



MuleSoft

Exam Questions MCPA-Level-1

MuleSoft Certified Platform Architect - Level 1

NEW QUESTION 1

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 2

A company has created a successful enterprise data model (EDM). The company is committed to building an application network by adopting modern APIs as a core enabler of the company's IT operating model. At what API tiers (experience, process, system) should the company require reusing the EDM when designing modern API data models?

- A. At the experience and process tiers
- B. At the experience and system tiers
- C. At the process and system tiers
- D. At the experience, process, and system tiers

Answer: C

Explanation:

Correct Answer

At the process and system tiers

>> Experience Layer APIs are modeled and designed exclusively for the end user's experience. So, the data models of experience layer vary based on the nature and type of such API consumer. For example, Mobile consumers will need light-weight data models to transfer with ease on the wire, where as web-based consumers will need detailed data models to render most of the info on web pages, so on. So, enterprise data models fit for the purpose of canonical models but not of good use for experience APIs.
>> That is why, EDMs should be used extensively in process and system tiers but NOT in experience tier.

NEW QUESTION 3

An organization uses various cloud-based SaaS systems and multiple on-premises systems. The on-premises systems are an important part of the organization's application network and can only be accessed from within the organization's intranet.

What is the best way to configure and use Anypoint Platform to support integrations with both the cloud-based SaaS systems and on-premises systems?

- A) Use CloudHub-deployed Mule runtimes in an Anypoint VPC managed by Anypoint Platform Private Cloud Edition control plane
- B) Use CloudHub-deployed Mule runtimes in the shared worker cloud managed by the MuleSoft-hosted Anypoint Platform control plane
- C) Use an on-premises installation of Mule runtimes that are completely isolated with NO external network access, managed by the Anypoint Platform Private Cloud Edition control plane
- D) Use a combination of Cloud Hub-deployed and manually provisioned on-premises Mule runtimes managed by the MuleSoft-hosted Anypoint Platform control plane

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

Answer: B

Explanation:

Correct Answer

Use a combination of CloudHub-deployed and manually provisioned on-premises Mule runtimes managed by the MuleSoft-hosted Platform control plane.

***** Key details to be taken from the given scenario:

>> Organization uses BOTH cloud-based and on-premises systems
>> On-premises systems can only be accessed from within the organization's intranet Let us evaluate the given choices based on above key details:
>> CloudHub-deployed Mule runtimes can ONLY be controlled using MuleSoft-hosted control plane. We CANNOT use Private Cloud Edition's control plane to control CloudHub Mule Runtimes. So, option suggesting this is INVALID
>> Using CloudHub-deployed Mule runtimes in the shared worker cloud managed by the MuleSoft-hosted Anypoint Platform is completely IRRELEVANT to given scenario and silly choice. So, option suggesting this is INVALID
>> Using an on-premises installation of Mule runtimes that are completely isolated with NO external network access, managed by the Anypoint Platform Private Cloud Edition control plane would work for On-premises integrations. However, with NO external access, integrations cannot be done to SaaS-based apps. Moreover CloudHub-hosted apps are best-fit for integrating with SaaS-based applications. So, option suggesting this is BEST WAY.
The best way to configure and use Anypoint Platform to support these mixed/hybrid integrations is to use a combination of CloudHub-deployed and manually provisioned on-premises Mule runtimes managed by the MuleSoft-hosted Platform control plane.

NEW QUESTION 4

Refer to the exhibit.

what is true when using customer-hosted Mule runtimes with the MuleSoft-hosted Anypoint Platform control plane (hybrid deployment)?

- A. Anypoint Runtime Manager initiates a network connection to a Mule runtime in order to deploy Mule applications
- B. The MuleSoft-hosted Shared Load Balancer can be used to load balance API invocations to the Mule runtimes
- C. API implementations can run successfully in customer-hosted Mule runtimes, even when they are unable to communicate with the control plane
- D. Anypoint Runtime Manager automatically ensures HA in the control plane by creating a new Mule runtime instance in case of a node failure

Answer: C

Explanation:

Correct Answer

API implementations can run successfully in customer-hosted Mule runtimes, even when they are unable to communicate with the control plane.

>> We CANNOT use Shared Load balancer to load balance APIs on customer hosted runtimes

>> For Hybrid deployment models, the on-premises are first connected to Runtime Manager using Runtime Manager agent. So, the connection is initiated first from On-premises to Runtime Manager. Then all control can be done from Runtime Manager.
>> Anypoint Runtime Manager CANNOT ensure automatic HA. Clusters/Server Groups etc should be configured before hand.
Only TRUE statement in the given choices is, API implementations can run successfully in customer-hosted Mule runtimes, even when they are unable to communicate with the control plane. There are several references below to justify this statement.

