcing policies on them.

>> We can create multiple instances for a same API and manage them differently for different purposes.

>> One instance can have a set of API policies applied and another instance of same API can have different set of policies applied for some other purpose.

>> These APIs and their instances are defined PER environment basis. So, one need to manage them seperately in each environment.

>> We can ensure that same configuration of API instances (SLAs, Policies etc..) gets promoted when promoting to higher environments using platform feature.

But this is optional only. Still one can change them per environment basis if they have to.

>> Runtime Manager is the place to manage API Implementations and their Mule Runtimes but NOT APIs itself. Though API policies gets executed in Mule Runtimes, We CANNOT enforce API policies in Runtime Manager. We would need to do that via API Manager only for a cherry picked instance in an environment. So, based on these facts, right statement in the given choices is - "The API policy is defined in API Manager for a specific API instance, and then ONLY applied to the specific API instance".

**NEW QUESTION 79**

An organization has implemented a Customer Address API to retrieve customer address information. This API has been deployed to multiple environments and has been configured to enforce client IDs everywhere.

A developer is writing a client application to allow a user to update their address. The developer has found the Customer Address API in Anypoint Exchange and wants to use it in their client application.

What step of gaining access to the API can be performed automatically by Anypoint Platform?

- A. Approve the client application request for the chosen SLA tier
- B. Request access to the appropriate API Instances deployed to multiple environments using the client application's credentials
- C. Modify the client application to call the API using the client application's credentials
- D. Create a new application in Anypoint Exchange for requesting access to the API

**Answer:** A

**Explanation:**

Correct Answer

Approve the client application request for the chosen SLA tier

\*\*\*\*\*

>> Only approving the client application request for the chosen SLA tier can be automated

>> Rest of the provided options are not valid

**NEW QUESTION 80**

Refer to the exhibit.

A developer is building a client application to invoke an API deployed to the STAGING environment that is governed by a client ID enforcement policy. What is required to successfully invoke the API?

- A. The client ID and secret for the Anypoint Platform account owning the API in the STAGING environment
- B. The client ID and secret for the Anypoint Platform account's STAGING environment
- C. The client ID and secret obtained from Anypoint Exchange for the API instance in the STAGING environment

D. A valid OAuth token obtained from Anypoint Platform and its associated client ID and secret

**Answer: C**

**Explanation:**

Correct Answer

The client ID and secret obtained from Anypoint Exchange for the API instance in the STAGING environment

\*\*\*\*\*

>> We CANNOT use the client ID and secret of Anypoint Platform account or any individual environments for accessing the APIs

>> As the type of policy that is enforced on the API in question is "Client ID Enforcement Policy", OAuth token based access won't work.

Right way to access the API is to use the client ID and secret obtained from Anypoint Exchange for the API instance in a particular environment we want to work on.

References:

Managing API instance Contracts on API Manager <https://docs.mulesoft.com/api-manager/1.x/request-access-to-api-task> <https://docs.mulesoft.com/exchange/to-request-access> <https://docs.mulesoft.com/api-manager/2.x/policy-mule3-client-id-based-policies>

**NEW QUESTION 84**

A system API is deployed to a primary environment as well as to a disaster recovery (DR) environment, with different DNS names in each environment. A process API is a client to the system API and is being rate limited by the system API, with different limits in each of the environments. The system API's DR environment provides only 20% of the rate limiting offered by the primary environment. What is the best API fault-tolerant invocation strategy to reduce overall errors in the process API, given these conditions and constraints?