References:

<https://docs.mulesoft.com/runtime-manager/deployment-strategies#hybrid-deployments> <https://help.mulesoft.com/s/article/On-Premise-Runtimes-Disconnected-From-US-Control-Plane-June-18th-201> <https://help.mulesoft.com/s/article/Runtime-Manager-cannot-manage-On-Prem-Applications-and-Servers-from->
<https://help.mulesoft.com/s/article/On-premise-Runtimes-Appear-Disconnected-in-Runtime-Manager-May-29th>

=====

=====

NEW QUESTION 5

Which of the following best fits the definition of API-led connectivity?

- A. API-led connectivity is not just an architecture or technology but also a way to organize people and processes for efficient IT delivery in the organization
- B. API-led connectivity is a 3-layered architecture covering Experience, Process and System layers
- C. API-led connectivity is a technology which enabled us to implement Experience, Process and System layer based APIs

Answer: A

Explanation:

Correct Answer

API-led connectivity is not just an architecture or technology but also a way to organize people and processes for efficient IT delivery in the organization.

NEW QUESTION 6

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 asvarious operations/resources
- B. Implement different system APIs named createCustomer, amendCustomer, retrieveCustomer and suspendCustomer as they are modular and has seperation of concerns
- C. Implement different system APIs named createCustomerInCRMZ, amendCustomerInCRMZ, retrieveCustomerFromCRMZ and suspendCustomerInCRMZ as they are modular and has seperation of concerns

Answer: B

Explanation:

Correct Answer

Implement different system APIs named createCustomer, amendCustomer, retrieveCustomer and suspendCustomer as they are modular and has seperation 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 7

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 (siloeD) 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 8

Which of the below, when used together, makes the IT Operational Model effective?

- A. Create reusable assets, Do marketing on the created assets across organization, Arrange time to time LOB reviews to ensure assets are being consumed or not
- B. Create reusable assets, Make them discoverable so that LOB teams can self-serve and browse the APIs, Get active feedback and usage metrics
- C. Create reusable assets, make them discoverable so that LOB teams can self-serve and browse the APIs

Answer: C

Explanation:

Correct Answer

Create reusable assets, Make them discoverable so that LOB teams can self-serve and browse the APIs, Get active feedback and usage metrics.

***** Diagram, arrow Description automatically generated

NEW QUESTION 9

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 10

An API client calls one method from an existing API implementation. The API implementation is later updated. What change to the API implementation would require the API client's invocation logic to also be updated?

- A. When the data type of the response is changed for the method called by the API client
- B. When a new method is added to the resource used by the API client
- C. When a new required field is added to the method called by the API client
- D. When a child method is added to the method called by the API client

Answer: C

Explanation:

Correct Answer

When a new required field is added to the method called by the API client

>> Generally, the logic on API clients need to be updated when the API contract breaks.

>> When a new method or a child method is added to an API , the API client does not break as it can still continue to use its existing method. So these two options are out.

>> We are left for two more where "datatype of the response if changed" and "a new required field is added".

>> Changing the datatype of the response does break the API contract. However, the question is insisting on the "invocation" logic and not about the response handling logic. The API client can still invoke the API successfully and receive the response but the response will have a different datatype for some field.

>> Adding a new required field will break the API's invocation contract. When adding a new required field, the API contract breaks the RAML or API spec agreement that the API client/API consumer and API provider has between them. So this requires the API client invocation logic to also be updated.

NEW QUESTION 10

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 13

When must an API implementation be deployed to an Anypoint VPC?

- A. When the API Implementation must invoke publicly exposed services that are deployed outside of CloudHub in a customer- managed AWS instance
- B. When the API implementation must be accessible within a subnet of a restricted customer-hosted network that does not allow public access
- C. When the API implementation must be deployed to a production AWS VPC using the Mule Maven plugin
- D. When the API Implementation must write to a persistent Object Store

Answer: A

NEW QUESTION 14

An API experiences a high rate of client requests (TPS) with small message payloads. How can usage limits be imposed on the API based on the type of client application?

- A. Use an SLA-based rate limiting policy and assign a client application to a matching SLA tier based on its type
- B. Use a spike control policy that limits the number of requests for each client application type
- C. Use a cross-origin resource sharing (CORS) policy to limit resource sharing between client applications, configured by the client application type
- D. Use a rate limiting policy and a client ID enforcement policy, each configured by the client application type

Answer: A

Explanation:

Correct Answer

Use an SLA-based rate limiting policy and assign a client application to a matching SLA tier based on its type.

>> SLA tiers will come into play whenever any limits to be imposed on APIs based on client type

NEW QUESTION 18

Traffic is routed through an API proxy to an API implementation. The API proxy is managed by API Manager and the API implementation is deployed to a CloudHub VPC using Runtime Manager. API policies have been applied to this API. In this deployment scenario, at what point are the API policies enforced on incoming API client requests?

- A. At the API proxy
- B. At the API implementation
- C. At both the API proxy and the API implementation
- D. At a MuleSoft-hosted load balancer

Answer: A

Explanation:

Correct Answer

At the API proxy

>> API Policies can be enforced at two places in Mule platform.

>> One - As an Embedded Policy enforcement in the same Mule Runtime where API implementation is running.

>> Two - On an API Proxy sitting in front of the Mule Runtime where API implementation is running.

>> As the deployment scenario in the question has API Proxy involved, the policies will be enforced at the API Proxy.

NEW QUESTION 22

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 designing an upstream API and its implementation, the development team has been advised to NOT set timeouts when invoking a downstream API, because that downstream API has no SLA that can be relied upon. This is the only downstream API dependency of that upstream API. Assume the downstream API runs uninterrupted without crashing. What is the impact of this advice?

- A. An SLA for the upstream API CANNOT be provided
- B. The invocation of the downstream API will run to completion without timing out
- C. A default timeout of 500 ms will automatically be applied by the Mule runtime in which the upstream API implementation executes
- D. A load-dependent timeout of less than 1000 ms will be applied by the Mule runtime in which the downstream API implementation executes

Answer: A

Explanation:

Correct Answer

An SLA for the upstream API CANNOT be provided.

>> First thing first, the default HTTP response timeout for HTTP connector is 10000 ms (10 seconds). NOT 500 ms.

>> Mule runtime does NOT apply any such "load-dependent" timeouts. There is no such behavior currently in Mule.

>> As there is default 10000 ms time out for HTTP connector, we CANNOT always guarantee that the invocation of the downstream API will run to completion without timing out due to its unreliable SLA times. If the response time crosses 10 seconds then the request may time out.

The main impact due to this is that a proper SLA for the upstream API CANNOT be provided.

NEW QUESTION 29

An API has been updated in Anypoint Exchange by its API producer from version 3.1.1 to 3.2.0 following accepted semantic versioning practices and the changes have been communicated via the API's public portal.
The API endpoint does NOT change in the new version.
How should the developer of an API client respond to this change?

- A. The update should be identified as a project risk and full regression testing of the functionality that uses this API should be run
- B. The API producer should be contacted to understand the change to existing functionality
- C. The API producer should be requested to run the old version in parallel with the new one
- D. The API client code ONLY needs to be changed if it needs to take advantage of new features

Answer: D

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 36

What CANNOT be effectively enforced using an API policy in Anypoint Platform?

- A. Guarding against Denial of Service attacks
- B. Maintaining tamper-proof credentials between APIs
- C. Logging HTTP requests and responses
- D. Backend system overloading

Answer: A

Explanation:

Correct Answer

Guarding against Denial of Service attacks

>> Backend system overloading can be handled by enforcing "Spike Control Policy"

>> Logging HTTP requests and responses can be done by enforcing "Message Logging Policy"

>> Credentials can be tamper-proofed using "Security" and "Compliance" Policies

However, unfortunately, there is no proper way currently on Anypoint Platform to guard against DOS attacks.

NEW QUESTION 40

An organization wants to make sure only known partners can invoke the organization's APIs. To achieve this security goal, the organization wants to enforce a Client ID Enforcement policy in API Manager so that only registered partner applications can invoke the organization's APIs. In what type of API implementation does MuleSoft recommend adding an API proxy to enforce the Client ID Enforcement policy, rather than embedding the policy directly in the application's JVM?

- A. A Mule 3 application using APIkit
- B. A Mule 3 or Mule 4 application modified with custom Java code
- C. A Mule 4 application with an API specification
- D. A Non-Mule application

Answer: D

Explanation:

Correct Answer

A Non-Mule application

>> All type of Mule applications (Mule 3/ Mule 4/ with APIkit/ with Custom Java Code etc) running on Mule Runtimes support the Embedded Policy Enforcement on them.