A. Invoke the system API deployed to the primary environment; add timeout and retry logic to the process API to avoid intermittent failures; if it still fails, invoke the system API deployed to the DR environment

B. Invoke the system API deployed to the primary environment; add retry logic to the process API to handle intermittent failures by invoking the system API deployed to the DR environment

C. In parallel, invoke the system API deployed to the primary environment and the system API deployed to the DR environment; add timeout and retry logic to the process API to avoid intermittent failures; add logic to the process API to combine the results

D. Invoke the system API deployed to the primary environment; add timeout and retry logic to the process API to avoid intermittent failures; if it still fails, invoke a copy of the process API deployed to the DR environment

**Answer: A**

**Explanation:**

Correct Answer

Invoke the system API deployed to the primary environment; add timeout and retry logic to the process API to avoid intermittent failures; if it still fails, invoke the system API deployed to the DR environment

\*\*\*\*\*

There is one important consideration to be noted in the question which is - System API in DR environment provides only 20% of the rate limiting offered by the primary environment. So, comparatively, very less calls will be allowed into the DR environment API opposed to its primary environment. With this in mind, let's analyse what is the right and best fault-tolerant invocation strategy.

\* 1. Invoking both the system APIs in parallel is definitely NOT a feasible approach because of the 20% limitation we have on DR environment. Calling in parallel every time would easily and quickly exhaust the rate limits on DR environment and may not give chance to genuine intermittent error scenarios to let in during the time of need.

\* 2. Another option given is suggesting to add timeout and retry logic to process API while invoking primary environment's system API. This is good so far. However, when all retries failed, the option is suggesting to invoke the copy of process API on DR environment which is not right or recommended. Only system API is the one to be considered for fallback and not the whole process API. Process APIs usually have lot of heavy orchestration calling many other APIs which we do not want to repeat again by calling DR's process API. So this option is NOT right.

\* 3. One more option given is suggesting to add the retry (no timeout) logic to process API to directly retry on DR environment's system API instead of retrying the primary environment system API first. This is not at all a proper fallback. A proper fallback should occur only after all retries are performed and exhausted on Primary environment first. But here, the option is suggesting to directly retry fallback API on first failure itself without trying main API. So, this option is NOT right too.

This leaves us one option which is right and best fit.

- Invoke the system API deployed to the primary environment

- Add Timeout and Retry logic on it in process API

- If it fails even after all retries, then invoke the system API deployed to the DR environment.

**NEW QUESTION 87**

An organization has created an API-led architecture that uses various API layers to integrate mobile clients with a backend system. The backend system consists of a number of specialized components and can be accessed via a REST API. The process and experience APIs share the same bounded-context model that is different from the backend data model. What additional canonical models, bounded-context models, or anti-corruption layers are best added to this architecture to help process data consumed from the backend system?

A. Create a bounded-context model for every layer and overlap them when the boundary contexts overlap, letting API developers know about the differences between upstream and downstream data models

B. Create a canonical model that combines the backend and API-led models to simplify and unify data models, and minimize data transformations.

C. Create a bounded-context model for the system layer to closely match the backend data model, and add an anti-corruption layer to let the different bounded contexts cooperate across the system and process layers

D. Create an anti-corruption layer for every API to perform transformation for every data model to match each other, and let data simply travel between APIs to avoid the complexity and overhead of building canonical models

**Answer: C**

**Explanation:**

Correct Answer

Create a bounded-context model for the system layer to closely match the backend data model, and add an anti-corruption layer to let the different bounded contexts cooperate across the system and process layers

\*\*\*\*\*

>> Canonical models are not an option here as the organization has already put in efforts and created bounded-context models for Experience and Process APIs.

>> Anti-corruption layers for ALL APIs is unnecessary and invalid because it is mentioned that experience and process APIs share same bounded-context model.

It is just the System layer APIs that need to choose their approach now.

>> So, having an anti-corruption layer just between the process and system layers will work well. Also to speed up the approach, system APIs can mimic the backend system data model.

#### NEW QUESTION 89

.....

## Relate Links

**100% Pass Your MCPA-Level-1 Exam with ExamBible Prep Materials**

<https://www.exambible.com/MCPA-Level-1-exam/>

## Contact us

We are proud of our high-quality customer service, which serves you around the clock 24/7.

Viste - <https://www.exambible.com/>