>> The only option that cannot have or does not support embedded policy enforcement and must have API Proxy is for Non-Mule Applications.

So, Non-Mule application is the right answer.

NEW QUESTION 42

What API policy would be LEAST LIKELY used when designing an Experience API that is intended to work with a consumer mobile phone or tablet application?

- A. OAuth 2.0 access token enforcement
- B. Client ID enforcement
- C. JSON threat protection
- D. IPwhitelist

Answer: D

Explanation:

Correct Answer

IP whitelist

>> OAuth 2.0 access token and Client ID enforcement policies are VERY common to apply on Experience APIs as API consumers need to register and access the APIs using one of these mechanisms

>> JSON threat protection is also VERY common policy to apply on Experience APIs to prevent bad or suspicious payloads hitting the API implementations.

>> IP whitelisting policy is usually very common in Process and System APIs to only whitelist the IP range inside the local VPC. But also applied occasionally on some experience APIs where the End User/ API Consumers are FIXED.

>> When we know the API consumers upfront who are going to access certain Experience APIs, then we can request for static IPs from such consumers and whitelist them to prevent anyone else hitting the API.

However, the experience API given in the question/ scenario is intended to work with a consumer mobile phone or tablet application. Which means, there is no way we can know all possible IPs that are to be whitelisted as mobile phones and tablets can so many in number and any device in the city/state/country/globe.

So, It is very LEAST LIKELY to apply IP Whitelisting on such Experience APIs whose consumers are typically Mobile Phones or Tablets.

NEW QUESTION 46

The responses to some HTTP requests can be cached depending on the HTTP verb used in the request. According to the HTTP specification, for what HTTP verbs is this safe to do?

- A. PUT, POST, DELETE
- B. GET, HEAD, POST
- C. GET, PUT, OPTIONS
- D. GET, OPTIONS, HEAD

Answer: D

Explanation:

Correct Answer

GET, OPTIONS, HEAD

<http://restcookbook.com/HTTP%20Methods/idempotency/>

NEW QUESTION 51

What is true about automating interactions with Anypoint Platform using tools such as Anypoint Platform REST APIs, Anypoint CU, or the Mule Maven plugin?

- A. Access to Anypoint Platform APIs and Anypoint CU can be controlled separately through the roles and permissions in Anypoint Platform, so that specific users can get access to Anypoint CLI while others get access to the platform APIs
- B. Anypoint Platform APIs can ONLY automate interactions with CloudHub, while the Mule Maven plugin is required for deployment to customer-hosted Mule runtimes
- C. By default, the Anypoint CLI and Mule Maven plugin are NOT included in the Mule runtime, so are NOT available to be used by deployed Mule applications
- D. API policies can be applied to the Anypoint Platform APIs so that ONLY certain LOBs have access to specific functions

Answer: C

Explanation:

Correct Answer

By default, the Anypoint CLI and Mule Maven plugin are NOT included in the Mule runtime, so are NOT available to be used by deployed Mule applications

>> We CANNOT apply API policies to the Anypoint Platform APIs like we can do on our custom written API instances. So, option suggesting this is FALSE.

>> Anypoint Platform APIs can be used for automating interactions with both CloudHub and customer-hosted Mule runtimes. Not JUST the CloudHub. So, option opposing this is FALSE.

>> Mule Maven plugin is NOT mandatory for deployment to customer-hosted Mule runtimes. It just helps your CI/CD to have smoother automation. But not a compulsory requirement to deploy. So, option opposing this is FALSE.

>> We DO NOT have any such special roles and permissions on the platform to separately control access for some users to have Anypoint CLI and others to have Anypoint Platform APIs. With proper general roles/permissions (API Owner, Cloudhub Admin etc..), one can use any of the options (Anypoint CLI or Platform APIs). So, option suggesting this is FALSE.

Only TRUE statement given in the choices is that - Anypoint CLI and Mule Maven plugin are NOT included in the Mule runtime, so are NOT available to be used by deployed Mule applications.

Maven is part of Studio or you can use other Maven installation for development. CLI is convenience only. It is one of many ways how to install app to the runtime. These are definitely NOT part of anything except your process of deployment or automation.

NEW QUESTION 56

A new upstream API is being designed to offer an SLA of 500 ms median and 800 ms maximum (99th percentile) response time. The corresponding API implementation needs to sequentially invoke 3 downstream APIs of very similar complexity.

The first of these downstream APIs offers the following SLA for its response time: median: 100 ms, 80th percentile: 500 ms, 95th percentile: 1000 ms.

If possible, how can a timeout be set in the upstream API for the invocation of the first downstream API to meet the new upstream API's desired SLA?

- A. Set a timeout of 50 ms; this times out more invocations of that API but gives additional room for retries
- B. Set a timeout of 100 ms; that leaves 400 ms for the other two downstream APIs to complete
- C. No timeout is possible to meet the upstream API's desired SLA; a different SLA must be negotiated with the first downstream API or invoke an alternative API
- D. Do not set a timeout; the Invocation of this API is mandatory and so we must wait until it responds

Answer: B

Explanation:

Correct Answer

Set a timeout of 100ms; that leaves 400ms for other two downstream APIs to complete

***** Key details to take from the given scenario:

>> Upstream API's designed SLA is 500ms (median). Lets ignore maximum SLA response times.

>> This API calls 3 downstream APIs sequentially and all these are of similar complexity.

>> The first downstream API is offering median SLA of 100ms, 80th percentile: 500ms; 95th percentile: 1000ms.

Based on the above details:

>> We can rule out the option which is suggesting to set 50ms timeout. Because, if the median SLA itself being offered is 100ms then most of the calls are going to timeout and time gets wasted in retried them and eventually gets exhausted with all retries. Even if some retries gets successful, the remaining time wont leave enough room for 2nd and 3rd downstream APIs to respond within time.

>> The option suggesting to NOT set a timeout as the invocation of this API is mandatory and so we must wait until it responds is silly. As not setting time out would go against the good implementation pattern and moreover if the first API is not responding within its offered median SLA 100ms then most probably it would either respond in 500ms (80th percentile) or 1000ms (95th percentile). In BOTH cases, getting a successful response from 1st downstream API does NO GOOD because already by this time the Upstream API SLA of 500 ms is breached. There is no time left to call 2nd and 3rd downstream APIs.

>> It is NOT true that no timeout is possible to meet the upstream APIs desired SLA.

As 1st downstream API is offering its median SLA of 100ms, it means MOST of the time we would get the responses within that time. So, setting a timeout of 100ms would be ideal for MOST calls as it leaves enough room of 400ms for remaining 2 downstream API calls.

NEW QUESTION 59

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 61

What is typically NOT a function of the APIs created within the framework called API-led connectivity?

- A. They provide an additional layer of resilience on top of the underlying backend system, thereby insulating clients from extended failure of these systems.
- B. They allow for innovation at the user Interface level by consuming the underlying assets without being aware of how data is being extracted from backend systems.
- C. They reduce the dependency on the underlying backend systems by helping unlock data from backend systems in a reusable and consumable way.
- D. They can compose data from various sources and combine them with orchestration logic to create higher level value.

Answer: A

Explanation:

Correct Answer

They provide an additional layer of resilience on top of the underlying backend system, thereby insulating clients from extended failure of these systems.

***** In API-led connectivity,

>> Experience APIs - allow for innovation at the user interface level by consuming the underlying assets without being aware of how data is being extracted from backend systems.

>> Process APIs - compose data from various sources and combine them with orchestration logic to create higher level value

>> System APIs - reduce the dependency on the underlying backend systems by helping unlock data from backend systems in a reusable and consumable way.

However, they NEVER promise that they provide an additional layer of resilience on top of the underlying backend system, thereby insulating clients from extended failure of these systems.

<https://dzone.com/articles/api-led-connectivity-with-mule>

NEW QUESTION 62

A REST API is being designed to implement a Mule application.

What standard interface definition language can be used to define REST APIs?

- A. Web Service Definition Language(WSDL)
- B. OpenAPI Specification (OAS)
- C. YAML
- D. AsyncAPI Specification

Answer: B

NEW QUESTION 66

Refer to the exhibit.

What is the best way to decompose one end-to-end business process into a collaboration of Experience, Process, and System APIs?

- A) Handle customizations for the end-user application at the Process API level rather than the Experience API level

- B) Allow System APIs to return data that is NOT currently required by the identified Process or Experience APIs
- C) Always use a tiered approach by creating exactly one API for each of the 3 layers (Experience, Process and System APIs)
- D) Use a Process API to orchestrate calls to multiple System APIs, but NOT to other Process APIs

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

Answer: B

Explanation:

Correct Answer

Allow System APIs to return data that is NOT currently required by the identified Process or Experience APIs.

>> All customizations for the end-user application should be handled in "Experience API" only. Not in Process API

>> We should use tiered approach but NOT always by creating exactly one API for each of the 3 layers. Experience APIs might be one but Process APIs and System APIs are often more than one. System APIs for sure will be more than one all the time as they are the smallest modular APIs built in front of end systems.
>> Process APIs can call System APIs as well as other Process APIs. There is no such anti-design pattern in API-Led connectivity saying Process APIs should not call other Process APIs.

So, the right answer in the given set of options that makes sense as per API-Led connectivity principles is to allow System APIs to return data that is NOT currently required by the identified Process or Experience APIs. This way, some future Process APIs can make use of that data from System APIs and we need NOT touch the System layer APIs again and again.

NEW QUESTION 68

What is true about API implementations when dealing with legal regulations that require all data processing to be performed within a certain jurisdiction (such as in the USA or the EU)?

- A. They must avoid using the Object Store as it depends on services deployed ONLY to the US East region
- B. They must use a Jurisdiction-local external messaging system such as Active MQ rather than Anypoint MQ
- C. They must be deployed to Anypoint Platform runtime planes that are managed by Anypoint Platform control planes, with both planes in the same Jurisdiction
- D. They must ensure ALL data is encrypted both in transit and at rest

Answer: C

Explanation:

Correct Answer

They must be deployed to Anypoint Platform runtime planes that are managed by Anypoint Platform control planes, with both planes in the same Jurisdiction.

>> As per legal regulations, all data processing to be performed within a certain jurisdiction. Meaning, the data in USA should reside within USA and should not go out. Same way, the data in EU should reside within EU and should not go out.

>> So, just encrypting the data in transit and at rest does not help to be compliant with the rules. We need to make sure that data does not go out too.

>> The data that we are talking here is not just about the messages that are published to Anypoint MQ. It includes the apps running, transaction states, application logs, events, metric info and any other metadata. So, just replacing Anypoint MQ with a locally hosted ActiveMQ does NOT help.

>> The data that we are talking here is not just about the key/value pairs that are stored in Object Store. It includes the messages published, apps running, transaction states, application logs, events, metric info and any other metadata. So, just avoiding using Object Store does NOT help.

>> The only option left and also the right option in the given choices is to deploy application on runtime and control planes that are both within the jurisdiction.

NEW QUESTION 72

Select the correct Owner-Layer combinations from below options

- A. * 1. App Developers owns and focuses on Experience Layer APIs* 2. Central IT owns and focuses on Process Layer APIs* 3. LOB IT owns and focuses on System Layer APIs
- B. * 1. Central IT owns and focuses on Experience Layer APIs* 2. LOB IT owns and focuses on Process Layer APIs* 3. App Developers owns and focuses on System Layer APIs
- C. * 1. App Developers owns and focuses on Experience Layer APIs* 2. LOB IT owns and focuses on Process Layer APIs* 3. Central IT owns and focuses on System Layer APIs

Answer: C

Explanation:

Correct Answer

* 1. App Developers owns and focuses on Experience Layer APIs

* 2. LOB IT owns and focuses on Process Layer APIs

* 3. Central IT owns and focuses on System Layer APIs

References:

<https://blogs.mulesoft.com/biz/api/experience-api-ownership/> <https://blogs.mulesoft.com/biz/api/process-api-ownership/> <https://blogs.mulesoft.com/biz/api/system-api-ownership/>

NEW QUESTION 76

What Anypoint Connectors support transactions?

- A. Database, JMS, VM
- B. Database, 3MS, HTTP
- C. Database, JMS, VM, SFTP
- D. Database, VM, File

Answer: A

NEW QUESTION 81

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 85

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 86

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. 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 87

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 89

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 Enforcment 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 90

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 95

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 97

.....

Thank You for Trying Our Product

We offer two products:

1st - We have Practice Tests Software with Actual Exam Questions

2nd - Questions and Answers in PDF Format

MCPA-Level-1 Practice Exam Features:

- * MCPA-Level-1 Questions and Answers Updated Frequently
- * MCPA-Level-1 Practice Questions Verified by Expert Senior Certified Staff
- * MCPA-Level-1 Most Realistic Questions that Guarantee you a Pass on Your FirstTry
- * MCPA-Level-1 Practice Test Questions in Multiple Choice Formats and Updatesfor 1 Year

100% Actual & Verified — Instant Download, Please Click
[Order The MCPA-Level-1 Practice Test Here](#)